Private Set Intersection

Hsi-Kang, Hsu
Yi-Xuan, Lee

1 Introduction

Secure multi-party computation (MPC) allows a set of parties to interact via protocols and jointly compute a function on their private inputs, without revealing anything but the output of the function. [Yao86] has presented a two-party garbled-circuit based protocol for general functions, and the multi-party protocol was given by [GMW87]. To make MPC useful in practice, researches has been focused on the improvement of efficiency. e.g. [Ash+12], [Cia+20]. For one's interest, we recommend the surveys [Lin20], [FY22], which give important results of MPC researches.
Although any functions can be securely computed, there are some specific functions that have more efficient solutions. Private set intersection (PSI) is one of them. In a PSI protocol, two parties want to compute the intersection of their input sets, while revealing nothing but the items in the intersection. PSI has many real life applications, for example, when a company wants to know the effectiveness of online advertising:
The company, which knows who made a purchase and what they spent, wants to know how many users both saw an ad and made a corresponding purchase. However, only the ad supplier knows the users who have seen a particular ad. Both parties are unwilling to expose the underlying data, and this is exactly an instance of a PSI problem. [Ion+20b] and their following work [Ion+20a], within Google, discuss a more realistic situation, and consider a particular variant of PSI.
In this project, we are going to introduce two of the most practical approaches for PSI:

- DH-based construction
- OT based construction

In general, OT-based protocols are (significantly) faster, but DH-based protocols requires less communication.

2 Preliminaries

2.1 Security Model

We follow the Real-Ideal paradigm as introduced in [EKR18]. [Gol04] and [Lin17] also give comprehensive overview of MPC security frameworks. Here we only use the definition of security against semi-honest or malicious adversaries.

2.2 PSI Functionality

Consider this PSI functionality $\mathcal{F}(X, Y) = (X \cap Y, \perp)$, i.e. party with input X learns the output $f_1(X, Y) = X \cap Y$, and party with input Y learns $f_2(X, Y) = \perp$ (learns nothing).

3 DH-based approach

In this section, we will give a DH-baed PSI protocol proposed by [RT21]. The main idea is to embed protocol messages in a polynomial, and make use of the ideal permutation oracle and random oracles to achieve malicious security. For small sets (500 or fewer items), their PSI protocol is the least time and communication among all known PSI protocols, including the semi-honest and malicious ones.
Notice that before their work, there is a constant ratio of performance gap between malicious and semi-honest protocols in the DH-based approach, thus their construction has a further contribution for closing this gap. This points out the importance of their protocol being malicious secure.

3.1 Key Agreement Protocol (KA)

We start with the definition of a 2-round KA protocol that consists of algorithms $\mathsf{KA.msg}_1$, $\mathsf{KA.msg}_2$, $\mathsf{KA.key}_1$, $\mathsf{KA.key}_2$ as shown in Table 1 with the following parameters:

- $\mathsf{KA}.\mathcal{R}$: the space of random bits for the two parties.
- $\mathsf{KA}.\mathcal{M}$: the space of possible messages for Party 2.
- $\mathsf{KA}.\mathcal{K}$: the space of possible output keys.

P_1		P_2
$a \leftarrow \mathsf{KA}.\mathcal{R}$		
$m_1 = \mathsf{KA.msg}_1(a)$	$\xrightarrow{m_1}$	
		$b \leftarrow \mathsf{KA}.\mathcal{R}$
	$\xleftarrow{m_2}$	$m_2 = \mathsf{KA.msg}_2(b, m_1)$
output $\mathsf{KA.key}_1(a, m_2)$		output $\mathsf{KA.key}_2(b, m_1)$

Table 1: 2-round KA Protocol

If the second message m_2 does not depend on m_1, i.e. $m_2 = \mathsf{KA.msg}_2(b)$, then m_1, m_2 can be sent simultaneously. In this case, we say that the KA protocol is **one-round**. For a concrete example, one can consider the 1-round Hashed-DHKA protocol as shown in table 2:
Given DDH instance $(\mathbb{G}, q, g)$ and random oracle $H : \mathbb{G} \to \{0,1\}^l$

- $\mathsf{KA}.\mathcal{R} = \mathbb{Z}_q$, $\mathsf{KA}.\mathcal{M} = \mathbb{G}$, $\mathsf{KA}.\mathcal{K} = \{0,1\}^l$
- $\mathsf{KA.msg}_1(a) = g^a$, $\mathsf{KA.msg}_2(b) = g^b$
- $\mathsf{KA.key}_1(a, g^b) = H((g^b)^a)$, $\mathsf{KA.key}_2(b, g^a) = H((g^a)^b)$

P_1		P_2
$a \leftarrow \mathsf{KA}.\mathcal{R}$		$b \leftarrow \mathsf{KA}.\mathcal{R}$
$m_1 = g^a$	$\xrightarrow{m_1}$	
	$\xleftarrow{m_2}$	$m_2 = g^b$
output $H((g^b)^a)$		output $H((g^a)^b)$

Table 2: 1-round Hashed-DHKA Protocol

3.2 Security Properties

Definition 3.1 (Correctness)**.** A KA scheme is **correct** if, when executed honestly as specified in table 1, the two parties give identical output. In other words, for all $a, b \in \mathsf{KA}.\mathcal{R}$:

$$\mathsf{KA.key}_1(a, \mathsf{KA.msg}_2(b, \mathsf{KA.msg}_1(a))) = \mathsf{KA.key}_2(b, \mathsf{KA.msg}_1(a))$$

- For Hashed-DHKA, the correctness is obvious.

Definition 3.2 (Secure against an eavesdropper)**.** A KA scheme is **secure against an eavesdropper** if the following distributions are indistinguishable:

$a, b \leftarrow \mathsf{KA}.\mathcal{R}$	$a, b \leftarrow \mathsf{KA}.\mathcal{R}$
$m_1 = \mathsf{KA.msg}_1(a)$	$m_1 = \mathsf{KA.msg}_1(a)$
$m_2 = \mathsf{KA.msg}_2(b, m_1)$	$m_2 = \mathsf{KA.msg}_2(b, m_1)$
$k = \mathsf{KA.key}_2(b, m_1)$	$k \leftarrow \mathsf{KA}.\mathcal{K}$
return (m_1, m_2, k)	return (m_1, m_2, k)

- For Hashed-DHKA, security against an eavesdropper comes from the Hashed Diffie-Hellman (HDH) assumption [ABR01], which states that:
$(g^a, g^b, H(g^{ab})) \approx_c (g^a, g^b, \mathcal{U})$ for $a, b \leftarrow \mathbb{Z}_q$ and $\mathcal{U}$: uniformly distribution over $\{0,1\}^l$.

Definition 3.3 (Non-malleable)**.** A KA scheme is **non-malleable** if for every PPT distinguisher $\mathcal{D}$ that has oracle access to $\mathrm{KA}.key_1(a, \cdot)$, provided $\mathcal{D}$ never queries the oracle on m_2, the following distributions are indistinguishable:

$a, b \leftarrow \mathsf{KA}.\mathcal{R}$	$a, b \leftarrow \mathsf{KA}.\mathcal{R}$
$m_1 = \mathsf{KA.msg}_1(a)$	$m_1 = \mathsf{KA.msg}_1(a)$
$m_2 = \mathsf{KA.msg}_2(b, m_1)$	$m_2 = \mathsf{KA.msg}_2(b, m_1)$
$k = \mathsf{KA.key}_1(a, m_2)$	$k \leftarrow \mathsf{KA}.\mathcal{K}$
return $\mathcal{D}^{\mathsf{KA.key}_1(a,\cdot)}(m_1, m_2, k)$	return $\mathcal{D}^{\mathsf{KA.key}_1(a,\cdot)}(m_1, m_2, k)$

- For Hashed-DHKA, non-malleability comes from the oracle Diffie-Hellman (ODH) assumption [ABR01], which states that:
$\mathcal{D}^{H_a(\cdot)}(g^a, g^b, H(g^{ab})) \approx_c \mathcal{D}^{H_a(\cdot)}(g^a, g^b, \mathcal{U})$ for $a, b \leftarrow \mathbb{Z}_q$, $\mathcal{U}$: uniformly distribution over $\{0,1\}^l$, and $H_a(x) = H(x^a)$ (correspond to the $\mathsf{KA.key}_1(a, \cdot)$ oracle).

$(view, \tilde{m}_1) \leftarrow \mathcal{A}$	$(view, \tilde{m}_1) \leftarrow \mathcal{A}$
$b \leftarrow \mathsf{KA}.\mathcal{R}$	
$m_2 = \mathsf{KA.msg}_2(b, \tilde{m}_1)$	$m_2 \leftarrow \mathsf{KA}.\mathcal{M}$
return $(view, m_2)$	return $(view, m_2)$

Definition 3.4 (Pseudorandom second message)**.** A KA scheme has **pseudorandom second messages** if m_2 is indistinguishable from random, even to someone who chooses m_1 adversarily. Formally, the following distributions are indistinguishable for all PPT $\mathcal{A}$:

- For Hashed-DHKA, m_2 does not depend on m_1, and $\mathsf{KA.msg}_2(b) = g^b$, which is truly random.

3.3 Ideal Permutation

In the ideal permutation model, the parties have access to a random permutation Π and its inverse Π^{-1}, write $\Pi^{\pm}$ to refer to the pair (Π, Π^{-1}).

Now we are able to construct the PSI protocol based on this KA protocol.

3.4 Malicious PSI from KA

The protocol presented in table 3 requires the following building blocks:

- A 2-round KA protocol with $\mathsf{KA}.\mathcal{M} = \mathbb{F}$ for some finite field $\mathbb{F}$.
- The KA protocol is non-malleable (Definition 3) and has pseudorandom messages in $\mathbb{F}$ (Definition 4)
- Under ideal permutation model: Parties have oracle access to $\Pi^{\pm}$, where $\Pi{:}\mathbb{F} \to \mathbb{F}$
- Under RO model: Parties have access to random oracles $H_1 : \{0,1\}^* \to \mathbb{F}, H_2 : \{0,1\}^* \times \mathbb{F} \to \{0,1\}^{2\kappa}$.

In this work the authors use the Hashed-DHKA with elligator encodings [Ber+13], whose protocol satisfies the above requirements and provides an efficient experimental result.
From the protocol construction, we could see the advantage of using KA as the underlying protocol: The only KA message m from $\mathcal{P}_1$ can be reused many times for $\mathcal{P}_2$ to generate n $\mathsf{KA}.msg_2$ and $\mathsf{KA.key}_2$, which significantly reduce the communication cost.
We may first take a look at the correctness of this protocol. If $y_j = x_i$ for some i, j, then

$$\begin{aligned} k_j &= \mathsf{KA.key}_1(a, \Pi(P(H_1(y_j)))) = \mathsf{KA.key}_1(a, \Pi(P(H_1(x_i)))) = \mathsf{KA.key}_1(a, \Pi(f_i)) \\ &= \mathsf{KA.key}_1(a, m_i') = \mathsf{KA.key}_1(a, \mathsf{KA.msg}_2(b_i, m)) = \mathsf{KA.key}_2(b_i, m) \end{aligned}$$

the last equality follows from the correctness of the underlying KA protocol. Therefore,

$$k_j' = H_2(y_j, k_j) = H_2(x_i, \mathsf{KA.key}_2(b_i, m))$$

$\mathcal{P}_1$ (Sender)		$\mathcal{P}_2$ (Receiver)
$Y = \{y_1, \cdots, y_n\} \subset \{0,1\}^*$		$X = \{x_1, \cdots, x_n\} \subset \{0,1\}^*$
$a \leftarrow \mathrm{KA}.\mathcal{R}$		
$m = \mathrm{KA}.msg_1(a)$	$\xrightarrow{m}$	
		for $i \in [n]$:
		$b_i \leftarrow \mathrm{KA}.\mathcal{R}$
		$m'_i = \mathrm{KA}.msg_2(b_i, m)$
		$f_i = \Pi^{-1}(m'_i)$
	$\xleftarrow{P}$	$P = \mathrm{interpol}_{\mathbb{F}}(\{(H_1(x_i)), f_i \mid x_i \in X\})$
(Abort if $\deg(P) < 1$)		
for $i \in [n]$:		
$k_i = \mathrm{KA}.key_1(a, \Pi(P(H_1(y_i))))$		
$k'_i = H_2(y_i, k_i)$		
$K = \{k'_1, \cdots, k'_n\}$	$\xrightarrow{K}$	
		output $\{x_i \mid H_2(x_i, \mathrm{KA}.key_2(b_i, m)) \in K\}$

Table 3: Malicious PSI protocol

which will be in the output of $\mathcal{P}_2$.
For $y_j \notin X$, since H_2 is a random oracle, the probability that $H_2(y_j, k_j)$ having collision with some $H_2(x_i, \mathsf{KA.key}_2(b_i, m))$ is negligible.
The following two lemma state the malicious security of this PSI protocol.

Lemma 3.1. The protocol is **secure against a malicious sender**, if KA has pseudorandom messages.

Proof. (sketch) The goals of the simulator are

- To provide a view of P.
  First, by PRP/PRF Switching Lemma [CN08], with indistinguishable change, one can simulate $\Pi^{\pm}$ by having it ack like a random oracle. Now, observe that if $f_i = \Pi^{-1}(m'_i)$ are all "fresh" queries (both $\Pi(f_i)$ and $\Pi^{-1}(m'_i)$ have not been queried), then the polynomial is truly random, i.e. $P \leftarrow \mathbb{F}[x]$. Abort those non-fresh queries also cause a indistinguishable change.

- To extract a set $\hat{Y}$ that explains the effect of K.
  This can be translated to the following question: What is the set $\hat{Y}$ s.t. when honestly playing the protocol with this set, the sender will send K to the receiver?
  Intuitively, the element $k'_i \in K$ is supposed to have the form $H_2(y_i, k_i)$, i.e. (y_i, k_i) must have been a query to the random oracle H_2. Moreover, k_i is supposed to have the form $\mathsf{KA.key}_1(a, \Pi(P(H_1(y_i))))$, i.e. $P(H_1(y_i))$ must have been a query to the permutation oracle Π.
  Thus, the simulator can observes all queries to H_2 and Π, record those to H_2 in a set $\mathcal{O}_2$, and provide the output of Π by a KA message for which it knows the random bit. Those $y'_i s$ that correspond to the queries as expected are what we want.

The complete behavior of simulator is given in table 4

Sim (Sender) with input $Y = \{y_1, \cdots, y_n\} \subset \{0,1\}^*$	
	$\boxed{H_2}$ record $\mathcal{O}_2 = \{(y, k, H_2(y,k))\}$
$\xrightarrow{m}$	
	$\boxed{\Pi}$ if queried after m: $b_f \leftarrow \mathsf{KA}.\mathcal{R}, \quad \Pi(f) = \mathsf{KA.msg}_2(b_f, m)$
$\xleftarrow{P}$	$P \leftarrow \mathbb{F}[x]$
$\xrightarrow{K}$	
	for fresh $\Pi(P(H_1(y_i)))$: $\quad b_{P(H_1(y))} \leftarrow \mathsf{KA}.\mathcal{R}$ extract $\hat{Y} = \left\{ y \middle\vert \exists k' : \left(y, \mathsf{KA.key}_2(b_{P(H_1(y))}, m), k'\right) \in \mathcal{O}_2 \text{ and } k' \in K \right\}$

Table 4: behavior of the simulator of malicious sender

□

Lemma 3.2. The protocol is **secure against a malicious receiver**, if KA is non-malleable.

Proof. (sketch) The simulator needs to

- Simulate the view of m.
  Just compute m honestly.
- Extract the set $\hat{X}$ that explains the effect of P.
  Notice that here we first do this extraction, and then pick proper set K according to the functionality output we obtain. Again, we may ask: What is the set $\hat{X}$ s.t. when honestly playing the protocol with this set, the receiver will send P to the sender?
  Intuitively, H_1 must have been queried on the values x_i's, and Π^{-1} must have been queried on the values m_i''s. Also, we are not willing to see $\Pi(f_i)$ being queried before $\Pi^{-1}(m_i')$ since, if it was the case, we cannot guarantee that receiver interpolate P by the correct value $f_i = \Pi^{-1}(\mathsf{KA.msg}_2(b_i, m))$.
- simulate the view of K.
  In addition to processing $z \in \hat{X} \cap Y$, we add redundant uniformly random key value to K to make it of proper size. The effect of doing this is non-negligible, which can be proved by the non-maleablity of underlying KA, with the aid of hybrid argument.

The complete behavior of simulator is given as table 5 □

3.5 Size of Adversary's Set

Notice that in the above construction, the input of each party is the same. What if one of the parties, say $\mathcal{P}_i$, has input size $n_i > n$? Or, if the adversary just prepare a large set? In the worst case, the malicious receiver pretend that it has the universal set of all possible items.

Sim (Receiver) with input $X = \{x_1, \cdots, x_n\} \subset \{0,1\}^*$	
$\boxed{H_1}$ $\mathcal{O}_1 = \{x :$ Adv queried $H_1(x)\}$ $\boxed{\Pi^{\pm}}$ $\mathcal{O}_\Pi = \{f : \Pi^{-1}(m) = f$ is queried, yet $\Pi(f) = m$ is not$\}$	
$a \leftarrow \mathsf{KA}.\mathcal{R}, \quad m = \mathsf{KA}.\mathsf{msg}_1(a)$	$\xrightarrow{m}$
	$\xleftarrow{P}$
extract $\hat{X} = \{x \mid x \in \mathcal{O}_1 \wedge P(H_1(x)) \in \mathcal{O}_\Pi\}$ receive functionality output $\mathcal{F}(\hat{X}, Y) = \hat{X} \cap Y = Z$ $k_z = \mathsf{KA}.\mathsf{key}_1(a, \Pi(P(H_1(z))))$ for all $z \in Z$ $K = \{H_2(z, k_z) \mid z \in Z\} \cup \{u_i : unif\}$ s.t. $\lvert K \rvert = X$	$\xrightarrow{K}$

Table 5: behavior of the simulator of malicious receiver

Can it obtain the intersection, which is exactly the set sender holds? Can the protocol itself prevent this from happening? By bounding the size of the set simulator extracts, it turns out that (we omit the proof here)

- For corrupted sender and $\hat{Y}$ extracted by simulator, $Pr[|\hat{Y}|] > n = \mathsf{negl}$
- For corrupted receiver and $\hat{X}$ extracted by simulator, $Pr[|\hat{X}|] > n' = \theta(n) = \mathsf{negl}$

i.e. the effect that a malicious sender preparing a large set is no more than preparing a set of cardinality n. (or $\theta(n)$ for malicious receiver) Therefore the PSI ideal functionality can be adjusted to the one given in table 6

$\mathcal{F}(X, Y) = (f_1(X, Y), f_2(X, Y))$:
Abort if $\lvert X \rvert > n$ and the receiver is honest, or if $\lvert X \rvert > n'$ and the receiver is corrupt.
Abort if $\lvert Y \rvert > n$.
return $f_1(X, Y) = X \cap Y$, $f_2(X, Y) = \perp$

Table 6: PSI ideal functionality

4 OT-based PSI

An oblivious pseudorandom function (OPRF) is a protocol between two parties in which one party, the sender, receives (or chooses) an uniform random seed s while the other party, the receiver, receives $F(s, r)$, where the input r is chosen by the receiver. More formally, the two parties realizes the following functionality

$$\mathcal{F}_{OPRF} : (\perp, r) \mapsto (s, F(s, r)),$$

where F is a PRF.

An OPRF protocol can be used to construct a PSI protocol. The two parties in the protocol are the client $\mathcal{C}$ and the server $\mathcal{S}$ with item sets $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ respectively. We first consider a simpler protocol, the *private set inclusion protocol*, where the client's set X has only one item x, i.e. $X = \{x\}$.

1. The two parties would participate in an OPRF protocol, with the item x as input. The client $\mathcal{C}$ would then receive $F(s, x)$ while the server $\mathcal{S}$ learns the seed s.

2. The server would then evaluate $F(s, \cdot)$ on all its items $y_i \in Y$ and send the set $\{F(s, y_i) : i \in [n]\}$ to $\mathcal{C}$.

3. The client $\mathcal{C}$ could then check whether $x \in Y$ by checking whether $F(s, x) \in \{F(s, y_i) : i \in [n]\}$.

With this private set inclusion protocol, a private set intersection (PSI) protocol can be constructed by repeating the inclusion protocol for each item x in the client's set. This reduction from the PSI protocol to the OPRF protocol allows us to focus our attention on constructing efficient OPRF protocols. One such construction, which appeared implicitly in [PSZ14] and was later improved in [Pin+15], relied on oblivious transfers (OT), thus giving this category of PSI constructions the name "**OT-based PSI**". The OT-extension protocol in [Ish+03] and the coding-theoretic interpretation of it ([KK13]) allowed this construction to be very efficient. In this section, we'll focus on the work of [Kol+16], which took the coding-theoretic interpretation to the extreme.

4.1 OPRF Protocol via the IKNP framework

We'll start by defining the oblivious transfer functionality and some of its variants.

Definition 4.1 (Oblivious Transfer (OT))**.** An oblivious transfer (OT) of l-bit messages is an two party protocol, denoted OT_l, that realizes the following functionality

$$\mathcal{F}_{OT} : (\{m_0, m_1\}, \sigma) \mapsto (\perp, m_\sigma),$$

where $m_0, m_1 \in \{0,1\}^l$ and $\sigma \in \{0,1\}$.

Definition 4.2 ($\binom{N}{1}$ Oblivious Transfer (OT))**.** An $\binom{N}{1}$ oblivious transfer (OT) of l-bit messages is an two party protocol, denoted $\binom{N}{1}$-OT_l, that realizes the following functionality

$$\mathcal{F}_{OT} : (\{m_0, m_1, \ldots, m_{N-1}\}, \sigma) \mapsto (\perp, m_\sigma),$$

where $m_0, m_1, \ldots, m_{N-1} \in \{0,1\}^l$ and $\sigma \in [N]$.

Definition 4.3 ($\binom{N}{1}$ Random Oblivious Transfer (ROT))**.** An $\binom{N}{1}$ random oblivious transfer (ROT) of l-bit messages is an two party protocol, denoted $\binom{N}{1}$-OT_l, that realizes the following functionality

$$\mathcal{F}_{OT} : (\perp, \sigma) \mapsto (\{m_0, m_1, \ldots, m_{N-1}\}, m_\sigma),$$

where $m_0, m_1, \ldots, m_{N-1} \leftarrow \{0,1\}^l$ and $\sigma \in [N]$.

OT can be constructed from public-key cryptography primitives, such as RSA or Diffie-Hellman. Therefore a single OT instance from these primitives are very expensive computationally. This is when OT-extension comes in handy. OT-extension protocols allow us to generate a large number of OT instances from just a few base OT instances plus some symmetric key operations. More formally, OT-extension allows us to reduce OT_l^m or ROT_l^m to OT_k^k where $m \gg k$.

The core idea of the IKNP OT-extension framework can be described as follows: suppose the receiver $\mathcal{R}$ has choice bits $r = (r_1, r_2, \ldots, r_m) \in \{0,1\}^m$. $\mathcal{R}$ would then sample random matrices $T, U \leftarrow \{0,1\}^{m \times k}$ such that

$$t_j \oplus u_j = \begin{cases} 0^k & \text{if } r_j = 0 \\ 1^k & \text{if } r_j = 1 \end{cases} \tag{1}$$

$\mathcal{R}$ and the sender $\mathcal{S}$ would then swap roles and participate in the OT_m^k protocol, where $\mathcal{R}$ sends the (t^i, u^i), $1 \leq i \leq k$, as the messages for the ith instance of OT_m^k while $\mathcal{S}$ samples a random vector $s \leftarrow \{0,1\}^k$ as the choice bits. As a result, $\mathcal{S}$ would receive messages $(q^i)_{i=1}^k$, where $q^i = t^i$ if $s[i] = 0$, $q^i = u^i$ otherwise. Let H be a random oracle. The sender $\mathcal{S}$ would then produce the m sets of messages $(m_{0,j}, m_{1,j})_{j=1}^m$ as the ROT_l^m output as

$$m_{0,j} = H(j \,\|\, q_j) \tag{2}$$
$$m_{1,j} = H(j \,\|\, q_j \oplus s) \tag{3}$$

The receiver $\mathcal{R}$ would then output its selected message $(m_{j,r[j]})_{j=1}^m$ as

$$m_{j,r[j]} = H(j \,\|\, t_j) \tag{4}$$

The key observation is that if we define the matrix $Q \in \{0,1\}^{m \times k}$ as $Q := [q^1, q^2, \ldots, q^k]$, then the jth row satisfies

$$q_j = t_j \oplus s \cdot (t_j \oplus u_j).$$

By (1), $q_j = t_j$ when $r_j = 0$ and $q_j = t_j \oplus s$ when $r_j = 1$. This shows the correctness of this protocol. The semi-honest security based on the real/ideal paradigm of this OT-extension protocol is shown in the following lemma:

Lemma 4.1 ([Ish+03])**.** Any distinguisher D which makes at most t calls to H can have at most a $(t+1) \cdot 2^{-k}$ advantage in distinguishing between the output of the real process and that of the ideal process.

Hence, the width of the OT-extension matrix T and U are set to the computational security parameter.

An implicit OPRF can be found in the protocol ROT_l^m. If we view the choice bits $r \in \{0,1\}^m$ as input and the random messages $(m_{0,j}, m_{1,j})_{j=1}^m$ the sender receives as the key, we can define the the PRF F as $F(r) := H(\bigoplus_{j=1}^m m_{r[j],j})$ where H is a random oracle. If we were to use this OPRF protocol as a single private set inclusion protocol, then the input r would the client's item. The communication cost for this single private set inclusion protocol would be the size of the OT-extension matrix, i.e. the size of T. Assume the items have σ bits in length and κ is the computational security parameter, then the OT-extension matrix has dimension $\sigma \times \kappa$, therefore has size $\sigma \cdot \kappa$.

4.2 A Coding-theoretic Interpretation of the IKNP framework

In [KK13], they viewed (1) as a row-wise *repetition encoding*. That is, each row of the matrix $T \oplus U$ encodes the choice 0/1 as $0^k/1^k$. By replacing repetition encoding with an linear error-correcting code $\mathcal{C} = \{C(1), C(2), \ldots, C(N)\}$ with N code words, each row can encode N choices. Therefore, with similar communication cost, the IKNP OT-extension framework can be modified to generate m instances of $\binom{N}{1}$-OT_l with k instances of OT_k.

To argue that the receiver only learns one string, suppose the receiver has choice $r_j \in [N]$ for the jth instance of the ROT. Then for any other choice $\tilde{r} \neq r_j$, the message associated with $\tilde{r}$ is

$$\begin{aligned} H(j \,\|\, q_j \oplus [s \cdot C(\tilde{r}] &= H(j \,\|\, t_j \oplus [s \cdot C(r_j)] \oplus [s \cdot C(\tilde{r})]) \\ &= H(j \,\|\, t_j \oplus [s \cdot (C(\tilde{r}) \oplus C(r_j))]) \end{aligned}$$

Notice that everything in this expression is known to the receiver except for s. Now if the minimum distance of the code $\mathcal{C}$ is κ, then the hamming weight of $C(\tilde{r}) \oplus C(r_j)$ is at least κ. Intuitively, this would require us to guess at least κ bits of s to violate the security.

[Kol+16] noticed that $\mathcal{C}$ does not need to have many properties of error-correcting code. For example, it does not need to be efficiently decodable. The only requirement is that for any two code words, the hamming distance is at least κ. In fact, it is sufficient for this property to hold with overwhelming probability over the choice of the code. In the following sections, we'll go over the core ideas of the OPRF construction in [Kol+16] and show how it is superior to previous constructions in terms of communication cost.

4.3 An OPRF Varaint and the BaRK-OPRF protocol

We first start by introducing the concept of a *d-Hamming correlation robust* hash function and *pseudorandom codes*(PRC).

Definition 4.4 (d-Hamming correlation robustness)**.** Let H be a hash function with input length n. Then H is **d-Hamming correlation robust** if, for all strings $z_1, \ldots, z_m \in \{0,1\}^*$, $a_1, \ldots, a_m, b_1, \ldots, b_m \in \{0,1\}^n$ with $\|b_i\|_H \geq d$, the following distribution, induced by random sampling $s \leftarrow \{0,1\}^n$, is pseudorandom

$$H(z_1 \| a_1 \oplus [b_1 \cdot s]), \ldots, H(z_m \| a_m \oplus [b_m \cdot s])$$

Definition 4.5 (Pseudorandom Code (PRC))**.** Let $\mathcal{C}$ be a family of functions. We say that $\mathcal{C}$ is a (d, ϵ)-**pseudorandom code** (PRC) if for all strings $x \neq x'$,

$$\Pr[C \leftarrow \mathcal{C} : \|C(x) \oplus C(x')\|_H < d] \leq 2^{-\epsilon}$$

That is, a (d, ϵ)-PRC guarantees the hamming distance between to codewords less than d with probability at most $2^{-\epsilon}$. A PRC can be instantiated by a PRF as in the following lemma.

Lemma 4.2. Suppose $F : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$ is a PRF. Define $\mathcal{C} := \{F(s, \cdot) : s \in \{0,1\}^\kappa\}$. Then $\mathcal{C}$ is a (d, ϵ)-PRC where:

$$2^{-\epsilon} = 2^{-n} \sum_{i=0}^{d-1} \binom{n}{i} + \mathsf{negl}(\kappa)$$

The goal right now is to generate m instances of OPRF protocol efficiently. This can be done similarly as the OT-extension protocol in [KK13]. But instead of generating a $\binom{N}{1}$-ROT$_l$ instance per row in the OT-extension matrix using error-correcting codes, we generate a $\binom{\infty}{1}$-ROT$_l$ per row using PRCs since PRCs can take arbitrary strings as input. Therefore, if we view the arbitrary string input as the PRF input, we can define the PRF F as follows:

$$F((q_j, s), r) = H(j \| q_j \oplus [s \cdot C(r)]) \tag{5}$$

That is, each row is now a PRF instance with key (q_j, s). Intuitively, the sender in the ROT protocol can evaluate the PRF at any point while the receiver can only evaluate at one point. There are a few subtleties with this PRF definition:

- The receiver learns more than just the output $F((q_j, s), r_j) = H(j \| t_j)$. It actually learns t_j and uses it to evaluate the PRF output $H(j \| t_j)$. Thus, this OPRF protocol leaks slightly more information than the PRF output.
- In this protocol, we realize many "OPRF" instances with related keys. Namely, the code C and the vector s are using in every OPRF instance.

In order to address the above subtleties, we'll define the notion of a *relaxed PRF* and a security game that captures the notion of security in face of a leaky OPRF protocol which uses related keys.

Definition 4.6 (Relaxed PRF)**.** Let F be a PRF. We say F is a *relaxed PRF* if there exists a function $\tilde{F}$ such that $F(k, r)$ can be efficiently computed given just $\tilde{F}(k, r)$.

We can think of the F defined in (5) as a relaxed PRF where $\tilde{F}$ is a function that outputs $t_j = q_j \oplus [s \cdot C(r)]$.

Definition 4.7 (m-related-key-PRF)**.** Let F be a relaxed PRF with output length v, for which we can write the seed as a pair (k^*, k). Then F has m-related-key-PRF (m-RK-PRF) security if the advantage of any PPT adversary in the following game is negligible:

1. The adversary chooses strings $x_1, \ldots, x_n$ and m pairs $(j_1, y_1), \ldots, (j_m, y_m)$, where $y_i \neq x_{j_i}$.
2. Challenger chooses random values appropriate for PRF seeds $k^*, k_1, \ldots, k_n$ and tosses a coin $b \leftarrow \{0, 1\}$.
   - If $b = 0$, the challenger outputs $\left\{\widetilde{F}\left((k^*, k_j), x_j\right)\right\}_j$ and $\left\{F\left((k^*, k_{j_i}), y_i\right)\right\}_i$.
   - If $b = 1$ the challenger chooses $z_1, \ldots, z_m \leftarrow \{0, 1\}^v$ and outputs $\left\{\widetilde{F}\left((k^*, k_j), x_j\right)\right\}_j$ and $\{z_i\}_i$,
3. The adversary outputs a bit b'. The advantage of the adversary is $\Pr[b = b'] - 1/2$.

We can interpret the m-related-key-PRF security game in the above definition as follows:

The functionality is parameterized by a relaxed prf f, a number m of instances, and two parties: a **sender** and **receiver**. On input $(r_1, \ldots, r_m)$ from the receiver,

- Choose random components for seeds to the PRF: $k^*, k_1, \ldots, k_m$ and give these to the sender.
- Give $\widetilde{F}\left(\left(k^*, k_1\right), r_1\right), \ldots, \widetilde{F}\left(\left(k^*, k_m\right), r_m\right)$ to the receiver.

Figure 1: Batched, related-key OPRF (BaRK-OPRF) ideal functionality

1. The adversary chooses n inputs $x_1, x_2, \ldots, x_n$, one for each of the n OPRF instances. It then chooses to see m outputs from the challenger, which could be m PRF outputs or m uniformally random outputs. If the challenger chooses to show the m PRF outputs, they are of the form $F((k^*, k_{j_i}), y_i)$, $1 \leq i \leq m$. Thus y_i are the inputs that challenger specifies to see and j_i is used to specify what key to use.

2. The challenger chooses the shared key component k^* and n key components $k_1, k_2, \ldots, k_n$ for each OPRF instance. That is, (k^*, k_i) is the key for the ith OPRF instance. The challenger then shows the adversaries the *relaxed* PRF outputs $\left\{\widetilde{F}\left(\left(k^*, k_j\right), x_j\right)\right\}_j$ and $\{z_i\}_i$.

3. In addition of showing the relaxed outputs, the challenger tosses a coin, and chooses whether to show the adversary m uniform outputs or m PRF outputs $\{F\left(\left(k^*, k_{j_i}\right), y_i\right)\}_i$ as specified by the adversaries output $(j_1, y_1), \ldots, (j_m, y_m)$.

4. The protocol is m-related-key-PRF secure if the adversary cannot distinguish whether the challenger showed it uniform outputs or PRF outputs. That is, the adversary cannot guess the coin toss b.

The following lemma instantiates the PRF F and its relaxed output function $\tilde{F}$ with our OT-extension context:

Lemma 4.3. Let $\mathcal{C}$ be a $(d, \epsilon + \log_2 m) - PRC$, where $1/2^\epsilon$ is a negligible function, Let H be a d-Hamming correlation robust hash function. Define the following relaxed PRF, for $C \in \mathcal{C}$:

$$F(((C, s), (\boldsymbol{q}, j)), r) = H(j \| \boldsymbol{q} \oplus [C(r) \cdot s])$$
$$\widetilde{F}(((C, s), (\boldsymbol{q}, j)), r) = (j, C, \boldsymbol{q} \oplus [C(r) \cdot s])$$

Then F has m-RK-PRF security.

We call this multi-instance OPRF protocol that uses related keys as the **batched, related-key OPRF** functionality. This functionality is formalized in figure 1. We refer to figure 2 in [Kol+16] for the BaRK-OPRF construction, which is essentially the OT-extension protocol in [KK13] with pseudorandom code instead of error-correcting code. This construction of the BaRK-ORPF functionality is semi-honest secure as shown in the following theorem.

Theorem 4.4. The BaRK-OPRF protocol in [Kol+16] figure 2 securely realizes the functionality of figure 1 instantiated with the relaxed PRF defined in lemma (4.3), in the presence of semi-honest adversaries, where κ is the computational security parameter.

The BaRK-OPRF ideal functionality itself does not provide the PRFs security. The PRF security of using related-keys while leaking relaxed outputs is given by lemma (4.3).

4.4 Improving Private Set Intersection with BaRK-OPRF

As mentioned before, an OPRF protocol can be used to construct a private set inclusion protocol and by repeating the inclusion protocol for each item in the client's set, we have a private set intersection (PSI) protocol. The BaRK-OPRF protocol allows us to efficiently construct multiple instances of OPRF.

Since under the hood of our BaRK-OPRF protocol is the IKNP OT-extension framework, the communication cost is given by the size of the OT-extension matrix. In [Kol+16], they showed that by tuning suitable parameters, a constant width of 448 for the OT-extension matrix is sufficient in most cases. This results in a 448 bit communication cost for each item in the client's set. In particular, this cost does not depend on the bit-length of the item's representation, which was the case in OPRF constructions via [Ish+03] or [KK13].

5 Related Works and Open Problems

Communication Lower Bounds of Threshold PSI A practical variant of PSI, called the *threshold PSI*, is for parties to privately learn the intersection only if the intersection is sufficiently large. An example for the usage of threshold PSI is a dating app, where two people are matched only if their common interest set is sufficiently large. In section 4, the goal was improving PSI protocols to achieve lower communication cost. [GS19] proves a communication complexity lower bound of $\Omega(t)$ for two-party threshold PSI, where t is the upper bound of the size of the set difference $|X \setminus (X \cap Y)| + |Y \setminus (X \cap Y)|$. They also show an almost matching upper bound of $\tilde{\mathcal{O}}(t)$ based on fully homomorphic encryption.

[Bad+21] generalized this study to the multi-party PSI setting. In the multi-party setting, the notion of "sufficiently large intersection" can be interpreted by two different definitions. We denote X_i are the ith party's set and $I = \bigcap_i X_i$ as the intersection. The first definition of sufficiently large intersection is to require $|X_i \setminus I| \leq t$ for all i. The ideal threshold PSI functionality adopting this definition is denoted $\mathcal{F}_{\mathsf{int}}$. The second definition is to require $|\bigcup_i X_i \setminus I| \leq t$. The ideal threshold PSI functionality adopting this definition is denoted $\mathcal{F}_{\mathsf{diff}}$. With 2 party PSI as a special case, they show a communication complexity upper bound of $\tilde{\mathcal{O}}(t)$ based on additive homomorphic encryption, which is a weaker assumption than fully homomorphic encryption. They also show a lower bound of $\Omega(n \cdot t)$ in point-to-point networks for both functionalities, where n is the number of parties, and for the $\mathcal{F}_{\mathsf{int}}$ functionality in broadcast networks. Lower bound for the other functionality $\mathcal{F}_{\mathsf{diff}}$ in broadcast networks is still an open problem.

References

[Yao86] Andrew Chi-Chih Yao. "How to generate and exchange secrets". In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. "How to Play ANY Mental Game". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. New York, New York, USA: Association for Computing Machinery, 1987, pp. 218–229. ISBN: 0897912217. DOI: 10.1145/28395.28420. URL: https://doi.org/10.1145/28395.28420.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES". In: *The Cryptographer's Track at RSA Conference*. 2001.

[Ish+03] Yuval Ishai et al. "Extending Oblivious Transfers Efficiently". In: *Advances in Cryptology - CRYPTO 2003*. Springer Berlin Heidelberg, 2003, pp. 145–161. DOI: 10.1007/978-3-540-45146-4_9. URL: https://doi.org/10.1007/978-3-540-45146-4_9.

[Gol04] Oded Goldreich. "Foundations of cryptography. II: Basic applications". In: 2 (May 2004). DOI: 10.1017/CBO9780511721656.

[CN08] Donghoon Chang and Mridul Nandi. "A Short Proof of the PRP/PRF Switching Lemma". In: *IACR Cryptology ePrint Archive* 2008 (Jan. 2008), p. 78.

[Ash+12] Gilad Asharov et al. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *IACR Cryptol. ePrint Arch.* 2011 (2012), p. 613.

[Ber+13] Daniel J. Bernstein et al. *Elligator: Elliptic-curve points indistinguishable from uniform random strings*. Cryptology ePrint Archive, Paper 2013/325. https://eprint.iacr.org/2013/325. 2013. DOI: 10.1145/2508859.2516734. URL: https://eprint.iacr.org/2013/325.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. "Improved OT Extension for Transferring Short Secrets". In: *Advances in Cryptology – CRYPTO 2013*. Springer Berlin Heidelberg, 2013, pp. 54–70. DOI: 10.1007/978-3-642-40084-1_4. URL: https://doi.org/10.1007/978-3-642-40084-1_4.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Faster Private Set Intersection Based on OT Extension". In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 797–812. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas.

[Pin+15] Benny Pinkas et al. "Phasing: Private Set Intersection Using Permutation-based Hashing". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 515–530. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas.

[Kol+16] Vladimir Kolesnikov et al. "Efficient Batched Oblivious PRF with Applications to Private Set Intersection". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016. DOI: 10.1145/2976749.2978381. URL: https://doi.org/10.1145/2976749.2978381.

[Lin17] Yehuda Lindell. "How to Simulate It – A Tutorial on the Simulation Proof Technique". In: Apr. 2017, pp. 277–346. ISBN: 978-3-319-57047-1. DOI: 10.1007/978-3-319-57048-8_6.

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018.

[GS19] Satrajit Ghosh and Mark Simkin. "The Communication Complexity of Threshold Private Set Intersection". In: *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing, 2019, pp. 3–29. DOI: 10.1007/978-3-030-26951-7_1. URL: https://doi.org/10.1007/978-3-030-26951-7_1.

[Cia+20] Michele Ciampi et al. *Round-Optimal and Communication-Efficient Multiparty Computation*. Cryptology ePrint Archive, Paper 2020/1437. https://eprint.iacr.org/2020/1437. 2020. URL: https://eprint.iacr.org/2020/1437.

[Ion+20a] Mihaela Ion et al. "On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)* (2020), pp. 370–389.

[Ion+20b] Mihaela Ion et al. "Private Intersection-Sum Protocols with Applications to Attributing Aggregate Ad Conversions". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2020, pp. 370–389. URL: https://eprint.iacr.org/2019/723.pdf.

[Lin20] Yehuda Lindell. *Secure Multiparty Computation (MPC)*. Cryptology ePrint Archive, Paper 2020/300. https://eprint.iacr.org/2020/300. 2020. DOI: 10.1145/3387108. URL: https://eprint.iacr.org/2020/300.

[Bad+21] Saikrishna Badrinarayanan et al. "Multi-party Threshold Private Set Intersection with Sublinear Communication". In: *Public-Key Cryptography – PKC 2021*. Springer International Publishing, 2021, pp. 349–379. DOI: 10.1007/978-3-030-75248-4_13. URL: https://doi.org/10.1007/978-3-030-75248-4_13.

[RT21] Mike Rosulek and Ni Trieu. *Compact and Malicious Private Set Intersection for Small Sets*. Cryptology ePrint Archive, Paper 2021/1159. https://eprint.iacr.org/2021/1159. 2021. DOI: 10.1145/3460120.3484778. URL: https://eprint.iacr.org/2021/1159.

[FY22] Dengguo Feng and Kang Yang. "Concretely efficient secure multi-party computation protocols: survey and more". In: *Security and Safety* (2022).