Unbalanced Private Set Union with Reduced Computation and Communication

Cong Zhang[1] Yu Chen[2] Weiran Liu[3] Liqiang Peng[3] Meng Hao[4]
Anyu Wang[1] Xiaoyun Wang[1]

[1] Tsinghua University
[2] Shandong University
[3] Alibaba Group
[4] Singapore Management University

ACM CCS 2024, October 14-18

Outline

Outline

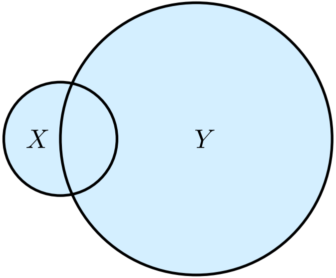
Unbalanced Private Set Union

Sender

$X = \{x_1, \ldots, x_m\}$

Receiver

$Y = \{y_1, \ldots, y_n\}$



$|X| \ll |Y|$

Learns nothing.

Learns $X \cup Y$.

Unbalanced Private Set Union

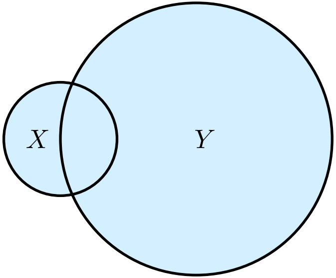
Sender

$X = \{x_1, \ldots, x_m\}$

Receiver

$Y = \{y_1, \ldots, y_n\}$

Learns nothing,



$|X| \ll |Y|$

Learns $X \cup Y$,
but knows nothing about $X \cap Y$.

Applications

PSU has found numerous applications, which include but not limit to:

- information security risk assessment [LV04]
- IP blacklist and vulnerability data aggregation [HLS$^+$16]
- joint graph computation [BS05]
- distributed network monitoring [KS05]
- building block for private DB supporting full join [KRTW19]
- private ID [GMR$^+$21]

Previous Work

Over the last few years, a line of works have greatly improved the efficiency of PSU in the semi-honest model.

1. PSU for balanced setting [KRTW19, GMR$^+$21, JSZ$^+$22, ZCL$^+$23, CZZ$^+$24].
   - Use cheap symmetric-key techniques coupled with OT.
   - Do not target unbalanced case specifically.
   - Communication cost is at least $O(|Y|)$ ($|Y| \gg |X|$).
2. PSU for unbalanced setting [TCLZ23].
   - Used lattice-based homomorphic encryption.
   - Communication cost is sublinear in $|Y|$.
   - Do not consider preprocessing optimization.
   - Has large constant due to patition technique.

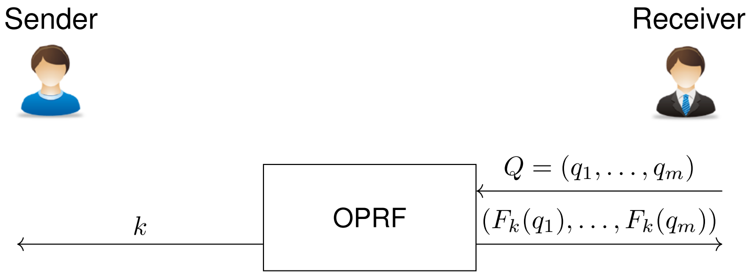
Our Result

We propose two new constructions for unbalanced PSU protocols based on different oblivious key-value store (OKVS) encoding strategies.

1. Based on one-time pads generated from oblivious PRF.
   - Sublinear communication complexity in $|Y|$.
   - New cryptographic preliminaries: permuted private equality test (p-PEQT).
     - DDH-based construction.
     - Permute+Share based construction.
2. Based on re-randomized public-key encryption.
   - Sublinear communication complexity in $|Y|$.
   - New ”pull-down-then-lift” methodology in OKVS encoding.

Outline

Oblivious PRF (OPRF)



Batch Private Information Retrieval (BatchPIR)

A BatchPIR scheme consists of three algorithms $(\mathsf{Query}, \mathsf{Answer}, \mathsf{Recover})$:

- $\mathsf{Query}(I = \{i_1, \ldots, i_b\} \in ([n])^b) \to (\mathsf{qu}, \mathsf{st})$
- $\mathsf{Answer}(D, \mathsf{qu}) \to \mathsf{ans}$
- $\mathsf{Recover}(\mathsf{st}, \mathsf{ans}) \to \{D_1, \ldots, D_b\}$

Correctness. For any dataset D, all distinct inputs $I = \{i_1, \ldots, i_b\}$, $(\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I)$:

$$\mathsf{Recover}(\mathsf{st}, \mathsf{Answer}(D, \mathsf{qu})) = \{D[i_1], \ldots, D[i_b]\}.$$

Query privacy. For all PPT adversaries $\mathcal{A}$ and all distinct batch query sets I_1, I_2 with $|I_1| = |I_2|$,

$$\Pr[\mathcal{A}(\mathsf{qu}) = 1 \mid (\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I_1)] - \Pr[\mathcal{A}(\mathsf{qu}) = 1 \mid (\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I_2)] \leq \mathsf{negl}(\kappa).$$

Batch Private Information Retrieval (BatchPIR)

A BatchPIR scheme consists of three algorithms $(\mathsf{Query}, \mathsf{Answer}, \mathsf{Recover})$:

- $\mathsf{Query}(I = \{i_1, \dots, i_b\} \in ([n])^b) \to (\mathsf{qu}, \mathsf{st})$
- $\mathsf{Answer}(D, \mathsf{qu}) \to \mathsf{ans}$
- $\mathsf{Recover}(\mathsf{st}, \mathsf{ans}) \to \{D_1, \dots, D_b\}$

Correctness. For any dataset D, all distinct inputs $I = \{i_1, \dots, i_b\}$, $(\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I)$:

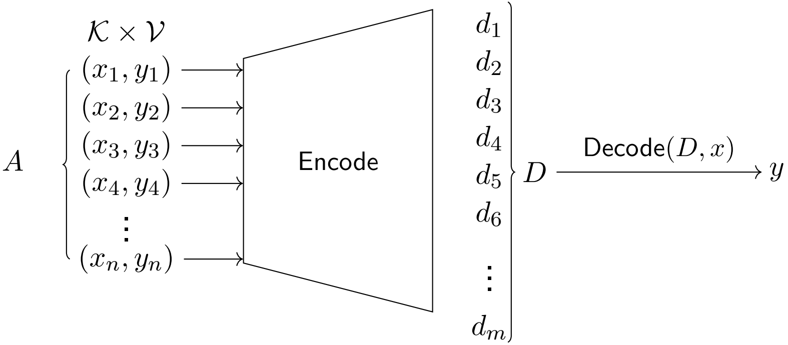
$$\mathsf{Recover}(\mathsf{st}, \mathsf{Answer}(D, \mathsf{qu})) = \{D[i_1], \dots, D[i_b]\}.$$

Query privacy. For all PPT adversaries $\mathcal{A}$ and all distinct batch query sets I_1, I_2 with $|I_1| = |I_2|$,

$$\Pr[\mathcal{A}(\mathsf{qu}) = 1 \mid (\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I_1)] - \Pr[\mathcal{A}(\mathsf{qu}) = 1 \mid (\mathsf{st}, \mathsf{qu}) \leftarrow \mathsf{Query}(I_2)] \leq \mathsf{negl}(\kappa).$$

Remark. Since (Batch)PIR does not protect database privacy, the server can directly send the entire database to the client, resulting in communication $O(n)$. We do not consider this trivial (Batch)PIR construction and always assume that the communication of (Batch)PIR is sublinear in n, i.e., $|\mathsf{qu}| + |\mathsf{ans}| = o(n)$.

Oblivious Key-Value Store



rate n/m
1 is optimal

Correctness. For all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys, $(x, y) \in A$, Decode(Encode$(A), x) = y$.
Obliviousness. For any $(x_1^0, \ldots, x_n^0) \neq (x_1^1, \ldots, x_n^1)$:

$$\text{Encode}((x_1^0, y_1), \ldots, (x_n^0, y_n)) \approx \text{Encode}((x_1^1, y_1), \ldots, (x_n^1, y_n)), \text{ where } y_i \xleftarrow{R} \mathcal{V}.$$

Randomness. For any $A = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x \notin \{x_1, \ldots, x_n\}$:

$$\text{Decode}(D, x) \approx U_{\mathcal{V}}, \text{ where } D \leftarrow \text{Encode}(A).$$

Oblivious Key-Value Store

All state-of-the-art OKVS schemes are binary linear OKVS. That is, the Encode algorithm is to solve the following linear equation:

$$\begin{bmatrix} -\mathsf{row}(x_1)- \\ -\mathsf{row}(x_2)- \\ \vdots \\ -\mathsf{row}(x_n)- \end{bmatrix}_{n\times m} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{1}$$

where $\mathsf{row} : \mathcal{K} \to \{0,1\}^m$ is defined by the random tapes of Encode algorithm.
The Decode algorithm is defined as:

$$\mathsf{Decode}(D, x) = \langle \mathsf{row}(x), D\rangle := \textstyle\sum_{j=1}^{m} \mathsf{row}(x)_j d_j = \sum_{\mathsf{row}(x)_j=1} d_j$$

Sparse OKVS

Our key observation: the binary vector $\mathsf{row}(x)$ has a long sparse part!

$$\mathsf{row}(x) = \underbrace{\mathsf{sparse}(x)}_{\text{constant weight } w} \;||\; \underbrace{\mathsf{dense}(x)}_{\text{random}} \in \{0,1\}^{s+d}$$

where $s = O(n), d = o(n)$.
As a result,

$$\begin{aligned}\mathsf{Decode}(D, x) &= \langle \mathsf{row}(x), D \rangle = \langle \mathsf{sparse}(x), D_0 \rangle + \langle \mathsf{dense}(x), D_1 \rangle \\ &= \sum_{\mathsf{sparse}(x)_i = 1} d_i + \langle \mathsf{dense}(x), D_1 \rangle\end{aligned}$$

where $|D_0| = s = O(n), |D_1| = d = o(n)$.

Typical Use Cases of OKVS

OKVS (with different instantiations) is widely used in private set operation protocols, e.g., PSI [PRTY19, GPR+21, RS21, RR22], PSU [KRTW19, GMR+21, ZCL+23, LG23].
A typical use case is as follows:

Sender$(A = \{(x_1, y_1), \ldots, (x_n, y_n)\})$

Receiver(q)

$D :=$ Encode(A)

$$\xrightarrow{\quad\quad\quad\quad D \quad\quad\quad\quad}$$

$y^* :=$ Decode(D, q)

Typical Use Cases of OKVS

OKVS (with different instantiations) is widely used in private set operation protocols, e.g., PSI [PRTY19, GPR+21, RS21, RR22], PSU [KRTW19, GMR+21, ZCL+23, LG23].
A typical use case is as follows:

Sender$(A = \{(x_1, y_1), \ldots, (x_n, y_n)\})$

Receiver(q)

$D :=$ Encode(A)

$$\xrightarrow{\quad\quad D \quad\quad}$$

$y^* :=$ Decode(D, q)

The obliviousness ensures sending D directly will not leak information about $\{x_1, \ldots, x_n\}$.

Typical Use Cases of OKVS

OKVS (with different instantiations) is widely used in private set operation protocols, e.g., PSI [PRTY19, GPR$^+$21, RS21, RR22], PSU [KRTW19, GMR$^+$21, ZCL$^+$23, LG23].
A typical use case is as follows:

Sender$(A = \{(x_1, y_1), \ldots, (x_n, y_n)\})$

Receiver(q)

$D :=$ Encode(A)

$$\xrightarrow{\quad D \quad}$$

Comm:$O(n)$.

$y^* :=$ Decode(D, q)

Communication-Efficient Decoding (CED)

Sender$(A = \{(x_1, y_1), \ldots, (x_n, y_n)\})$

Receiver(q)

$D = D_0 || D_1 := \text{Encode}(A)$

$\xrightarrow{\qquad\qquad D_1 \qquad\qquad}$

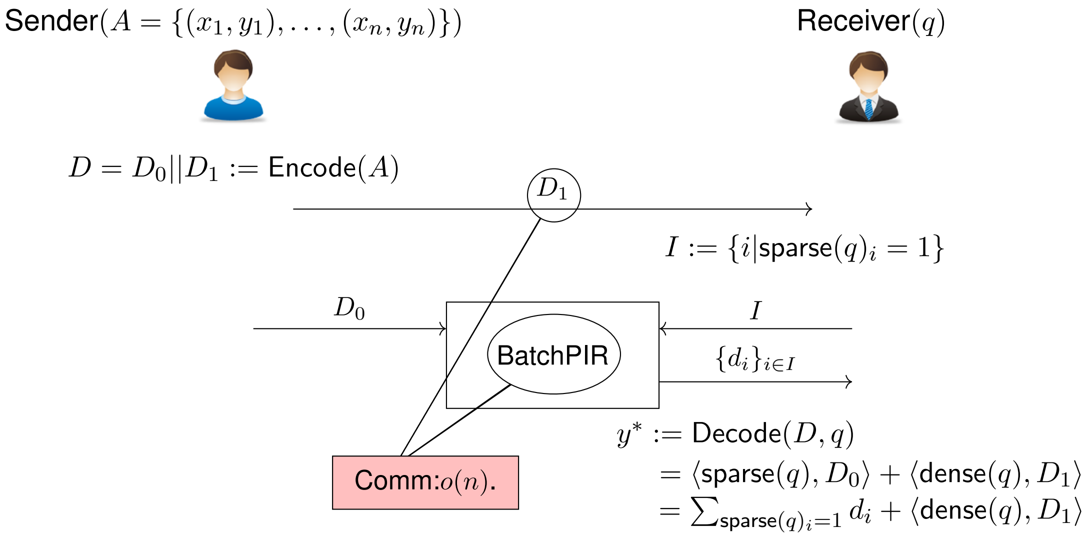
$I := \{i | \text{sparse}(q)_i = 1\}$

$\xrightarrow{\quad D_0 \quad}$ $\boxed{\text{BatchPIR}}$ $\xleftarrow{\quad I \quad}$

$\xrightarrow{\quad \{d_i\}_{i \in I} \quad}$

$$\begin{aligned} y^* &:= \text{Decode}(D, q) \\ &= \langle \text{sparse}(q), D_0 \rangle + \langle \text{dense}(q), D_1 \rangle \\ &= \sum\nolimits_{\text{sparse}(q)_i = 1} d_i + \langle \text{dense}(q), D_1 \rangle \end{aligned}$$

Communication-Efficient Decoding (CED)



Outline

Permuted Private Equality Test (p-PEQT)

Sender

Receiver

$$\vec{r'} = (r'_1, \dots, r'_n), \pi \longrightarrow \boxed{\text{p-PEQT}} \longleftarrow \vec{r} = (r_1, \dots, r_n)$$

$$\boxed{\text{p-PEQT}} \longrightarrow \vec{b} = (b_1, \dots, b_n)$$

$$b_i = \begin{cases} 1 & r_{\pi(i)} = r'_{\pi(i)}; \\ 0 & r_{\pi(i)} \neq r'_{\pi(i)} \end{cases}$$

p-PEQT from DDH

Sender($\vec{r}'$, π) | Receiver($\vec{r}$)

$$a \leftarrow \mathbb{Z}_q, v_i := H(r_i)^a, i \in [n]$$

$$\xleftarrow{\{v_i\}_{i \in [n]}}$$

$$b \leftarrow \mathbb{Z}_q$$

$$v'_i := H(r'_{\pi(i)})^b, \bar{v}_i = (v_{\pi(i)})^b, i \in [n]$$

$$\xrightarrow{\{v'_i, \bar{v}_i\}_{i \in [n]}}$$

$$b_i = \begin{cases} 1 & v'^a_i = \bar{v}_i; \\ 0 & v'^a_i \neq \bar{v}_i \end{cases}$$

Our First Construction

Sender$(X = \{x_1, \ldots, x_m\})$ 

Receiver$(Y = \{y_1, \ldots, y_n\})$ 

$X^* := CuckooHash(X)$

$Y^* := SimpleHash(Y)$

$\vec{r} \leftarrow \mathbb{F}^{m_c}$

Sender $\xrightarrow{X^*}$ **OPRF**

Sender $\xleftarrow{Q = \{q_i\}_{i \in [m_c]}}$ **OPRF** $\xrightarrow{k}$ Receiver

$q_i = F_k(X^*[i]), i \in [m_c]$

$i \in [m_c], \alpha \in [|Y^*[i]|]:$

$k_{i,\alpha} := Y^*[i][\alpha]$

$v_{i,\alpha} := r_i + F_k(Y^*[i][\alpha])$

Sender $\xrightarrow{X^*}$ **CED** $\xleftarrow{\{(k_{i,\alpha}, v_{i,\alpha})\}}$ Receiver

Sender $\xleftarrow{\{v'_i\}_{i \in [m_c]}}$ **CED**

$r'_i = v'_i - q_i, i \in [m_c]$

Our First Construction

Sender$(X = \{x_1, \ldots, x_m\})$

Receiver$(Y = \{y_1, \ldots, y_n\})$

$\pi \leftarrow$ Perm$([m_c])$

$$\xrightarrow{\vec{r}', \pi} \boxed{\text{p-PEQT}} \xleftarrow{\vec{r}}$$

$$\boxed{\text{p-PEQT}} \xrightarrow{\vec{b}}$$

$$b_i = \begin{cases} 1 & X^*[\pi(i)] \in Y^*[\pi(i)]; \\ 0 & X^*[\pi(i)] \notin Y^*[\pi(i)] \end{cases}$$

$i \in [m_c]$:

$$\xrightarrow{(X^*[\pi(i)], \perp)} \boxed{\text{OT}} \xleftarrow{b_i}$$

$$\boxed{\text{OT}} \xrightarrow{z_i}$$

$$X \cup Y := Y \cup \{z_i\}$$

Our Second Construction

Sender($X = \{x_1, \ldots, x_m\}$)

Receiver($Y = \{y_1, \ldots, y_n\}$)

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s_i := \mathsf{Enc}(pk, 0), i \in [n]$

Sender $\xrightarrow{X}$ **CED** $\xleftarrow{\{(y_i, s_i)\}_{i \in [n]}}$ Receiver

CED $\xrightarrow{S' = \{s'_j\}_{j \in [m]}}$ Sender

$\bar{s}_j := \mathsf{ReRand}(s'_j), j \in [m]$

Sender $\xrightarrow{\{\bar{s}_j\}_{j \in [m]}}$ Receiver

$$b_j = \begin{cases} 1 & \mathsf{Dec}(sk, \bar{s}_j) = 0; \\ 0 & \mathsf{Dec}(sk, \bar{s}_j) \neq 0 \end{cases}$$

$i \in [m_c]$:

Sender $\xrightarrow{(x_j, \perp)}$ **OT** $\xleftarrow{b_j}$ Receiver

OT $\xrightarrow{z_j}$ Receiver

$X \cup Y$
$:= Y \cup \{z_j\}$

Outline

Implementation

we implement our protocols and fully re-implement prior unbalanced PSU [TCLZ23] based on the same open-source library mpc4j[1].

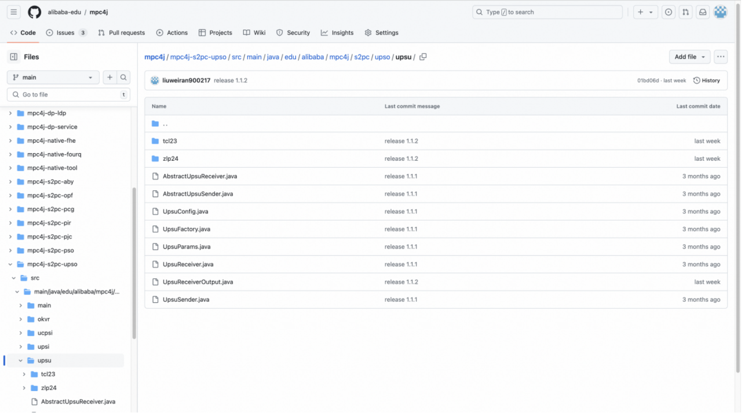


Figure: Our implementation

[1] https://github.com/alibaba-edu/mpc4j

Performance

Param.		Protocol	Comm. (MB)		Running time (s)							
n	m		Setup	Online	LAN		100Mbps		10Mbps		1Mbps	
					Setup	Online	Setup	Online	Setup	Online	Setup	Online
2^{18}	2^4	TCLZ-pub [TCLZ23]	0.76	1.52	2.98	1.15	3.67	1.99	3.93	3.11	9.71	14.46
		TCLZ-sym [TCLZ23]	0.77	2.27	2.98	1.18	4.02	2.48	4.50	4.04	10.25	20.90
		Our PSU$_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	0.71	11.72	1.13	15.55	2.44	29.60	2.89	174.53	8.30
		Our PSU$_{\mathsf{op}}^{\mathsf{ps}}$	19.09	0.73	12.05	1.14	15.02	2.71	28.79	3.27	173.45	9.29
		Our PSU$_{\mathsf{pk}}$	19.08	0.74	38.01	7.52	40.16	8.11	52.31	8.32	197.53	13.92
	2^6	TCLZ-pub [TCLZ23]	0.76	1.52	2.84	1.12	3.38	2.01	3.90	3.03	9.59	14.50
		TCLZ-sym [TCLZ23]	0.77	2.27	2.83	1.11	3.72	2.88	4.15	4.44	10.13	21.93
		Our PSU$_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	0.72	14.99	0.86	17.19	1.72	31.25	2.18	175.06	8.04
		Our PSU$_{\mathsf{op}}^{\mathsf{ps}}$	19.09	0.75	14.00	0.88	17.72	2.00	30.94	2.36	175.34	7.98
		Our PSU$_{\mathsf{pk}}$	19.08	0.74	34.88	2.32	37.57	2.92	52.14	3.62	196.42	9.13
	2^8	TCLZ-pub [TCLZ23]	0.76	1.53	2.85	1.15	3.48	1.91	3.92	3.08	9.55	14.66
		TCLZ-sym [TCLZ23]	0.77	2.28	2.87	1.13	3.89	2.25	4.11	3.77	9.95	20.98
		Our PSU$_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	1.44	21.78	0.94	24.58	1.88	38.30	3.42	183.08	14.15
		Our PSU$_{\mathsf{op}}^{\mathsf{ps}}$	19.09	1.52	21.43	0.95	24.41	2.13	38.66	3.13	182.88	14.53
		Our PSU$_{\mathsf{pk}}$	19.08	1.70	59.15	2.24	61.30	3.07	76.01	4.35	219.35	17.46
	2^{10}	TCLZ-pub [TCLZ23]	0.76	1.58	2.80	1.17	3.43	1.94	3.69	3.17	9.70	14.97
		TCLZ-sym [TCLZ23]	0.77	2.33	2.95	1.11	3.70	2.31	4.33	3.88	9.99	21.36
		Our PSU$_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	5.71	22.67	1.28	25.34	3.37	39.50	6.99	184.55	49.83
		Our PSU$_{\mathsf{op}}^{\mathsf{ps}}$	19.09	6.07	22.17	1.16	25.39	3.17	39.45	7.43	184.40	54.09
		Our PSU$_{\mathsf{pk}}$	19.08	3.90	124.79	2.96	128.02	4.21	141.98	7.05	286.85	36.41

Table: Performance comparison. The best result is marked in green. The second best result is marked in blue.

Performance

Param. n	Param. m	Protocol	Comm. (MB) Setup	Comm. (MB) Online	LAN Setup	LAN Online	100Mbps Setup	100Mbps Online	10Mbps Setup	10Mbps Online	1Mbps Setup	1Mbps Online
2^{20}	2^4	TCLZ-pub [TCLZ23]	0.76	1.86	14.47	1.68	15.27	2.55	15.62	3.90	20.76	18.51
		TCLZ-sym [TCLZ23]	0.77	3.52	14.61	1.63	15.26	3.04	15.81	5.40	21.92	31.90
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	0.76	42.34	3.07	45.15	4.17	58.63	4.76	205.86	10.83
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ps}}$	19.09	0.77	42.09	3.10	45.77	4.59	59.81	5.26	203.93	11.92
		Our $\mathsf{PSU}_{\mathsf{pk}}$	19.08	0.88	89.39	8.84	94.72	9.40	108.37	10.71	252.10	16.82
	2^6	TCLZ-pub [TCLZ23]	0.76	1.87	14.20	1.61	14.48	2.44	15.16	3.78	21.33	17.95
		TCLZ-sym [TCLZ23]	0.77	3.52	15.23	1.55	15.42	3.63	15.64	6.19	21.37	32.77
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	1.45	42.53	1.39	46.21	2.24	58.51	3.34	203.39	15.14
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ps}}$	19.09	1.48	42.63	1.34	58.96	2.74	73.43	4.77	216.41	27.73
		Our $\mathsf{PSU}_{\mathsf{pk}}$	19.08	0.89	135.13	8.08	135.17	8.27	146.61	8.94	293.32	15.65
	2^8	TCLZ-pub [TCLZ23]	0.76	1.88	14.66	1.59	15.14	2.41	15.05	3.81	20.70	18.00
		TCLZ-sym [TCLZ23]	0.77	3.53	14.13	1.54	15.13	2.90	15.38	5.41	21.61	32.00
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	2.86	54.80	1.29	58.14	2.59	73.32	4.56	216.79	26.45
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ps}}$	19.09	2.94	55.18	1.35	58.96	2.74	73.43	4.77	216.41	27.73
		Our $\mathsf{PSU}_{\mathsf{pk}}$	19.08	2.97	138.53	3.19	136.75	3.93	151.06	6.16	299.52	29.09
	2^{10}	TCLZ-pub [TCLZ23]	0.76	1.92	14.42	1.60	15.22	2.45	15.15	3.86	21.41	18.43
		TCLZ-sym [TCLZ23]	0.77	3.58	14.26	1.55	15.21	2.97	16.02	5.48	21.48	32.55
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ddh}}$	19.08	5.76	85.49	1.93	89.58	3.71	102.54	7.90	245.92	50.95
		Our $\mathsf{PSU}_{\mathsf{op}}^{\mathsf{ps}}$	19.09	6.15	85.76	1.92	91.20	3.82	101.64	7.96	247.21	54.96
		Our $\mathsf{PSU}_{\mathsf{pk}}$	19.08	6.80	232.43	3.65	234.26	5.37	247.47	10.14	431.10	61.93

Table: Performance comparison. The best result is marked in green. The second best result is marked in blue.

Performance

Param.		Protocol	Comm. (MB)		Running time (s)							
					LAN		100Mbps		10Mbps		1Mbps	
n	m		Setup	Online	Setup	Online	Setup	Online	Setup	Online	Setup	Online
2^{22}	2^4	TCLZ-pub [TCLZ23]	0.76	3.59	60.60	3.21	60.69	4.51	60.64	6.89	67.03	34.86
		TCLZ-sym [TCLZ23]	0.77	11.13	61.39	3.02	61.74	5.28	63.20	13.31	67.48	97.47
		Our $\text{PSU}_{\text{op}}^{\text{ddh}}$	19.08	0.94	182.08	7.40	187.18	9.94	202.34	10.08	345.77	17.38
		Our $\text{PSU}_{\text{op}}^{\text{ps}}$	19.09	0.96	185.43	7.55	190.21	9.34	203.74	10.60	349.34	18.19
		Our PSU_{pk}	19.08	1.46	302.23	13.27	305.25	14.03	319.06	15.46	465.35	26.58
	2^6	TCLZ-pub [TCLZ23]	0.76	3.59	62.04	3.05	62.10	4.24	61.96	6.75	67.95	34.47
		TCLZ-sym [TCLZ23]	0.77	11.13	61.90	2.98	61.98	5.96	60.80	14.35	67.73	98.07
		Our $\text{PSU}_{\text{op}}^{\text{ddh}}$	19.08	1.64	221.34	6.03	223.13	7.01	229.80	9.05	373.27	20.91
		Our $\text{PSU}_{\text{op}}^{\text{ps}}$	19.09	1.66	216.53	6.61	217.72	7.32	235.72	8.51	380.17	21.50
		Our PSU_{pk}	19.08	2.16	424.83	11.25	417.94	11.98	442.52	13.69	584.50	29.81
	2^8	TCLZ-pub [TCLZ23]	0.76	3.60	60.62	3.26	61.03	4.39	61.71	6.89	66.99	34.31
		TCLZ-sym [TCLZ23]	0.77	11.15	62.10	3.11	61.25	5.24	62.72	13.42	68.35	98.23
		Our $\text{PSU}_{\text{op}}^{\text{ddh}}$	19.08	5.80	181.34	4.07	181.20	5.79	193.74	9.17	342.64	52.94
		Our $\text{PSU}_{\text{op}}^{\text{ps}}$	19.09	5.88	184.88	3.93	188.85	6.20	196.29	9.55	340.90	54.68
		Our PSU_{pk}	19.08	3.55	539.48	10.01	548.18	11.53	556.73	13.76	716.74	40.50
	2^{10}	TCLZ-pub [TCLZ23]	0.76	3.65	60.36	3.07	61.46	4.82	63.69	7.13	67.58	34.65
		TCLZ-sym [TCLZ23]	0.77	11.19	59.93	3.08	63.29	5.36	63.01	13.36	66.54	97.89
		Our $\text{PSU}_{\text{op}}^{\text{ddh}}$	19.08	10.76	234.33	4.55	238.26	6.51	261.40	14.22	405.45	96.39
		Our $\text{PSU}_{\text{op}}^{\text{ps}}$	19.09	11.15	236.02	4.56	239.26	6.89	254.70	14.93	409.62	99.66
		Our PSU_{pk}	19.08	11.19	551.38	9.28	554.46	13.23	567.37	18.95	706.52	103.62

Table: Performance comparison. The best result is marked in green. The second best result is marked in blue.

Thanks for Your Attention!

Any Questions?

code: `http://github.com/alibaba-edu/mpc4j`

eprint: `https://eprint.iacr.org/2024/1340`

Reference

[BS05] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT 2005*, 2005.

[CZZ+24] Yu Chen, Min Zhang, Cong Zhang, Minglang Dong, and Weiran Liu. Private set operations from multi-query reverse private membership test. In *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography,*, Lecture Notes in Computer Science, 2024.

[GMR+21] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In *PKC 2021*, 2021.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *CRYPTO 2021*, 2021.

[HLS+16] Kyle Hogan, Noah Luther, Nabil Schear, Emily Shen, David Stott, Sophia Yakoubov, and Arkady Yerukhimovich. Secure multiparty computation for cooperative cyber risk assessment. In *SecDev 2016*, 2016.

[JSZ+22] Yanxue Jia, Shi-Feng Sun, Hong-Sheng Zhou, Jiajun Du, and Dawu Gu. Shuffle-based private set union: Faster and more secure. In *USENIX Security 22*, 2022.

[KRTW19] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. Scalable private set union from symmetric-key techniques. In *ASIACRYPT*, 2019.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *CRYPTO 2005*, 2005.

[LG23] Xiang Liu and Ying Gao. Scalable multi-party private set union from multi-query secret-shared private membership test. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 237–271. Springer, 2023.

[LV04] Arjen K. Lenstra and Tim Voss. Information security risk assessment, aggregation, and mitigation. In *ACISP 2004*, 2004.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse OT extension. In *CRYPTO 2019*, 2019.

[RR22] Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 2505–2517. ACM, 2022.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In *EUROCRYPT 2021*, 2021.

[TCLZ23] Binbin Tu, Yu Chen, Qi Liu, and Cong Zhang. Fast unbalanced private set union from fully homomorphic encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 2959–2973. ACM, 2023.

[ZCL+23] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. Linear private set union from Multi-Query reverse private membership test. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 337–354, Anaheim, CA, August 2023. USENIX Association.