Finding One Common Item, Privately

Tyler Beauregard[*] Janabel Xia[†] Mike Rosulek[‡]

May 10, 2022

Abstract

Private set intersection (PSI) allows two parties, who each hold a set of items, to learn which items they have in common, without revealing anything about their other items. Some applications of PSI would be better served by revealing only one common item, rather than the entire set of all common items. In this work we develop special-purpose protocols for privately finding one common item (FOCI) from the intersection of two sets. The protocols differ in how that item is chosen — *e.g.*, uniformly at random from the intersection; the "best" item in the intersection according to one party's ranking; or the "best" item in the intersection according to the sum of both party's scores.

1 Introduction

Suppose Alice and Bob want to schedule a meeting, without sharing their entire calendars with each other. One method they might use is **private set intersection (PSI)**. If they run a PSI protocol, with each party using the set of available time slots as their input, then they will learn only the set of common available times — *i.e.*, the intersection of those sets — and nothing else about their calendars.

However, for the application of scheduling a meeting, it is not necessary for them to learn the *entire intersection* of their availabilities. Instead, it is enough that they learn just *a single item* from the intersection. We refer to this problem as (privately) **finding one common item (FOCI)**. We may consider different ways that that single item may be chosen. The parties may want to simply learn a random common item. Alternatively, one or both parties may have preferences about the items (*e.g.*, "I am free at these times but prefer Tuesdays/Thursdays and prefer mornings.") and they want to learn the "best" item in the intersection according to those preferences.

1.1 Related Work

To the best of our knowledge, there has not been work studying this particular variant of PSI. We briefly recall the state of the art for plain PSI, and also discuss secure multi-party computation methods that could be used to achieve FOCI.

Plain PSI. The first PSI protocols date back to the classic Diffie-Hellman-based PSI of Huberman, Franklin, and Hogg [HFH99]. Their protocol has roots dating back to Meadows [Mea86]. Our protocols take inspiration from the protocol of Huberman, Franklin, and Hogg; we elaborate on this

---

[*] Truman State University, `trb4137@truman.edu`

[†] Massachusetts Institute of Technology, `janabel@mit.edu`

[‡] Oregon State University, `rosulekm@eecs.oregonstate.edu`

connection later. Many other protocols have built on this paradigm, improving its efficiency [JL10, RT21] and extending it to achieve security against malicious adversaries [DKT10, RT21]. Besides the Diffie-Hellman paradigm, there are other approaches for PSI — most notably, obliviius polynomial evaluation [KS05, Haz15] and oblivious transfer [PSZ14, PSSZ15, KKRT16, RR17, PRTY19, CM20, PRTY20, RR22].

PSI based on oblivious transfer is the most efficient for large sets, and the fastest PSI protocol in that paradigm is due to to Rindal and Raghuraman [RR22]. For small sets, PSI based on the Diffie-Hellman approach is more efficient, and the fastest protocol in that paradigm is due to Rosulek and Trieu [RT21]. In their work, they found that the Diffie-Hellman approach was faster for sets of around 500 items or fewer.

Computing on the intersection. Finding one common item is a special case of *computing arbitrary functions of the intersection.* There is a line of work on this problem, where some PSI techniques are used but the intersection is fed into a generic secure multi-party computation protocol [HEK12, PSSZ15, PSWW18, PSTY19, GMR+21].

1.2 Our Results

It is possible to privately find one common item, using the approaches just listed above (for computing arbitrary functions of the intersection).[1] However, we point out two issues with these approaches:

1. They all use techniques from oblivious-transfer-based PSI. These techniques are the most scalable for large sets, but they have certain inherent fixed costs (base OTs). In the case of plain PSI, these fixed costs are a significant fraction of the entire protocol cost for small sets. For this reason, Diffie-Hellman techniques are more efficient on small sets (in practice, several hundred items for each party).

   Our motivating application to calendar scheduling is indeed in this regime of set sizes, with ∼360 half-hour time slots in one month of business hours.

2. They all use general-purpose MPC (*e.g.*, garbled circuits or GMW protocol) to compute the function of the intersection. This adds an inherent level of complexity to the protocol. On the other hand, Diffie-Hellman PSI techniques are relatively simple. While describing a real-world and large-scale deployment of PSI, Ion *et al.* [IKN+19] explicitly listed *protocol simplicity* as a major design constraint, motivating simplicity as follows:

   > *It is difficult to overstate the importance of simplicity in a practical deployment, especially one involving multiple businesses. A simple protocol is easier to explain to the multiple stakeholders involved, and greatly eases the decision to use a new technology. It is also easier to implement without errors, test, audit for correctness, and modify. It is also often easier to optimize by parallelizing or performing in a distributed manner. Simplicity further helps long-term maintenance, since, as time passes, a constantly increasing group of people needs to understand the details of how a solution works.*

We propose simple protocols for the following variants of privately finding one common item:

---

[1] All protocols for computing functions of the intersection can be readily augmented to support data associated with the items, *e.g.*, scores/ranks.

- Alice learns the cardinality of the intersection and Bob learns one item chosen uniformly from the intersection.
- Bob has assigned ranks to each of his items, and he learns the item in the intersection with the highest rank. Alice learns the cardinality of the intersection, but nothing about Bob's ranks.
- Both parties have assigned scores to each of the items, and for every item in the intersection we define its *combined score* as the sum of Alice's and Bob's scores for that item. Bob learns the item in the intersection with the highest combined score. Alice learns the cardinality of the intersection and the (unordered) set of combined scores for items in the intersection — *i.e.*, she does not learn which scores are associated with specific items, and she does not learn the individual contributions of Alice's/Bob's scores to the combined scores.

All of our protocols are conceptually simple and practical. Each is proven secure against *semi-honest* adversaries, under the standard DDH assumption, and in the random oracle model. The second protocol (with only Bob ranking the items) requires an order-revealing encryption [BLR+15], but there exist compact ORE schemes based only the minimal assumption of a PRF [LW16].

Our protocols reveal more than the minimum amount of information — *i.e.*, more than just the identity of one common item. All three protocols leak the cardinality of the intersection to Alice, for example. However, each protocol hides non-trivial information about the sets; each protocol reveals nothing about items not in the intersection; and leakage about the intersection is disassociated from specific items.

2 Preliminaries

2.1 Decisional Diffie-Hellman Assumption

Definition 1. *Let $\mathbb{G}$ be a cyclic group with generator g and order q. The* ***decisional Diffie-Hellman (DDH) assumption*** *is that the following two distributions are indistinguishable:*

```
$\mathsf{DH}_{1,\mathbb{G}}$:
    $a, b \leftarrow \mathbb{Z}_q$
    return $(g^a, g^b, g^{ab})$
```

```
$\mathsf{Rand}_{1,\mathbb{G}}$:
    $a, b, c \leftarrow \mathbb{Z}_q$
    return $(g^a, g^b, g^c)$
```

Using a standard and straight-forward rerandomization technique ([Bon98]), the DDH assumption is equivalent to the following:

Proposition 2. *Let $\mathbb{G}$ be a cyclic group with generator g and order q. The DDH assumption is equivalent to the assumption that, for all n (polynomially bounded by the security parameter), the following two distributions are indistinguishable:*

```
$\mathsf{DH}_{n,\mathbb{G}}$:
    $a_1, \dots, a_n, b \leftarrow \mathbb{Z}_q$
    return $(g^{a_1}, \dots, g^{a_n}, g^b, g^{a_1 b}, \dots, g^{a_n b})$
```

```
$\mathsf{Rand}_{n,\mathbb{G}}$:
    $a_1, \dots, a_n, b, c_1, \dots, c_n \leftarrow \mathbb{Z}_q$
    return $(g^{a_1}, \dots, g^{a_n}, g^b, g^{c_1}, \dots, g^{c_n})$
```

2.2 Secure Two-Party Computation

In this work we consider secure two-party computation in the presence of semi-honest adversaries. Let the two parties be denoted P_1 and P_2, and let their private inputs be x_1 and x_2, respectively. Let $f(x_1, x_2) = (f_1(x_1, x_2), f_2(x_1, x_2))$ denote an ideal functionality, which receives x_1, x_2 from the parties and gives output $f_i(x_1, x_2)$ to party P_i.

Let $\mathit{view}_i^{\pi}(x_1, x_2)$ denote the view of party P_i (consisting of internal randomness and protocol messages received) when the parties run protocol π honestly, on respective inputs x_1 and x_2.

Definition 3. *A protocol π securely realizes a functionality $f = (f_1, f_2)$ if, for $i \in \{1, 2\}$ there exists a simulator $\mathcal{S}_i$ such that for all x_1, x_2, the distributions $\mathit{view}_i^{\pi}(x_1, x_2)$ and $\mathcal{S}_i(x_i, f_i(x_1, x_2))$ are indistinguishable.*

In other words, the view of party P_i can be simulated given only their input x_i and ideal output $f_i(x_1, x_2)$.

2.3 Symmetric-Key Encryption

We require a simple one-time, symmetric-key encryption scheme, where decryption fails if the wrong (independently random) key is used. Let $\mathcal{K}$ be the set of keys and let $\mathcal{M}$ be the set of plaintexts. Specifically, we require the following properties:

- Correctness: $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$ with probability 1 for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$.
- One-time security: For all $m_0, m_1 \in \mathcal{M}$, the distributions $\mathcal{E}_0$ and $\mathcal{E}_1$ are indistinguishable, where:

```
$\mathcal{E}_b$:
$k \leftarrow \mathcal{K}$
return $\mathsf{Enc}(k, m_b)$
```

- Robust decryption: For all $m \in \mathcal{M}$, the following process outputs TRUE with negligible probability:

```
$k, k' \leftarrow \mathcal{K}$
$c \leftarrow \mathsf{Enc}(k, m)$
return $\bot \neq \mathsf{Dec}(k', m)$
```

2.4 Order-Revealing Encryption

Order-revealing encryption (ORE) is a symmetric-key encryption scheme that reveals no more than the ordering of the plaintexts. See [BLR+15, LW16] for example constructions.

We specialize the notation of ORE for later convenience.

- Syntax: An ORE consists of algorithms $\mathsf{Enc}, \mathsf{Dec}, \mathsf{Argmax}$. The set of keys is $\mathcal{K}$ and the set of plaintexts is $\mathcal{M}$. Without loss of generality, $\mathcal{M} = \mathbb{Z}_N$ for some integer N, and we use the natural total ordering of $\mathbb{Z}_N$.
- Correctness: $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$ with probability 1 for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$.
- Order-revealing: $\mathsf{Argmax}(\mathsf{Enc}(k, m_1), \ldots, \mathsf{Enc}(k, m_n)) = \arg\max_j m_j$, with probability 1 for all $k \in \mathcal{K}$ and $m_1, \ldots, m_n \in \mathcal{M}$.

- Security: for all **distinct** $m_1, \ldots, m_n \in \mathcal{M}$, the following distributions are indistinguishable:

```
$\mathcal{D}_0$:
$k \leftarrow \mathcal{K}$
for $i = 1$ to $n$:
    $c_i = \mathsf{Enc}(k, \boxed{m_i})$
return shuffle($\{c_1, \ldots, c_n\}$)
```

```
$\mathcal{D}_1$:
$k \leftarrow \mathcal{K}$
for $i = 1$ to $n$:
    $c_i = \mathsf{Enc}(k, \boxed{i})$
return shuffle($\{c_1, \ldots, c_n\}$)
```

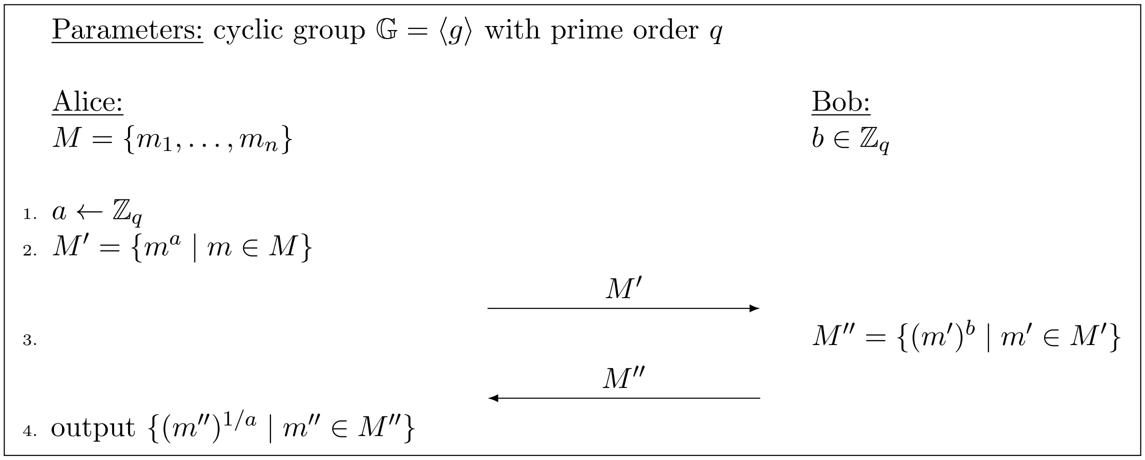
In other words, encryptions of distinct plaintexts are indistinguishable from encryptions of the sequence $1, \ldots, n$.

3 Finding a Random Item of the Intersection

Our first simple protocol allows Bob to learn a single, randomly chosen, item from the intersection, while Alice learns only the cardinality of the intersection. For simplicity, we present our protocols for the case where both parties have n items, but all of the protocols are easily generalized for the case where the parties have sets of different sizes.

3.1 Warmup: Cardinality-Only Protocol & Blind Exponentiation

We start by recalling the classic protocol of Huberman, Franklin, and Hogg [HFH99], which allows Alice & Bob to learn the cardinality of their intersection. The heart of the protocol is a *blind exponentiation* subprotocol, in which Alice has a set of items that get raised to a secret exponent known to Bob. Alice learns only the *unordered set* of resulting values. The subprotocol is shown in Figure 1.



Parameters: cyclic group $\mathbb{G} = \langle g \rangle$ with prime order q

Alice:
$M = \{m_1, \ldots, m_n\}$

Bob:
$b \in \mathbb{Z}_q$

1. $a \leftarrow \mathbb{Z}_q$
2. $M' = \{m^a \mid m \in M\}$

$\xrightarrow{M'}$

3. $M'' = \{(m')^b \mid m' \in M'\}$

$\xleftarrow{M''}$

4. output $\{(m'')^{1/a} \mid m'' \in M''\}$

Figure 1: Blind Exponentiation subprotocol.

Our convention when writing protocols and proving security is that **sets are unordered.** So when Alice/Bob send each other a set during the protocol, that set is assumed to be randomly permuted (or equivalently, the set is sorted since the sets turn out to contain pseudorandom values).

Lemma 4. *Alice's output is (the unordered set) $\{m^b \mid m \in M\}$. Furthermore, if Alice is semi-honest, then her view in Figure 1 can be simulated given only this output.*

Proof. Correctness follows from the fact that

$$\{(m'')^{1/a} \mid m'' \in M''\} = \{((m')^b)^{1/a} \mid m' \in M'\} = \{((m^a)^b)^{1/a} \mid m \in M\}.$$

Alice's view consists of M'' and random exponent a. This can be simulated by a simulator choosing random a and then raising every item of the output to the a power. □

Lemma 5. *If Bob is semi-honest, and Alice's inputs have the form $m_i = H(x_i)$, for distinct x_i values (chosen by the adversary), then Bob's view in Figure 1 is indistinguishable from random (assuming the DDH assumption, and with H a random oracle).*

Proof. Consider the following reduction algorithm. Given $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$: Simulate a random oracle while programming it as $H(x_i) = \alpha_i$ — this is possible since the x_i's are distinct. Then simulate Alice's message M' as $M' = \{\beta_1, \ldots, \beta_n\}$. If each $\beta_i = \alpha_i^a$ then Bob's view is exactly as in the protocol. If the β_i values are independently random, then Bob's view is of a random set M'. The two cases are indistinguishable from the DDH assumption (Section 2.1). □

Cardinality-only protocol. In the cardinality protocol of [HFH99], the parties first perform blind exponentiation. If Alice's input set is X, then her input to blind exponentiation subprotocol is $\{H(x_i) \mid x_i \in X\}$. She learns $X' = \{H(x_i)^b \mid x_i \in X\}$ where b is a random exponent chosen by Bob. Bob also sends $Y' = \{H(y_i)^b \mid y_i \in Y\}$, where Y is his input set. The cardinality $|X' \cap Y'|$ corresponds to the cardinality $|X \cap Y|$. The protocol corresponds to all but the last protocol message of Figure 3.

Since the outputs of the blind exponentiation subprotocol are randomly permuted, Alice does not know the correspondence betwen matching $H(z)^b$ values and her original x_i values.

3.2 Choosing a random item

After performing the basic cardinality protocol, Alice can simply identify a random element from the intersection according to its $H(z)^b$ value. In this way, Bob will learn a single item from the intersection, while Alice learns only the cardinality of intersection. For the sake of completeness, we describe the ideal functionality for this FOCI variant in Figure 2, and the protocol in Figure 3.

1. receive input X from Alice and Y from Bob.
2. give $|X \cap Y|$ to Alice.
3. sample $z^* \leftarrow X \cap Y$ uniformly; set $z^* = \perp$ if $X \cap Y = \emptyset$.
4. give z^* to Bob.

Figure 2: Ideal functionality for sampling a random item from the intersection.

Lemma 6. *The protocol in Figure 3 is correct.*

Proof. If $z \in X \cap Y$ then $H(z)^b$ will surely be included in A' and also as one of the K_i values. For all other items $x \neq y$, $\Pr[H(x)^b = H(y)^b] = \Pr[H(x) = H(y)]$ — *i.e.*, these items contribute to the intersection only in the case of a collision under the random oracle. □

Lemma 7. *The protocol in Figure 3 securely realizes Figure 2 against a semi-honest Bob.*

Proof. Simulation for Bob is trivial. Bob's view consists only of his view from BlindExp (which is indistinguishable from random), and the final protocol message j^*, which is trivially computable from his ideal output. □

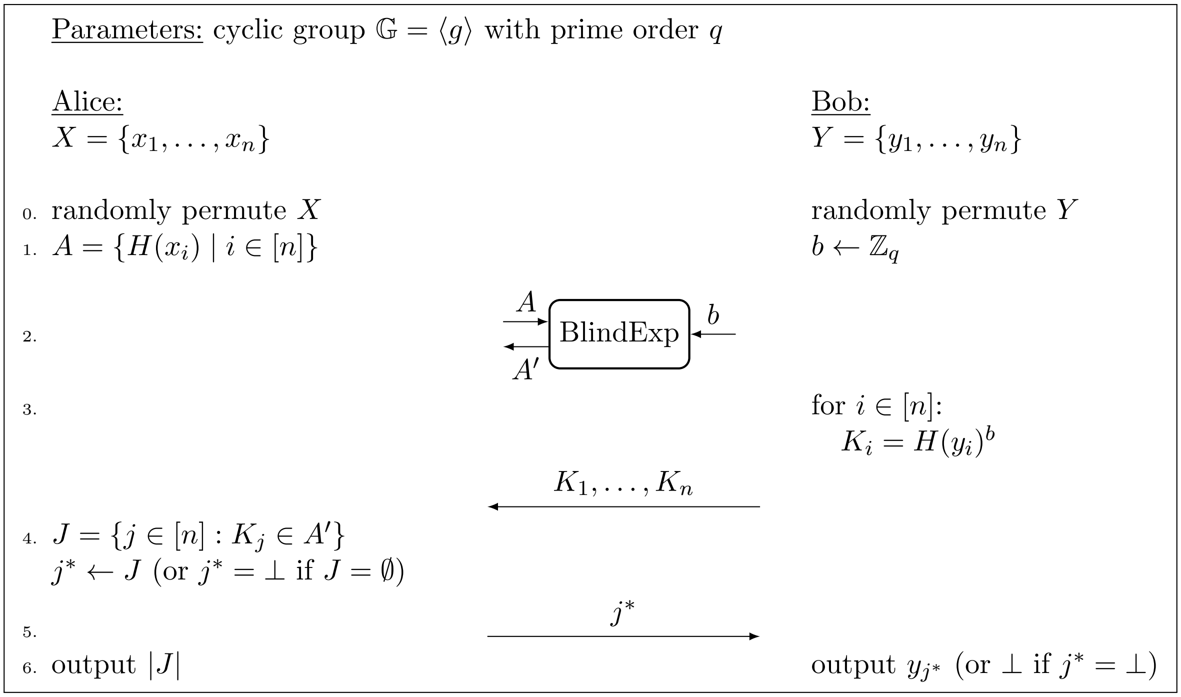


Figure 3: Protocol for identifying a random item from the intersection.

Lemma 8. *The protocol in Figure 3 securely realizes Figure 2 against a semi-honest Alice.*

Proof. Alice's view consists mainly of her output A' from the blind exponentiation subprotocol and the K_i values from Bob. Using a standard reduction from DDH (which programs the $H(x_i)$ and $H(y_i)$ values in the random oracle), all values of the form $H(z)^b$ are indistinguishable from random. For $z \in X \cap Y$, such $H(z)^b$ value appears in A' and as one of the K_i values. For $y \in Y \setminus X$, the corresponding $H(y)^b$ appears only as one of the K_i values. Since the A' set is unordered, and Bob's set is randomly permuted, then Alice's view can be simulated knowing only $|X \cap Y|$ — *i.e.*, knowing how many values repeat between A' and the set of K_i values. $\square$

4 Finding the Best Item According to a Unilateral Rank

In this section we consider the following variant. Alice holds a set of x_i values, and Bob holds a set of (y_i, v_i) pairs. The value v_i denotes Bob's *rank* of the item y_i — *i.e.*, a number between 1 and n. We consider the problem of identifying the common item with highest rank. We assume that Bob has assigned **distinct ranks** to each of the items in his set.

In this variant, Alice learns only how many common items they have (*i.e.*, the cardinality of the intersection). In particular, she does not learn anything about the relative rankings of items in the intersection vs items outside of the intersection (*e.g.*, she cannot learn that the intersection contains only Bob's least favorite items). Bob learns only the identity of the item with highest rank.

In Figure 4 we formally define the ideal functionality for this variant of sampling from the intersection. In the case that there are no common items, V will be empty. We use the following notational conventions for that case: if $V = \emptyset$ then $\max(V) = \perp$; if the value of $j^* = \perp$ then $y_{j^*} = \perp$.

```
receive input $X$ from Alice and $Y$ from Bob.
define $K(Y) = \{y \mid \exists v : (y, v) \in Y\}$
compute $V = \{v \mid \exists y : y \in X \wedge (y, v) \in Y\}$
give $|V|$ to Alice.
compute $v^* = \max(V)$ and find $y^*$ such that $(y^*, v^*) \in Y$
give $y^*$ to Bob.
```

Figure 4: Ideal functionality for sampling the best item from the intersection, according to a unilateral rank.

4.1 Intersection Protocol

The high-level idea behind the protocol is as follows. The parties can first perform the basic PSI-cardinality protocol from Section 3. This protocol computes a key $K_z = H(z)^b$ associated to each item z. Alice learns the key corresponding to every item in her set, but all other keys appear random to her.

Hence, Bob can use these keys to encrypt some information about his items' ranks. What should be the payload / associated data that Bob encrypts with each key? It should be enough to allow Alice to determine the highest ranked item in the intersection, without revealing that rank, and without revealing anything also about the relative ranks of items in the intersection.

The appropriate tool for the job is order-revealing encryption (ORE; Section 2.4). If Bob has item y with rank v, then he can use the PSI key K_v to *encrypt an ORE encryption of* v. Alice can therefore decrypt the outer ciphertexts to obtain ORE encryptions of the ranks of all items in the intersection. These ORE ciphertexts allow Alice to identify the item with highest rank, but they leak nothing else about the ranks.

Lemma 9. *The protocol in Figure 5 is correct.*

Proof. If $(y_i, v_i) \in Y$ and $y_i \in X$ then A' will contain $H(y_i)^b$. and we will also have $E_i = \mathsf{Enc}(H(y_i)^b, O_i)$. As such, Alice will eventually decrypt this E_i to obtain O_i, an ORE encryption of v_i. If $y_i \notin X$ then Alice will decrypt E_i with only independently generated keys, which will fail with overwhelming probability (cf. robust decryption Section 2.3). She will later compute $\mathsf{Argmax}(\{O_j\})$ which by the ORE correctness is the index j^* of the maximum v_j rank in the intersection. ☐

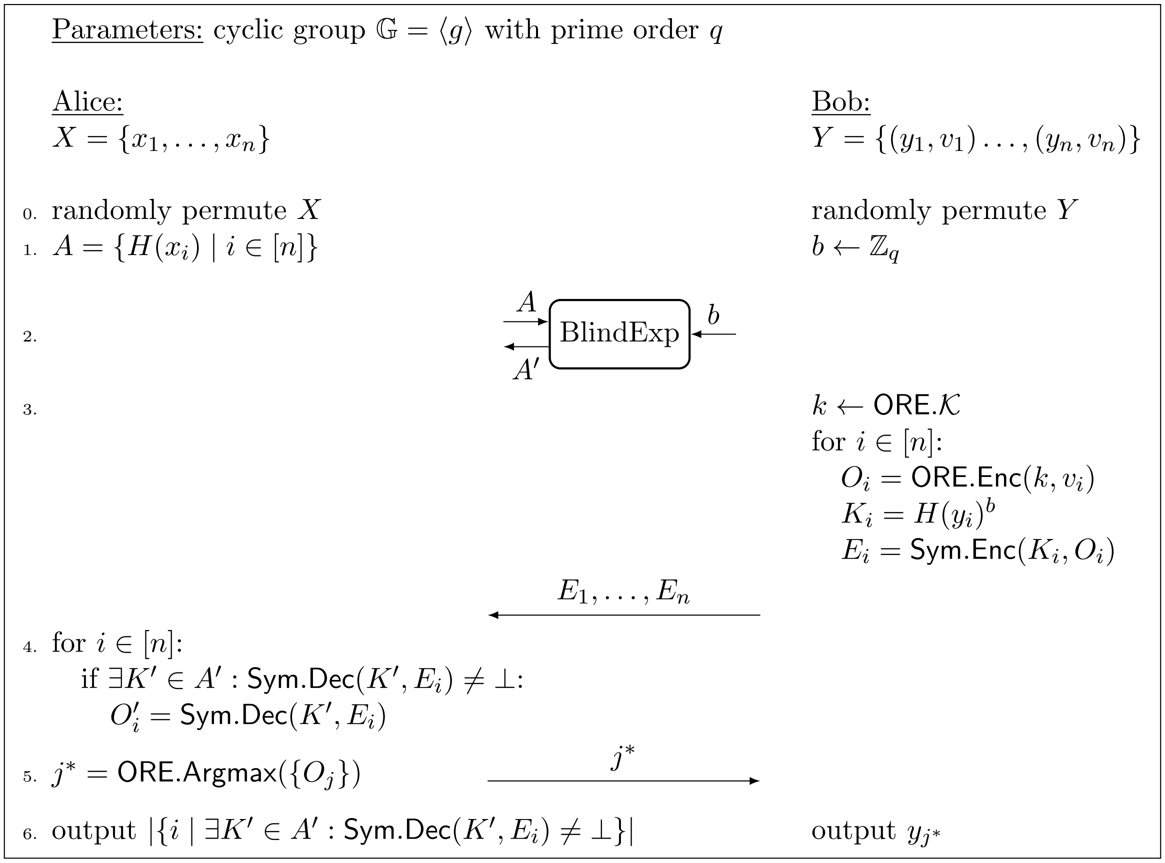
Lemma 10. *The protocol in Figure 5 securely realizes Figure 4 against a semi-honest Bob.*

Proof. Simulation for Bob is trivial. Bob's view consists only of his view from BlindExp (which is indistinguishable from random), and the final protocol message j^*, which can be easily computed from his ideal output. ☐

Lemma 11. *The protocol in Figure 5 securely realizes Figure 4 against a semi-honest Alice (assuming the DDH assumption).*

Proof. Alice's view consists of received protocol messages $A', E_1, \ldots, E_n$, and her view of the random oracle. These values are computed as in Hybrid 0 in Figure 6. Here $\mathcal{A}$ denotes the adversary that receives Alice's view along with oracle access to the random oracle H. For convenience in Hybrid 0 we have named values $H(z)^b$ as K_z^* — if both Alice and Bob have a common element z then they will both refer to the same K_z^*.

In Hybrid 3 we present a simulator for Alice's view. Although this hybrid is written to take both parties' sets as input, it uses these inputs only to calculate the size m of the intersection. It



```
Parameters: cyclic group $\mathbb{G} = \langle g \rangle$ with prime order $q$

Alice:                                                Bob:
$X = \{x_1, \ldots, x_n\}$                            $Y = \{(y_1, v_1) \ldots, (y_n, v_n)\}$

0. randomly permute $X$                               randomly permute $Y$
1. $A = \{H(x_i) \mid i \in [n]\}$                    $b \leftarrow \mathbb{Z}_q$
2.            $A \rightarrow$ BlindExp $\leftarrow b$;  $A' \leftarrow$ BlindExp
3.                                                    $k \leftarrow \mathsf{ORE}.\mathcal{K}$
                                                      for $i \in [n]$:
                                                          $O_i = \mathsf{ORE.Enc}(k, v_i)$
                                                          $K_i = H(y_i)^b$
                                                          $E_i = \mathsf{Sym.Enc}(K_i, O_i)$
                        $\xleftarrow{E_1, \ldots, E_n}$
4. for $i \in [n]$:
       if $\exists K' \in A' : \mathsf{Sym.Dec}(K', E_i) \neq \bot$:
           $O'_i = \mathsf{Sym.Dec}(K', E_i)$
5. $j^* = \mathsf{ORE.Argmax}(\{O_j\})$    $\xrightarrow{j^*}$
6. output $|\{i \mid \exists K' \in A' : \mathsf{Sym.Dec}(K', E_i) \neq \bot\}|$      output $y_{j^*}$
```

Figure 5: Protocol for identifying the best item according to a unilateral rank.

then uses m to compute the remainder of the view. The hybrid also uses permutations μ, π — μ is used to index into elements of A', and π is used to randomly choose which m values are simulated as part of the intersection. Note that A' is given to Alice only as an unordered set — *i.e.*, indices of these items are not meaningful.

It suffices show that adjacent hybrids in Figure 6 are indistinguishable.

Hybrids 0 & 1: The only difference is that K^*_z values are chosen uniformly. The hybrids are indistinguishable via a reduction to the DDH problem. Specifically, consider a reduction algorithm that receives $(\alpha_1, \ldots, \alpha_m, B, \beta_1, \ldots, \beta_m)$. For each $z_i \in \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$, the reduction algorithm programs $H(z_i) = \alpha_i$ and sets $K^*_{z_i} = \beta_i$. Otherwise, the reduction algorithm runs the code of Hybrid 1. If the input is from the DH distribution — *i.e.*, if $B = g^b$ and $\beta_i = \alpha_i^b$ — then the output of the reduction algorithm is exactly that of Hybrid 0. If the input is from the random distribution, then the reduction algorithm is exactly as Hybrid 1.

Hybrids 1 & 2: Consider a value y_i that is distinct from all $\{x_j\}$ values — *i.e.*, an item not in common to the two parties. Then the only place $K^*_{y_i}$ is used in Hybrid 1 is as the value K_i, when the ciphertext $E_i = \mathsf{Enc}(K_i, S_i)$ is generated. Hence, a straight-forward reduction to the one-time security of Enc (Section 2.3) shows that E_i is indistinguishable from an encryption of a dummy value 0. Performing such a reduction for each such y_i yields Hybrid 2.

Hybrids 2 & 3: Instead of sampling all K^*_i values upfront, they are sampled later, as needed. In the second for-loop of Hybrid 2, m of the ciphertexts (m = the cardinality of the intersection) are encrypted with keys appearing in A'. Furthermore, since the y_i values are randomly shuffled (and the ordering of A' is not meaningful), the choice of m common keys is random. The same is true of Hybrid 3, which uses the random permutations μ and π to select which m keys are common.

The only other difference is that the O_i values in Hybrid 2 are encryptions of v_i plaintexts, whereas in Hybrid 3 they are encryptions of $\{1, \ldots, m\}$ plaintexts. By a straightforward reduction to the ORE security property (Section 2.4), the two hybrids are indistinguishable. ☐

5 Finding the Best Item According to a Combined Score

In this section we consider the following variant. Alice holds a set of (x_i, u_i) pairs, and Bob holds a set of (y_i, v_i) pairs. If Alice and Bob hold a common item, say $z = x_i = y_j$, then define that item's *score* as $u_i + v_j$. In other words, an item's score is the sum of its scores from both parties. We consider the problem of identifying the common item with highest score.

In this variant, Alice will learn (1) how many common items they have (*i.e.*, the cardinality of the intersection), and (2) the set of *combined* scores for all common items. Alice does not learn the individual contributions of the parties (*i.e.*, the u_i and v_j value that are added to give an item's score), nor does she learn which scores correspond to which items, or which items are in the intersection. Bob learns only the identity of the item with highest combined score.

In Figure 7 we formally define the ideal functionality for this variant of sampling from the intersection. Alice receives a vector $(w_1, \ldots, w_n)$, such that if k items are common to the parties, then all but k entries in the vector will be $\perp$. The remaining k entries will contain the combined scores of the common items. The vector w is uniformly permuted, and so Alice learns only the cardinality of the intersection and the (unordered) set of combined scores for items in the intersection.

In the case that there are no common items, all w_i values will be $\perp$. We use the following notational conventions for that case: if every $w_i = \perp$ then $\arg\max_i w_i = \perp$; if the value of $j^* = \perp$ then $y_{j^*} = \perp$.

In our protocol Alice will learn a value of the form D^{w_i} and will need to compute $\mathrm{dlog}_D(D^{w_i}) = w_i$. Our protocol therefore supports inputs where **the scores (w_i values) have polynomial magnitude.**

5.1 2-Blind Exponentiation

Our protocol requires a variant of the blind exponentiation subprotocol from Section 3.1. See Figure 8.

In this variant, Alice has a set of pairs. For each such pair (ℓ, r) Alice wants to learn (ℓ^b, r^d) where b, d are exponents chosen by Bob. The two components of each pair are kept together, but the set of pairs is randomly shuffled. Alice learns only the *unordered set* of (ℓ^b, r^d) values.

Lemma 12. *Alice's output is $\{(\ell^b, r^d) \mid (\ell, r) \in M\}$. Furthermore, if Alice is semi-honest and Bob's inputs b, d are uniform, then Alice's view in Figure 8 can be simulated given only this output.*

Lemma 13. *If Bob is semi-honest, and Alice's inputs have the form $(\ell_i, r_i) = (H(x_i), t_i H(x_i))$ for distinct x_i values (x_i and t_i values chosen by the adversary), then Bob's view in Figure 8 is indistinguishable from random (assuming the DDH assumption).*

Hybrid 0 (Real)

```
inputs $\{x_i \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

shuffle $\{(y_i, v_i)\}$
$H \leftarrow$ random oracle
$b \leftarrow \mathbb{Z}_q$
$k \leftarrow \mathsf{ORE}.\mathcal{K}$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $K_z^* = H(z)^b$

$A' = \{K_{x_i}^*\}_{i \in [n]}$

for $i \in [n]$:
    $K_i = K_{y_i}^* b$
    $O_i = \mathsf{ORE.Enc}(k, v_i)$
    $E_i = \mathsf{Enc}(K_i, O_i)$

ret $\mathcal{A}^H(A', E_1, \ldots, E_n)$
```

Hybrid 1

```
inputs $\{x_i \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

shuffle $\{(y_i, v_i)\}$
$H \leftarrow$ random oracle
$k \leftarrow \mathsf{ORE}.\mathcal{K}$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $\boxed{K_z^* \leftarrow \mathbb{G}}$

$A' = \{K_{x_i}^*\}_{i \in [n]}$

for $i \in [n]$:
    $K_i = K_{y_i}^*$
    $O_i = \mathsf{ORE.Enc}(k, v_i)$
    $E_i = \mathsf{Enc}(K_i, O_i)$

ret $\mathcal{A}^H(A', E_1, \ldots, E_n)$
```

Hybrid 2

```
inputs $\{x_i \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

shuffle $\{(y_i, v_i)\}$
$H \leftarrow$ random oracle
$k \leftarrow \mathsf{ORE}.\mathcal{K}$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $K_z^* \leftarrow \mathbb{G}$

$A' = \{K_{x_i}^*\}_{i \in [n]}$

for $i \in [n]$:
    ┌─────────────────────────────────
    │ if $\exists j : y_i = x_j$:
    │     $K_i = K_{y_i}^*$
    │     $O_i = \mathsf{ORE.Enc}(k, v_i)$
    │     $E_i = \mathsf{Enc}(K_i, O_i)$
    │ else:
    │     $E_i = \mathsf{Enc}(K_i, 0)$
    └─────────────────────────────────

ret $\mathcal{A}^H(A', E_1, \ldots, E_n)$
```

Hybrid 3 (Simulation):

```
inputs $\{x_i \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

┌─────────────────────────────────
│ $\mu, \pi \leftarrow$ random
│     permutations on $[n]$
│
│ $m = |\{x_i\}_i \cap \{y_i\}|$
└─────────────────────────────────
$H \leftarrow$ random oracle
$k \leftarrow \mathsf{ORE}.\mathcal{K}$

for $i \in [n]$:
    $K_i' \leftarrow \mathbb{G}$
$A' = \{K_i'\}_{i \in [n]}$

for $i \in [n]$:
    if $\pi(i) \leq m$:
        $O_i = \mathsf{ORE.Enc}(k, \pi(i))$
        $E_i = \mathsf{Enc}(K_{\mu(i)}', O_i)$
    else:
        $\tilde{K} \leftarrow \{0,1\}^\lambda$
        $E_i = \mathsf{Enc}(\tilde{K}, 0)$

ret $\mathcal{A}^H(A', E_1, \ldots, E_n)$
```

Figure 6: Hybrids in the security proof for the protocol in Figure 5

```
receive input $X$ from Alice and $Y$ from Bob.
assign a random ordering to $Y$ as $\{(y_1, v_1), \dots, (y_n, v_n)\}$
for $i \in [n]$:
    if $\exists (x, u) \in X$ with $x = y_i$:  $w_i := u + v_i$
    else:                                      $w_i := \bot$
give $w_1, \dots, w_n$ to Alice
set $j^* := \arg\max_i w_i$, and give $y_{j^*}$ to Bob (see text for conventions)
```

Figure 7: Ideal functionality for sampling the best item from the intersection, according to a combined score.

```
Parameters: cyclic group $\mathbb{G} = \langle g \rangle$ with prime order $q$

Alice:                                                    Bob:
$M = \{(\ell_1, r_1), \dots, (\ell_n, r_n)\}$             $b, d \in \mathbb{Z}_q$

1. $a, c \leftarrow \mathbb{Z}_q$
2. $M' = \{(\ell^a, r^c) \mid (\ell, r) \in M\}$
                         $\xrightarrow{M'}$
3.                                                        $M'' = \{((\ell')^b, (r')^d) \mid (\ell', r') \in M'\}$
                         $\xleftarrow{M''}$
4. output $\{((\ell'')^{1/a}, (r'')^{1/c}) \mid (\ell'', r'') \in M''\}$
```

Figure 8: 2-Blind-Exp subprotocol.

5.2 Intersection Protocol

We first present the high-level intuition behind the protocol. The challenge is to allow Alice to learn the combined score $u_i + v_j$ for a common item $x_i = y_j$, without revealing the individual u_i and v_j terms.

The main idea is to blind Alice's value u_i with some random mask, and blind Bob's value v_j with a corresponding mask, so that the two masks can cancel out revealing only $u_i + v_j$. The main question is: *what random value shall serve as the mask?* Alice and Bob must apply the same mask if they have a common item ($x_i = y_j$), so the mask must be derived from the identity of the item. Our approach is as follows.

- For each (x_i, u_i) in Alice's set, she computes $g^{u_i} \cdot H(x_i)$.
- Using a blind exponentiation protocol, Alice obtains $[g^{u_i} \cdot H(x_i)]^d$ where d is a random exponent chosen by Bob. Here the value $H(x_i)^d$ is pseudorandom from Alice's view, so it serves as a blinding mask to the score g^{u_i}.
- For each item (y_j, v_j) in Bob's set, he can compute $[g^{v_j} \cdot H(y_j)^{-1}]^d$. He can encrypt these values (similar to the previous protocol), so that Alice learns them only if she has the matching item in her set.

Given her blinded value and the blinded value obtained from Bob, she can compute:

$$[g^{u_i} \cdot H(x_i)]^d \cdot [g^{v_j} \cdot H(y_j)^{-1}]^d = (g^d)^{u_i + v_j}$$

If Bob also sends her g^d then she can compute the discrete log with respect to base g^d to obtain $u_i + v_j$. As mentioned above, computing the discrete log requires the magnitude of these scores to be polynomial in size.

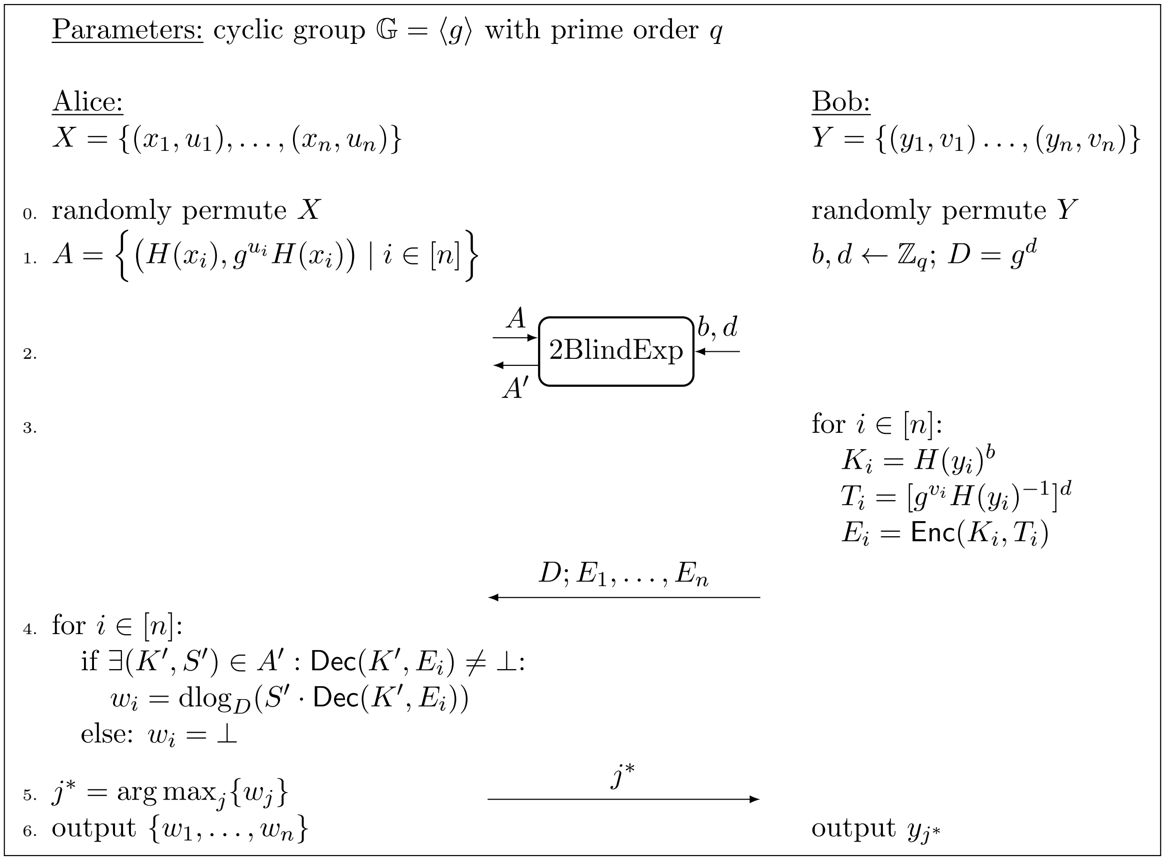


Figure 9: Protocol for identifying the best item according to a combined score.

Lemma 14. *The protocol in Figure 9 is correct.*

Proof. If $z = x_j = y_i$ for some i, j then A' will contain a tuple $\Big(H(z)^b, [g^{u_j} H(z)]^d\Big)$ and we will also have $E_i = \mathsf{Enc}\Big(H(z)^b, [g^{v_i} H(z)^{-1}]^d\Big)$. As such, Alice will eventually decrypt this E_i and compute

$$w_i = \operatorname*{dlog}_D\Big([g^{u_j} H(z)]^d \cdot [g^{v_i} H(z)^{-1}]^d\Big) = \operatorname*{dlog}_D\Big(D^{u_j + v_i}\Big) = u_j + v_i$$

If $y_i \notin \{x_1, \ldots, x_n\}$ then Alice will decrypt E_i with only independently generated keys, which will fail with overwhelming probability (cf. robust decryption Section 2.3). Hence, she sets $w_i = \bot$.

Overall, Alice's vector w contains exactly the combined scores of all items in the intersection. From this, the correctness of the last protocol message follows easily. □

Lemma 15. *The protocol in Figure 9 securely realizes Figure 7 against a semi-honest Bob.*

Proof. Simulation for Bob is trivial. Bob's view consists only of his view from 2BlindExp (which is indistinguishable from random), and the final protocol message j^*, which can be easily computed from his ideal output. □

Lemma 16. *The protocol in Figure 9 securely realizes Figure 7 against a semi-honest Alice.*

Proof. Alice's view consists of received protocol messages $A', D, E_1, \dots, E_n$, and her view of the random oracle. These values are computed as in Hybrid 0 in Figure 10. Here $\mathcal{A}$ denotes the adversary that receives Alice's view along with oracle access to the random oracle H. For convenience in Hybrid 0 we have given values $H(z)^b$ and $H(z)^d$ names K^*_z and T^*_z, respectively — if both Alice and Bob have a common element z then they will both refer to the same K^*_z and T^*_z.

In Hybrid 3 we present a simulator for Alice's view. Although this hybrid is written to take both parties' sets as input, it uses these inputs only to first compute a vector w that is Alice's output from the ideal functionality. It then uses w to compute the remainder of the view. The hybrid also uses a partial permutation μ that is used to index into the elements of the set A'. This is for notational simplicity, as A' is given to Alice only as an unordered set — *i.e.*, indices of these items are not meaningful.

It suffices show that adjacent hybrids in Figure 10 are indistinguishable.

Hybrids 0 & 1: The only difference is that K^*_z and T^*_z are chosen uniformly. The hybrids are indistinguishable via two separate reductions to the DDH problem. Specifically, consider a reduction algorithm that receives $(\alpha_1, \dots, \alpha_m, B, \beta_1, \dots, \beta_m)$. For each $z_i \in \{x_1, \dots, x_n, y_1, \dots, y_n\}$, the reduction algorithm programs $H(z_i) = \alpha_i$ and sets $K^*_{z_i} = \beta_i$. Otherwise, the reduction algorithm runs the code of Hybrid 1. If the input is from the DH distribution — *i.e.*, if $B = g^b$ and $\beta_i = \alpha_i^b$ — then the output of the reduction algorithm is exactly that of Hybrid 0. If the input is from the random distribution, then the reduction algorithm is like that of Hybrid 0 except that K^*_i values are chosen as in Hybrid 1.

With another reduction to the DDH assumption (setting $D = B$ and $T^*_{z_i} = \beta_i$), the output of the reduction algorithm becomes exactly that of Hybrid 1.

Hybrids 1 & 2: Consider a value y_i that is distinct from all $\{x_j\}$ values — *i.e.*, an item not in common to the two parties. Then the only place $K^*_{y_i}$ is used in Hybrid 1 is as the value K_i, when the ciphertext $E_i = \mathsf{Enc}(K_i, S_i)$ is generated. Hence, a straight-forward reduction to the one-time security of Enc (Section 2.3) shows that E_i is indistinguishable from an encryption of a dummy value 0. Performing such a reduction for each such y_i yields Hybrid 2.

Hybrids 2 & 3: Instead of sampling all K^*_i and T^*_i values upfront, they are sampled later, as needed. If $z = y_i = x_j$ for some i, j, then Hybrid 2 would first sample $S'_j = D^{u_j} T^*_z$ and then $S_i = D^{v_i}(T^*_z)^{-1}$. Since these are the only two places where T^*_z is used, and T^*_z is uniform, this is equivalent to Hybrid 3's behavior of setting $S'_j \leftarrow \mathbb{G}$ and then $S_i = D^{u_j+v_i}(S'_j)^{-1}$. If $z = y_i \notin \{x_1, \dots, x_n\}$ then Hybrid 2 uses K^*_z *only* as encryption to a single ciphertext. ☐

Acknowledgements

The first two authors are supported by NSF award DMS-1757995.

References

[BLR+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Elisabeth Oswald and Marc Fischlin, editors, *EU-*

Hybrid 0 (Real)

```
inputs $\{(x_i, u_i) \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

$H \leftarrow$ random oracle
$b, d \leftarrow \mathbb{Z}_q$; $D = g^d$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $K^*_z = H(z)^b$
    $T^*_z = H(z)^d$

for $i \in [n]$:
    $K'_i = K^*_{x_i}$
    $S'_i = D^{u_i} T^*_{x_i}$
$A' = \{(K'_i, S'_i)\}_{i \in [n]}$

for $i \in [n]$:
    $K_i = K^*_{y_i}$
    $S_i = D^{v_i} (T^*_{y_i})^{-1}$
    $E_i = \mathsf{Enc}(K_i, S_i)$

ret $\mathcal{A}^H(A', D, E_1, \ldots, E_n)$
```

Hybrid 1

```
inputs $\{(x_i, u_i) \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

$H \leftarrow$ random oracle
$\boxed{D \leftarrow \mathbb{G}}$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $\boxed{K^*_z \leftarrow \mathbb{G}}$
    $\boxed{T^*_z \leftarrow \mathbb{G}}$

for $i \in [n]$:
    $K'_i = K^*_{x_i}$
    $S'_i = D^{u_i} T^*_{x_i}$
$A' = \{(K'_i, S'_i)\}_{i \in [n]}$

for $i \in [n]$:
    $K_i = K^*_{y_i}$
    $S_i = D^{v_i} (T^*_{y_i})^{-1}$
    $E_i = \mathsf{Enc}(K_i, S_i)$

ret $\mathcal{A}^H(A', D, E_1, \ldots, E_n)$
```

Hybrid 2

```
inputs $\{(x_i, u_i) \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

$H \leftarrow$ random oracle
$D \leftarrow \mathbb{G}$

for $z \in \{x_1, \ldots, x_n\}$
        $\cup \{y_1, \ldots, y_n\}$:
    $K^*_z \leftarrow \mathbb{G}$
    $T^*_z \leftarrow \mathbb{G}$

for $i \in [n]$:
    $K'_i = K^*_{x_i}$
    $S'_i = D^{u_i} T^*_{x_i}$
$A' = \{(K'_i, S'_i)\}_{i \in [n]}$

for $i \in [n]$:
    $K_i = K^*_{y_i}$
    $S_i = D^{v_i} (T^*_{y_i})^{-1}$
    $\boxed{\text{if } \exists j : y_i = x_j:}$
        $\boxed{E_i = \mathsf{Enc}(K_i, S_i)}$
    $\boxed{\text{else: } E_i = \mathsf{Enc}(K_i, 0)}$

ret $\mathcal{A}^H(A', D, E_1, \ldots, E_n)$
```

Hybrid 3 (Simulator)

```
inputs $\{(x_i, u_i) \mid i \in [n]\}$
    and $\{(y_i, v_i) \mid i \in [n]\}$

for $i \in [n]$:
    if $\exists j : x_j = y_i$:
        $\mu(i) := j$
        $w_i := u_j + v_i$
    else $w_i := \bot$

        $\hookrightarrow$

$H \leftarrow$ random oracle
$D \leftarrow \mathbb{G}$

for $i \in [n]$:
    $K'_i \leftarrow \mathbb{G}$
    $S'_i \leftarrow \mathbb{G}$
$A' = \{(K'_i, S'_i)\}_{i \in [n]}$

        $\hookrightarrow$

for $i \in [n]$:
    if $w_i \neq \bot$:
        $K_i = K'_{\mu(i)}$
        $S_i = D^{w_i} (S'_{\mu(i)})^{-1}$
        $E_i = \mathsf{Enc}(K_i, S_i)$
    else:
        $K_i \leftarrow \mathbb{G}$
        $E_i = \mathsf{Enc}(K_i, 0)$

ret $\mathcal{A}^H(A', D, E_1, \ldots, E_n)$
```

Figure 10: Hybrids in the security proof for the protocol in Figure 9

ROCRYPT 2015, Part II, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, April 2015.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, Heidelberg, 1998. Invited paper.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020.

[DKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, Heidelberg, December 2010.

[GMR+21] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 591–617. Springer, Heidelberg, May 2021.

[Haz15] Carmit Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 90–120. Springer, Heidelberg, March 2015.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

[HFH99] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *ACM CONFERENCE ON ELECTRONIC COMMERCE*. ACM, 1999.

[IKN+19] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. Cryptology ePrint Archive, Report 2019/723, 2019. `https://eprint.iacr.org/2019/723`.

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 418–435. Springer, Heidelberg, September 2010.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005.

[LW16] Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1167–1178. ACM Press, October 2016.

[Mea86] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134, April 1986.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015.

[PSTY19] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Heidelberg, May 2019.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 125–157. Springer, Heidelberg, April / May 2018.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014.

[RR17] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

[RR22] Peter Rindal and Srinivasan Raghuraman. Blazing fast psi from improved okvs and subfield vole. Cryptology ePrint Archive, Report 2022/320, 2022. https://ia.cr/2022/320.

[RT21] Mike Rosulek and Ni Trieu. Compact and malicious private set intersection for small sets. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1166–1181. ACM Press, November 2021.