Fast *Unbalanced* Private Set Union from Fully Homomorphic Encryption

Binbin Tu, Yu Chen

Shandong University

May 4, 2022

Overview

- Background
- Motivation
- Our Contributions
- Related Works
- Overview of Our Techniques
- Implementation

Background

Parameters: Two parties: the sender $\mathcal{S}$ with set X and the receiver $\mathcal{R}$ with set Y.
Functionality:

1. Wait for an input $X = \{x_1, x_2, \cdots, x_m\} \subset \{0,1\}^*$ from sender $\mathcal{S}$, and an input $Y = \{y_1, y_2, \cdots, y_n\} \subset \{0,1\}^*$ from receiver $\mathcal{R}$.
2. Give output $X \cup Y$ to the receiver $\mathcal{R}$.

Figure: Ideal functionality $\mathcal{F}_{\text{PSU}}^{m,n}$ for private set union with one-sided output

- Private set union (PSU) allows two parties to compute the union of their sets without revealing anything except the union.
- Application: cyber risk assessment and management via joint IP blacklists and joint vulnerability data, privacy-preserving data aggregation, private ID, etc.

Motivation

- The balanced PSU
  - Most of the works on PSU are designed in the balanced case. These protocols typically perform only marginally better when one of the sets is much smaller than the other.
  - The communication cost of the balanced PSU is **at least linearly** with the size of the **larger** set.
- The unbalanced PSU
  - Jia et al. give a construction of uPSU* with shuffling technique, but their uPSU* protocol **leaks the information of the size of set intersection** to the sender. This is a critical information leakage for uPSU, in particular, the protocol leaks the set intersection when the sender inputs one-element set.
  - The communication cost of uPSU* is still **at least linearly** with the size of the **larger** set.

Motivation

- The unbalanced PSI
  - Chen et al. first consider unbalanced case and design an efficient unbalanced private set intersection (uPSI) based on the leveled fully homomorphic encryption (FHE). Their fast uPSI breaks the bound of communication complexity linear with the size of the larger set and achieves the communication complexity **linear in the size of the smaller set, and logarithmic in the larger set**.
- Open problem: *Is it possible to design a* ***secure*** *and* ***fast*** *unbalanced PSU protocol which breaks the bound of communication complexity linear with the size of the larger set?*

Our Contributions

- We first give a basic uPSU protocol based on fully homomorphic encryption with communication linear in the smaller set.

$\mathcal{S}(x_i)$		$\mathcal{R}(y_i \in Y, i \in [n])$
$c = \text{FHE.Enc}_{pk_S}(x_i)$	$\xrightarrow{c}$	$f(x) = \Pi_{y_i \in Y}(x - y_i)$
	$\xleftarrow{c_i'}$	$r_i \leftarrow \mathbb{Z}_q$, $c' = \text{FHE.Enc}_{pk_S}(f(x_i) + r_i)$
		$f(y_i) + r_i = \text{Dec}_{sk_R}(c_i')$
	$\xrightarrow{r_i' = r_i + f(x_i)}$	If $r_i = r_i'$, $x_i \in Y$
		If $r_i \neq r_i'$, $x_i \notin Y$

Figure: Basic uPSU from FHE

- high computational cost
- deep homomorphic circuit

Our Contributions

- We use an array of optimizations following [CLR17,CHL+18,CMG+21], such as cuckoo hashing, batching, windowing, partitioning, modulus switching, etc, to get an uPSU* with optimizations.

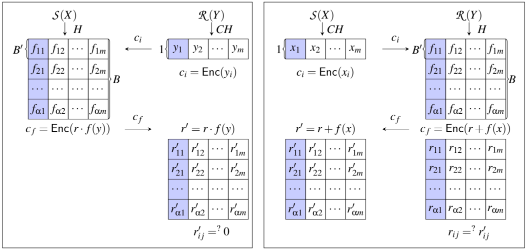


Figure 3: Comparison of uPSI [10] (left) and our basic uPSU with optimizations (right)

- Problems:
  - The receiver not only gets the set union, but also knows that some of its subsets have the item in the set intersection.
  - The worst case. The partitioning may leaks the element of the set intersection

Our Contributions

- We introduce a new cryptographic protocol named **matrix private equality test with permutation (mPEQT-wP)**.
- We **instantiate our mPEQT-wP** efficiently and obtain secure and fast uPSU protocols.
  - We construct mPEQT-wP based on permute and share protocol and multi-point oblivious pesudorandom function (mp-OPRF), and this protocol requires the communication cast $O(m \log m)$.
  - We construct mPEQT-wP based on decisional Diffie-Hellman (DDH) assumption and this protocol requires the communication cast $O(m)$.

Related Works

- We revisit some recent private set operation protocols including uPSU[JSZ+22], PSU[KRTW19,GMR+21,JSZ+22,ZCL+22] and uPSI protocols[CLR17], and show the communication cost and security comparison of PSU in the semi-honest setting.

Table: Communication Comparison of PSU in the semi-honest setting

Protocols	Communication	Security
PSU* [KRTW19]	$O(n \log n)$	$-\!\!\!\!\checkmark$
PSU[GMR+21]	$O(n \log n)$	$\checkmark$
PSU[ZCL+22]	$O(n)$	$\checkmark$
PSU[JSZ+22]	$O(n \log n)$	$\checkmark$
uPSU* [JSZ+22]	$O(n + m \log m)$	$=\!\!\!\!\checkmark$
Our uPSU	$O(m \log n)$	$\checkmark$

‡ n denotes the size of the large set, and m denotes the size of the small set. PSU* denotes it is not full secure. $-\!\!\!\!\checkmark$ denotes the PSU leaks information of some subsets have the items in the set intersection. $=\!\!\!\!\checkmark$ denotes the PSU leaks information of the size of the set intersection.

Related Works

- uPSI [CLR17]

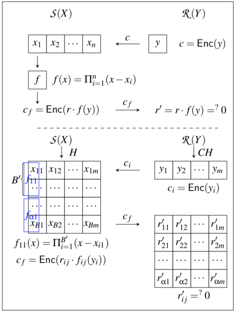
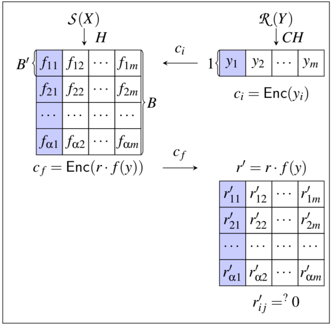


Figure 1: uPSI protocol and its optimizations [10]



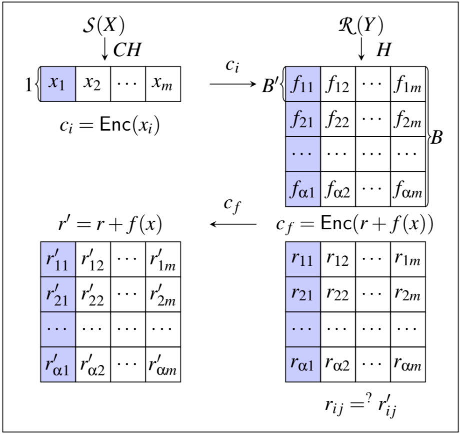
Overview of Our Techniques

- We start with our basic uPSU protocol based on FHE

$\mathcal{S}(x_i)$		$\mathcal{R}(y_i \in Y, i \in [n])$
$c = \mathsf{FHE.Enc}_{pk_S}(x_i)$	$\xrightarrow{c}$	$f(x) = \Pi_{y_i \in Y}(x - y_i)$
	$\xleftarrow{c'_i}$	$r_i \leftarrow \mathbb{Z}_q$, $c' = \mathsf{FHE.Enc}_{pk_S}(f(x_i) + r_i)$
		$f(y_i) + r_i = \mathsf{Dec}_{sk_R}(c'_i)$
	$\xrightarrow{r'_i = r_i + f(x_i)}$	If $r_i = r'_i$, $x_i \in Y$
		If $r_i \neq r'_i$, $x_i \notin Y$

Figure: Basic uPSU from FHE

Overview of Our Techniques



- Leak some subsets have the items in the set intersection.
- Worst: leak the items in the set intersection.
  - If $r_{22} = r'_{22}$, $x_2 \in X \cap Y$, $f_{i2}(x_2)$ to $\mathcal{S}$.

Overview of Our Techniques

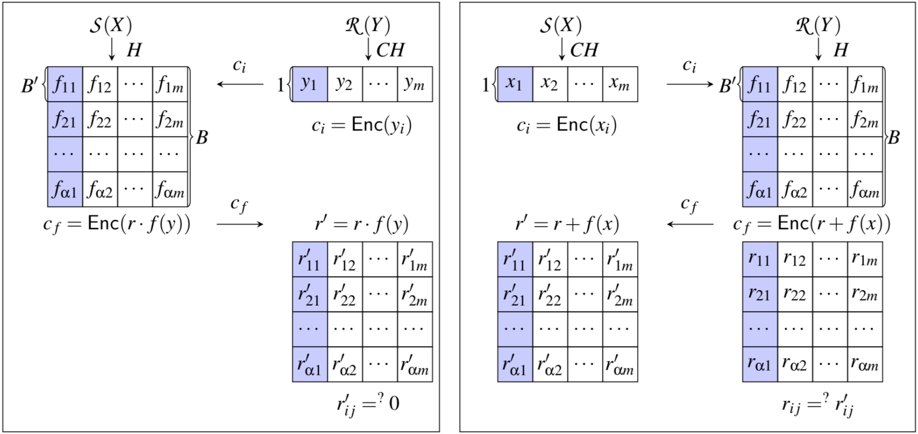


Figure 3: Comparison of uPSI [10] (left) and our basic uPSU with optimizations (right)

Overview of Our Techniques

- We develop a new cryptographic protocol named matrix private equality test with permutation (mPEQT-wP)
- Compare with the private equality test (PEQT), mPEQT-wP can provide matrix private equality test with positions permutation

Parameters: Two parties: P_1 with a matrix R, P_0 with a matrix R' and a permutation $\pi = (\pi_c, \pi_r)$, where π_c (over $\{1, 2, \cdots, m\}$) and π_r (over $\{1, 2, \cdots, \alpha\}$),

$$\mathbf{R} = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ r_{21} & \cdots & r_{2m} \\ \vdots & \ddots & \vdots \\ r_{\alpha 1} & \cdots & r_{\alpha m} \end{bmatrix}, \mathbf{R}' = \begin{bmatrix} r'_{11} & \cdots & r'_{1m} \\ r'_{21} & \cdots & r'_{2m} \\ \vdots & \ddots & \vdots \\ r'_{\alpha 1} & \cdots & r'_{\alpha m} \end{bmatrix}$$

Functionality:

1. Wait for an input $\mathbf{R}'$ and a permutation $\pi = (\pi_c, \pi_r)$ from P_0, and an input $\mathbf{R}$ from P_1.
2. Give the bit matrix $\mathbf{B}$ to P_1, where

$$\mathbf{B} = \begin{bmatrix} b_{\pi(11)} & \cdots & b_{\pi(1m)} \\ b_{\pi(21)} & \cdots & b_{\pi(2m)} \\ \vdots & \ddots & \vdots \\ b_{\pi(\alpha 1)} & \cdots & b_{\pi(\alpha m)} \end{bmatrix},$$

if $r_{\pi(ij)} = r'_{\pi(ij)}$, $b_{\pi(ij)} = 1$, else, $b_{\pi(ij)} = 0$, for $i \in [\alpha]$, $j \in [m]$.

Figure 4: Matrix private equality test with permutation $\mathcal{F}_{\text{mPEQT-wP}}$

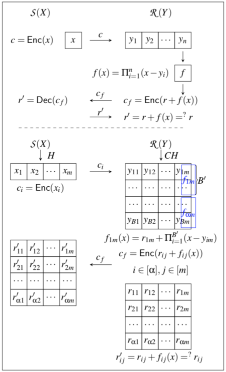


Figure 6: uPSU protocol and its optimizations

Overview of Our Techniques

- We construct mPEQT-wP from permute and share protocol [GMR+21, JSZ+22] and multi-point oblivious pesudorandom function (mp-OPRF), and this protocol requires the communication cast $O(m \log m)$
- We construct mPEQT-wP based on decisional Diffie-Hellman (DDH) assumption and this protocol requires the communication cast $O(m)$

Input: The receiver inputs a matrix $\mathbf{R} = [r_{ij}]$, $i \in [\alpha]$, $j \in [m]$; the sender inputs a matrix $\mathbf{R}' = [r'_{ij}]$, $i \in [\alpha]$, $j \in [m]$ and a permutation $\pi = (\pi_c, \pi_r)$ where π_c (over $\{1, 2, \cdots, m\}$) and π_r (over $\{1, 2, \cdots, \alpha\}$).
Output: The receiver outputs a bit matrix $\mathbf{B}$; the sender outputs $\perp$.

1. [PS functionality] $\mathcal{S}$ and $\mathcal{R}$ invoke the ideal permute and share functionality $\mathcal{F}_{\mathrm{PS}}$ twice: first, both parties permute and share the columns of $\mathbf{R}$, where each column of $\mathbf{R}$ can be seen as an item. $\mathcal{R}$ inputs each column $\mathbf{r_j}$, $j \in [m]$ of $\mathbf{R}$ and $\mathcal{S}$ inputs the permutation π_c. As a result, $\mathcal{R}$ gets $\mathbf{S}_{\pi_c} = [s_{\pi_c(ij)}]$ and $\mathcal{S}$ gets $\mathbf{S}'_{\pi_c} = [s'_{\pi_c(ij)}]$, where $s_{\pi_c(ij)} \oplus s'_{\pi_c(ij)} = r_{\pi_c(ij)}$. Then both parties permute and share the rows of $\mathbf{S}_{\pi_c}$, where each rows of $\mathbf{S}_{\pi_c}$ can be seen as an item. $\mathcal{R}$ inputs each rows of $\mathbf{S}_{\pi_c}$ and $\mathcal{S}$ inputs the permutation π_r. As a result, $\mathcal{R}$ gets $\mathbf{S}_{\pi_r} = [s_{\pi_r(ij)}]$ and $\mathcal{S}$ gets $\mathbf{S}'_{\pi_r} = [s'_{\pi_r(ij)}]$, where $s_{\pi_r(ij)} \oplus s'_{\pi_r(ij)} = s_{\pi_c(ij)}$. Finally, $\mathcal{R}$, $\mathcal{S}$ gets the shuffled matrix shares $\mathbf{S}_\pi = \mathbf{S}_{\pi_r}$ and $\mathbf{S}'_\pi = \pi_r(\mathbf{S}'_{\pi_c}) \oplus \mathbf{S}'_{\pi_r}$, where $s_{\pi(ij)} \oplus s'_{\pi(ij)} = \pi(r_{ij})$, $i \in [\alpha]$, $j \in [m]$.
2. [mp-OPRF functionality] $\mathcal{R}$ acts as P_0 with shuffled shares $\mathbf{S}$, and obtains the outputs $F_k(s_{\pi(ij)})$, $i \in [\alpha]$, $j \in [m]$, and $\mathcal{S}$ obtain the key k.
3. $\mathcal{S}$ computes $F_k(r'_{\pi(ij)} \oplus s'_{\pi(ij)})$, $i \in [\alpha]$, $j \in [m]$ and sends them to $\mathcal{R}$.
4. $\mathcal{R}$ sets $b_{\pi(ij)} = 1$, if $F_k(s_{\pi(ij)}) = F_k(r'_{\pi(ij)} \oplus s'_{\pi(ij)})$, else, $b_{\pi(i)} = 0$, and gains a bit matrix $\mathbf{B} = [b_{\pi(ij)}]$, $i \in [\alpha]$, $j \in [m]$.

Figure 11: mPEQT-wP from PS and mp-OPRF

Input: The receiver inputs a matrix $\mathbf{R} = [r_{ij}]$, $i \in [\alpha]$, $j \in [m]$; the sender inputs a matrix $\mathbf{R}' = [r'_{ij}]$, $i \in [\alpha]$, $j \in [m]$ and a permutation $\pi = (\pi_c, \pi_r)$ where π_c (over $\{1, 2, \cdots, m\}$) and π_r (over $\{1, 2, \cdots, \alpha\}$).
Output: The receiver outputs a bit matrix $\mathbf{B}$; the sender outputs $\perp$.

1. $\mathcal{R}$ and $\mathcal{S}$ choose random number a, b and compute $H_{ij} = H(r_{ij})^a, H'_{ij} = H(r'_{ij})^b$ for $i \in [\alpha]$, $j \in [m]$, where $H = H(\cdot)$ are (multiplicative) group elements output by hash functions H. $\mathcal{R}$ sends $H_{ij} = H(r_{ij})^a$ to $\mathcal{S}$.
2. $\mathcal{S}$ computes $H''_{ij} = (H(r_{ij})^a)^b$ and uses the permutation $\pi = (\pi_c, \pi_r)$ and computes $H''_{\pi(ij)} = \pi(H''_{ij})$, $H'_{\pi(ij)} = \pi(H'_{ij})$ and sends them to $\mathcal{R}$.
3. $\mathcal{R}$ set $b_{\pi(ij)} = 1$, if $H''_{\pi(ij)} = H'^{a}_{\pi(ij)}$, else $b_{\pi(ij)} = 0$, and gains a bit matrix $\mathbf{B} = [b_{\pi(ij)}]$, $i \in [\alpha]$, $j \in [m]$.

Figure 12: Instantiation of mPEQT-wP based on DDH

Overview of Our Techniques

- We start with our basic uPSU protocol based on FHE and using optimization techniques [CLR17, CHL+18, CMG+21] to divide a large PSU into many small PSU to reduce the depth of homomorphic circuit.
- Then, by using mPEQT-wP and OT protocol, we can obtain secure and fast uPSU protocols that is secure against semi-honest adversaries

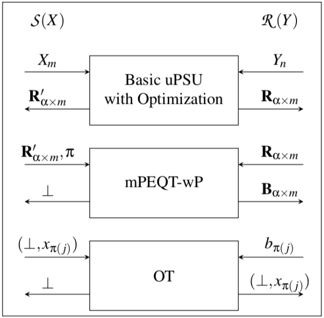


Figure 5: Core design idea of Our full uPSU protocol

Implementation

- Hash+FHE [CLR17, CHL+18, CMG+21]
- mPEQT-wP: PS [GMR+21, JSZ+22] + mpOPRF [JSZ+22]
- Compare with uPSI [CLR17]
  - padding the matrix R