Scalable Multi-Party Private Set Union from Multi-Query Secret-Shared Private Membership Test

Xiang Liu[1], Ying Gao[1,2]

School of Cyber Security and Technology, Beihang University
Zhongguancun Laboratory

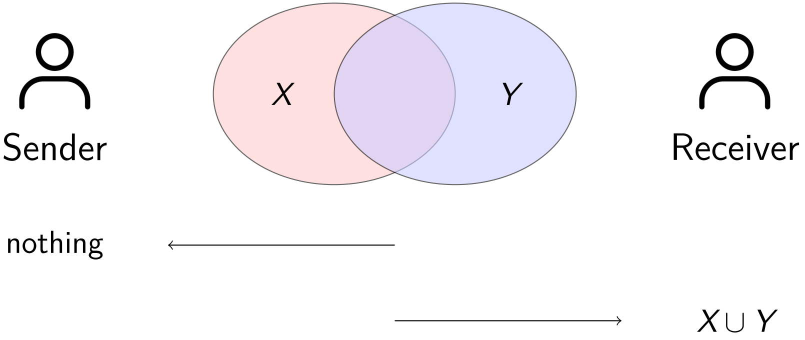
December 8th

Asiacrypt 2023

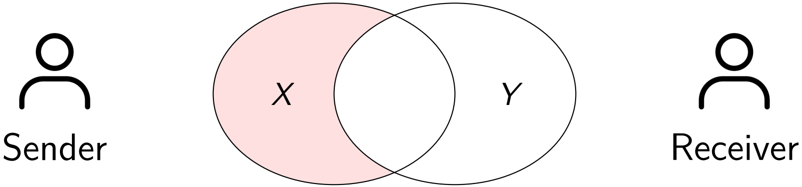
Contents

Contents

Private Set Union (PSU)

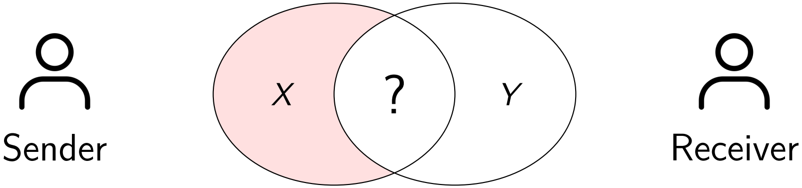


Private Set Union (PSU)



can compute $(X \cup Y) \setminus Y = X \setminus Y$

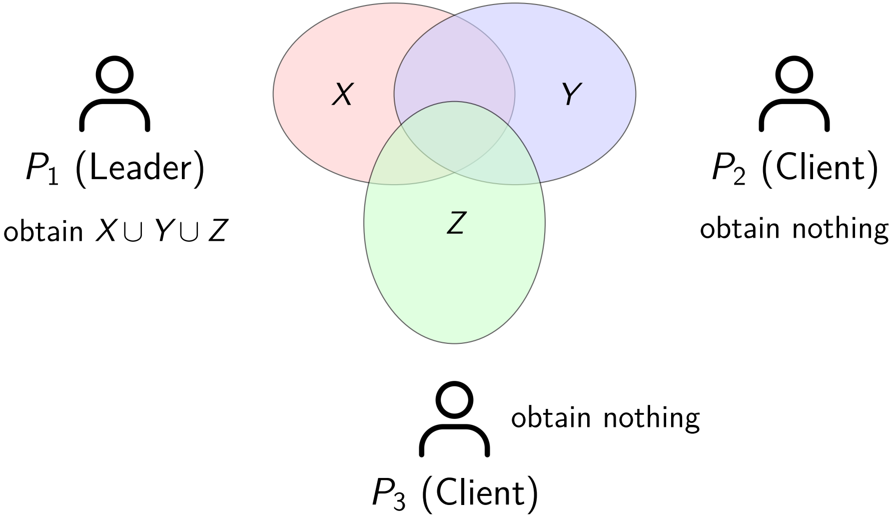
Private Set Union (PSU)



can compute $(X \cup Y) \setminus Y = X \setminus Y$

but knows nothing about $X \cap Y$

Multi-Party Private Set Union (MPSU)



Applications

- Cyber risk assessment and management via joint IP blacklists and joint vulnerability data [HLS+16; LV04]
- Privacy-preserving data aggregation [BSMD10]
- Building block for private database full join [KRTW19]
- Building block for private ID [GMR+21; ZLDL23]
- ...

Previous Work and Motivation

1. Additively homomorphic encryption (AHE) based constructions [KS05; Fri07; GHJ22]
   - resist arbitrary collusion
   - need a non-constant number of AHE operations, high computation cost
   - lack of implementation, can't estimate their performances
2. Other constructions
   - secure in the honest majority setting [SCK12; BA16]
   - [SCK12] has high computation and communication complexity
   - [BA16; VCE22] are only practical on small sets

Previous Work and Motivation

1. Additively homomorphic encryption (AHE) based constructions [KS05; Fri07; GHJ22]
   - resist arbitrary collusion
   - need a non-constant number of AHE operations, high computation cost
   - lack of implementation, can't estimate their performances
2. Other constructions
   - secure in the honest majority setting [SCK12; BA16]
   - [SCK12] has high computation and communication complexity
   - [BA16; VCE22] are only practical on small sets

Can we construct a truly scalable MPSU protocol?

Our Contributions

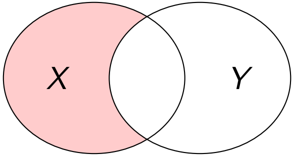
We focus on semi-honest setting, and assume that the adversary doesn't corrupt the leader and clients simultaneously.

- Introduce a new primitive called multi-query secret-shared private membership test (mq-ssPMT)
- Propose a new MPSU framework based on mq-ssPMT and secret-shared shuffle
- Our framework of MPSU can be slightly modified to compute multi-party private set intersection (MPSI), and the cardinality of the intersection and union (MPSI-CA, MPSU-CA)
- Demonstrate the scalability of our MPSU protocol with an implementation

Contents

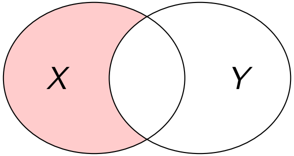
Two-Party PSU Framework

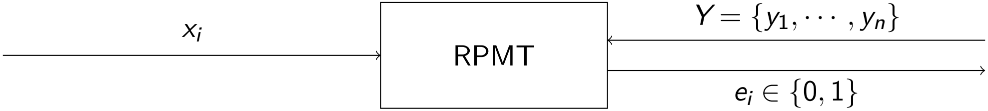
- Convert the union to the difference $X \cup Y = (X \setminus Y) \cup Y$
- $X \setminus Y$ can be computed efficiently by a combination of reverse private membership test (RPMT) and oblivious transfer (OT) [KRTW19]



Two-Party PSU Framework

- Convert the union to the difference $X \cup Y = (X \setminus Y) \cup Y$
- $X \setminus Y$ can be computed efficiently by a combination of reverse private membership test (RPMT) and oblivious transfer (OT) [KRTW19]

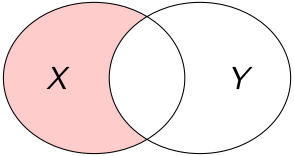


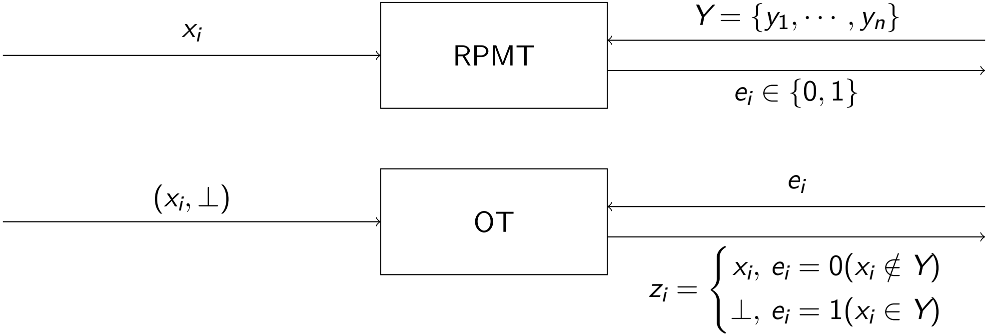


$$e_i = \begin{cases} 1, & x_i \in Y \\ 0, & x_i \notin Y \end{cases}$$

Two-Party PSU Framework

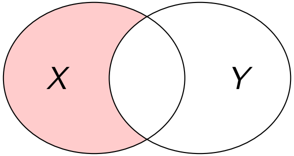
- Convert the union to the difference $X \cup Y = (X \setminus Y) \cup Y$
- $X \setminus Y$ can be computed efficiently by a combination of reverse private membership test (RPMT) and oblivious transfer (OT) [KRTW19]

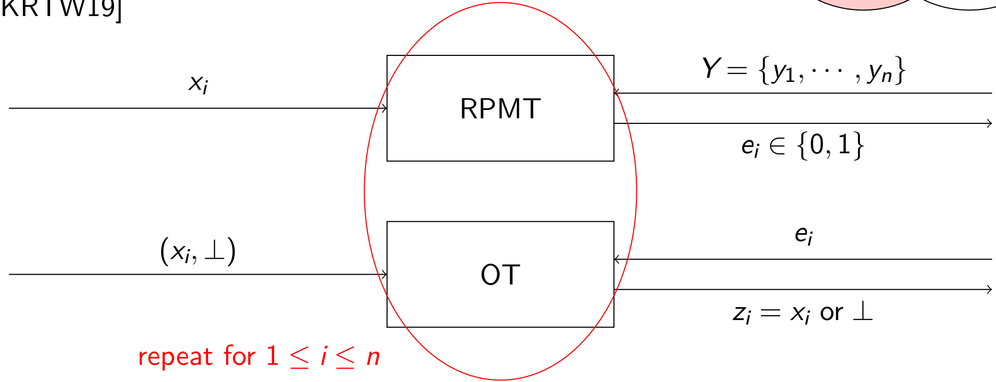




Two-Party PSU Framework

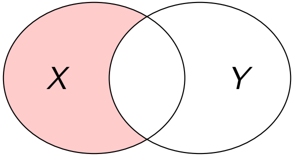
- Convert the union to the difference $X \cup Y = (X \setminus Y) \cup Y$
- $X \setminus Y$ can be computed efficiently by a combination of reverse private membership test (RPMT) and oblivious transfer (OT) [KRTW19]

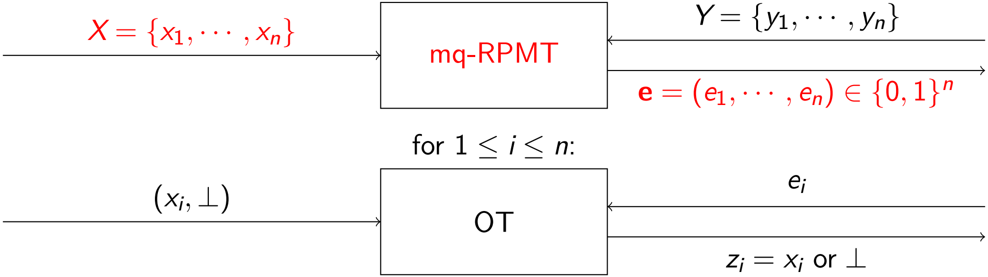




Two-Party PSU Framework

- Convert the union to the difference $X \cup Y = (X \setminus Y) \cup Y$
- $X \setminus Y$ can be computed efficiently by a combination of reverse private membership test (RPMT) and oblivious transfer (OT) [KRTW19]

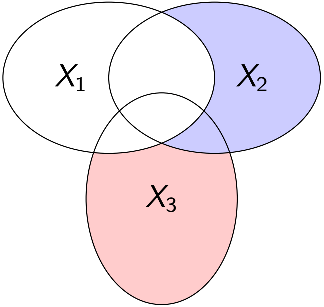




- multi-query RPMT (mq-RPMT) - query multiple times in an RPMT instance [ZCL+23]

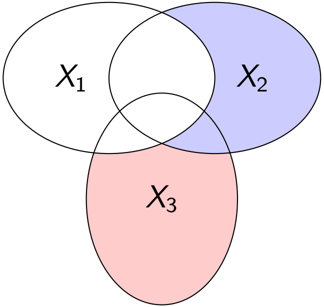
Our MPSU Main Idea

- Convert the union to the difference
  $X_1 \cup X_2 \cup X_3 = X_1 \cup (X_2 \setminus X_1) \cup (X_3 \setminus (X_2 \cup X_1))$
- Compute the differences separately and then merge them



Our MPSU Main Idea

- Convert the union to the difference
  $X_1 \cup X_2 \cup X_3 = X_1 \cup (X_2 \setminus X_1) \cup (X_3 \setminus (X_2 \cup X_1))$
- Compute the differences separately and then merge them

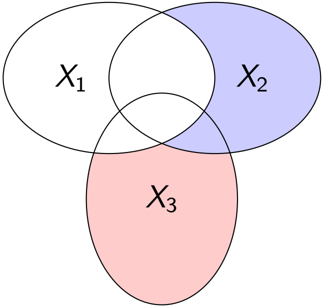


Two problems arise:

- How to compute the difference of more than two sets, such as $X_3 \setminus (X_2 \cup X_1)$?
- The difference sets should not be revealed. How to merge them securely?

Our MPSU Main Idea

- Convert the union to the difference
  $X_1 \cup X_2 \cup X_3 = X_1 \cup (X_2 \setminus X_1) \cup (X_3 \setminus (X_2 \cup X_1))$
- Compute the differences separately and then merge them

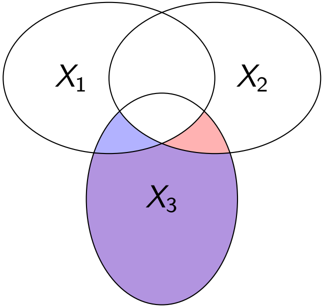


Two problems arise:

- How to compute the difference of more than two sets, such as $X_3 \setminus (X_2 \cup X_1)$?
- The difference sets should not be revealed. How to merge them securely?

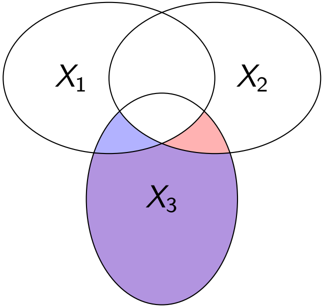
Compute $X_3 \setminus (X_2 \cup X_1)$

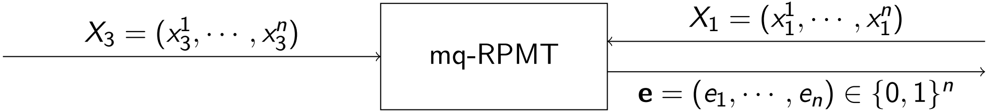
- Convert the difference of multi sets to the intersection of two differences $X_3 \setminus (X_2 \cup X_1) = (X_3 \setminus X_2) \cap (X_3 \setminus X_1)$
- Compute the differences separately, and then compute the intersection



Compute $X_3 \setminus (X_2 \cup X_1)$

- Convert the difference of multi sets to the intersection of two differences $X_3 \setminus (X_2 \cup X_1) = (X_3 \setminus X_2) \cap (X_3 \setminus X_1)$
- Compute the differences separately, and then compute the intersection
- If we use mq-RPMT, it will reveal $|X_3 \setminus X_1|$ and $|X_3 \setminus X_2|$

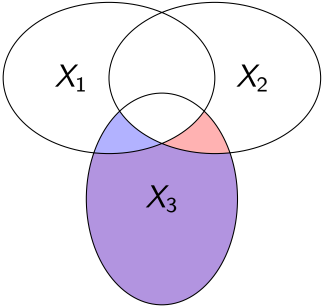




$|X_3 \setminus X_1| =$ hamming weight of $\mathbf{e}$

Compute $X_3 \setminus (X_2 \cup X_1)$

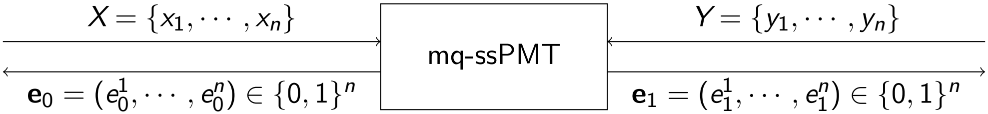
- Convert the difference of multi sets to the intersection of two differences $X_3 \setminus (X_2 \cup X_1) = (X_3 \setminus X_2) \cap (X_3 \setminus X_1)$
- Compute the differences separately, and then compute the intersection



- If we use mq-RPMT, it will reveal $|X_3 \setminus X_1|$ and $|X_3 \setminus X_2|$
- So we need to protect the output of mq-RPMT, meanwhile keep its ability to compute the difference

Multi-Query Secret-Shared Private Membership Test (mq-ssPMT)

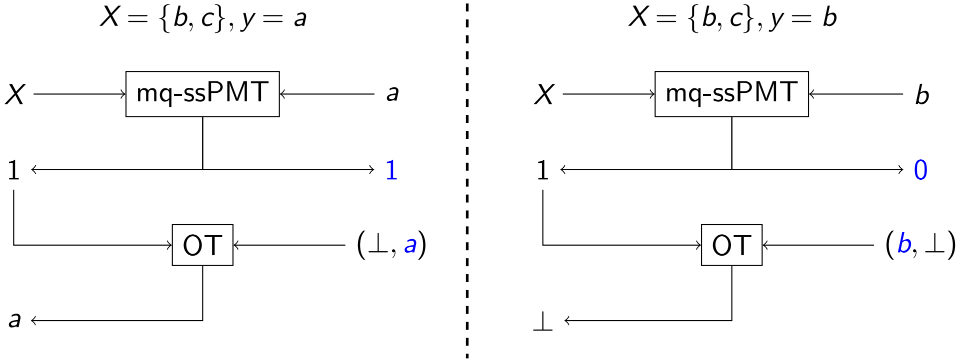
- If the output of mq-RPMT is shared to two parties, we get multi-query secret-shared private membership test (mq-ssPMT)



$$1 \le i \le n : e_0^i \oplus e_1^i = \begin{cases} 1, & y_i \in X \\ 0, & y_i \notin X \end{cases}$$

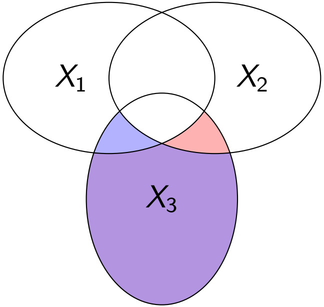
Multi-Query Secret-Shared Private Membership Test (mq-ssPMT)

- Similar to mq-RPMT, we can combine mq-ssPMT and OT to compute the difference
- And mq-ssPMT doesn't reveal any information



Compute $X_3 \setminus (X_2 \cup X_1)$

- Convert the difference of multi sets to the intersection of two differences $X_3 \setminus (X_2 \cup X_1) = (X_3 \setminus X_2) \cap (X_3 \setminus X_1)$
- Compute the differences separately, and then compute the intersection



- If we use mq-RPMT, it will reveal $|X_3 \setminus X_1|$ and $|X_3 \setminus X_2|$
- So we need to protect the output of mq-RPMT, meanwhile keep its ability to compute the difference

- Now we have mq-ssPMT, but we can't directly compute $X_3 \setminus X_2$ and $X_3 \setminus X_1$
- And how to compute the intersection without using an MPSI protocol?

Compute $X_3 \setminus (X_2 \cup X_1)$

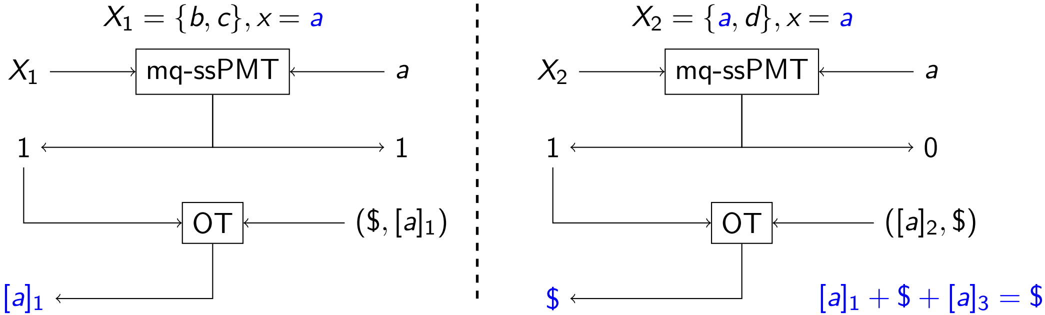
Our approach:

- Use a (3, 3) addtive secret sharing to share element $x = [x]_1 + [x]_2 + [x]_3$
- Use the share $[x]_i$ as the message of OT with P_i

Compute $X_3 \setminus (X_2 \cup X_1)$

Our approach:

- Use a (3, 3) addtive secret sharing to share element $x = [x]_1 + [x]_2 + [x]_3$
- Use the share $[x]_i$ as the message of OT with P_i

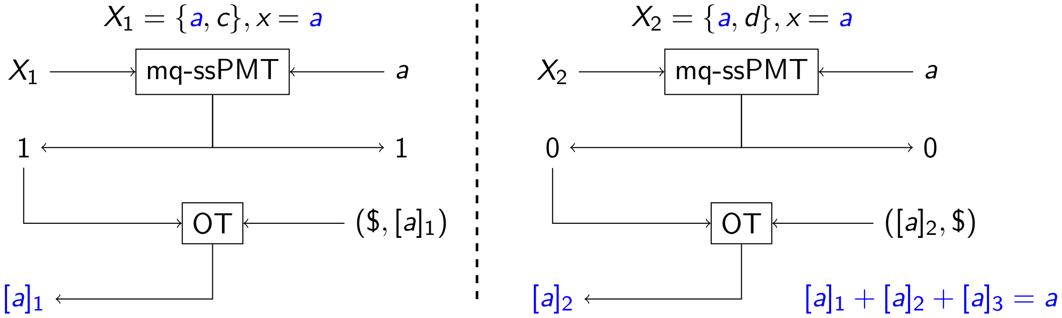


- If $x \in X_1$ or $x \in X_2$, the reconstruction of the secret will be random

Compute $X_3 \setminus (X_2 \cup X_1)$

Our approach:

- Use a (3, 3) addtive secret sharing to share element $x = [x]_1 + [x]_2 + [x]_3$
- Use the share $[x]_i$ as the message of OT with P_i

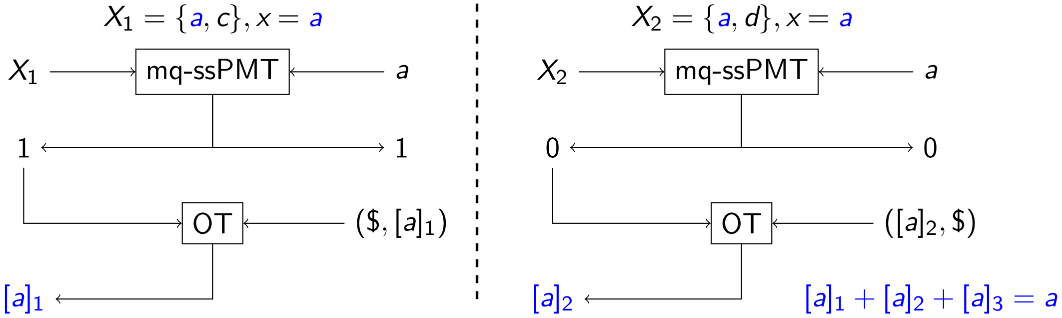


- If $x \notin X_1$ and $x \notin X_2$, the reconstruction of the secret will be x

Compute $X_3 \setminus (X_2 \cup X_1)$

Our approach:

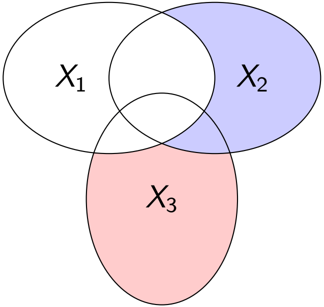
- Use a (3, 3) addtive secret sharing to share element $x = [x]_1 + [x]_2 + [x]_3$
- Use the share $[x]_i$ as the message of OT with P_i



- can reconstruct $x \Leftrightarrow x \notin X_1$ and $x \notin X_2 \Leftrightarrow x \in X_3 \setminus (X_2 \cup X_1)$

Our MPSU Main Idea

- Convert the union to the difference
  $X_1 \cup X_2 \cup X_3 = X_1 \cup (X_2 \setminus X_1) \cup (X_3 \setminus (X_2 \cup X_1))$
- Compute the differences separately and then merge them



Two problems arise:

- How to compute the difference of more than two sets, such as $X_3 \setminus (X_2 \cup X_1)$? ✓
- The difference sets should not be revealed. How to merge them securely?

Shuffle and Reshare

- Directly sending the share of $X_2 \setminus X_1$ and $X_3 \setminus (X_2 \cup X_1)$ to P_1 is not secure
- We should destroy the linkages of the difference set and the shares, but how?

Shuffle and Reshare

- Directly sending the share of $X_2 \setminus X_1$ and $X_3 \setminus (X_2 \cup X_1)$ to P_1 is not secure
- We should destroy the linkages of the difference set and the shares, but how?
- Use a multi-party secret-shared shuffle protocol [EB22]

Shuffle and Reshare

- Directly sending the share of $X_2 \setminus X_1$ and $X_3 \setminus (X_2 \cup X_1)$ to P_1 is not secure
- We should destroy the linkages of the difference set and the shares, but how?
- Use a multi-party secret-shared shuffle protocol [EB22]

$$X_1 = \{a, b\}, X_2 = \{a, c\}, X_3 = \{e, f\}$$

before shuffling

		P_1	P_2	P_3
$X_2 \setminus X_1$	r	\$	$[a]_2$	0
	c	$[c]_1$	$[c]_2$	0
$X_3 \setminus (X_2 \cup X_1)$	e	$[e]_1$	$[e]_2$	$[e]_3$
	f	$[f]_1$	$[f]_2$	$[f]_3$

Shuffle and Reshare

- Directly sending the share of $X_2 \setminus X_1$ and $X_3 \setminus (X_2 \cup X_1)$ to P_1 is not secure
- We should destroy the linkages of the difference set and the shares, but how?
- Use a multi-party secret-shared shuffle protocol [EB22]

$$X_1 = \{a, b\}, X_2 = \{a, c\}, X_3 = \{e, f\}$$

before shuffling

		P_1	P_2	P_3
$X_2 \setminus X_1$	r	\$	$[a]_2$	0
	c	$[c]_1$	$[c]_2$	0
$X_3 \setminus (X_2 \cup X_1)$	e	$[e]_1$	$[e]_2$	$[e]_3$
	f	$[f]_1$	$[f]_2$	$[f]_3$

$$\xrightarrow[\text{secret-shared shuffle}]{\text{multi-party}}$$

after shuffling

P_1	P_2	P_3	
$[e]'_1$	$[e]'_2$	$[e]'_3$	e
$[r]'_1$	$[r]'_2$	$[e]'_3$	r
$[f]'_1$	$[f]'_2$	$[f]'_3$	f
$[c]'_1$	$[c]'_2$	$[c]'_3$	c

Shuffle and Reshare

- Directly sending the share of $X_2 \setminus X_1$ and $X_3 \setminus (X_2 \cup X_1)$ to P_1 is not secure
- We should destroy the linkages of the difference set and the shares, but how?
- Use a multi-party secret-shared shuffle protocol [EB22]
- After shuffling, P_1 collects all the shares and outputs the union

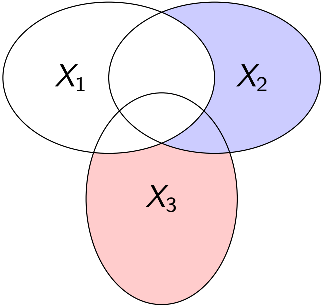
before shuffling

$$\begin{array}{ll|ccc|}
 & & P_1 & P_2 & P_3 \\
\hline
\multirow{2}{*}{$X_2 \setminus X_1$} & r & \$ & [a]_2 & 0 \\
 & c & [c]_1 & [c]_2 & 0 \\
\hline
\multirow{2}{*}{$X_3 \setminus (X_2 \cup X_1)$} & e & [e]_1 & [e]_2 & [e]_3 \\
 & f & [f]_1 & [f]_2 & [f]_3 \\
\hline
\end{array}
\xrightarrow[\text{secret-shared shuffle}]{\text{multi-party}}
\begin{array}{cccl}
P_1 & P_2 & P_3 & \\
[e]'_1 & [e]'_2 & [e]'_3 & e \\
[r]'_1 & [r]'_2 & [e]'_3 & r \\
[f]'_1 & [f]'_2 & [f]'_3 & f \\
[c]'_1 & [c]'_2 & [c]'_3 & c
\end{array}$$

after shuffling

Our MPSU Main Idea

- Convert the union to the difference
  $X_1 \cup X_2 \cup X_3 = X_1 \cup (X_2 \setminus X_1) \cup (X_3 \setminus (X_2 \cup X_1))$
- Compute the differences separately and then merge them



Two problems arise:

- How to compute the difference of more than two sets, such as $X_3 \setminus (X_2 \cup X_1)$? ✓
- The difference sets should not be revealed. How to merge them securely? ✓

- This framework can be easily extended to the setting of any number of parties

Contents

Instantiation from mq-RPMT in [ZCL+23]

- [ZCL+23] proposed two constructions of mq-RPMT, one is PKE-based, the other is SKE-based

Instantiation from mq-RPMT in [ZCL+23]

- [ZCL+23] proposed two constructions of mq-RPMT, one is PKE-based, the other is SKE-based
- At the end of SKE-based mq-RPMT, the sender $\mathcal{S}$ and receiver $\mathcal{R}$ run a 2PC protocol (like GMW protocol). Then $\mathcal{S}$ sends his share to $\mathcal{R}$, and $\mathcal{R}$ reconstructs the output

$\cdots\cdots$ $\qquad\qquad$ $\cdots\cdots$

$\xrightarrow{\{s_1^*, \cdots, s_n^*\}}$ GMW $\xleftarrow{(k, s)}$

$\xleftarrow[\mathbf{e}_0 = (e_0^1, \cdots, e_0^n) \in \{0,1\}^n]{}$ $\qquad$ $\xrightarrow[\mathbf{e}_1 = (e_1^1, \cdots, e_1^n) \in \{0,1\}^n]{}$

$\xrightarrow{\mathbf{e}_0}$ $\qquad$ output $\mathbf{e}_0 \oplus \mathbf{e}_1$

Instantiation from mq-RPMT in [ZCL+23]

- [ZCL+23] proposed two constructions of mq-RPMT, one is PKE-based, the other is SKE-based
- At the end of SKE-based mq-RPMT, the sender $\mathcal{S}$ and receiver $\mathcal{R}$ run a 2PC protocol (like GMW protocol). Then $\mathcal{S}$ sends his share to $\mathcal{R}$, and $\mathcal{R}$ reconstructs the output
- If we omit the reconstruction phase, it's exactly an mq-ssPMT

$$\cdots\cdots \qquad\qquad \cdots\cdots$$

$$\xrightarrow{\{s_1^*, \cdots, s_n^*\}} \boxed{\text{GMW}} \xleftarrow{(k, s)}$$

$$\mathbf{e}_0 = (e_0^1, \cdots, e_0^n) \in \{0,1\}^n \longleftarrow \boxed{\text{GMW}} \longrightarrow \mathbf{e}_1 = (e_1^1, \cdots, e_1^n) \in \{0,1\}^n$$

output $\mathbf{e}_0$ output $\mathbf{e}_1$

Other Instantiations

- mq-ssPMT can be replaced by n instances of ssPMT [CO18; LPR+21; ZMS+21], which only queries one item in each instance. But it increases overhead significantly
- It can also be realized by circuit-PSI [PSTY19; RR22]. It can be seen as the simplest form of circuit-PSI
    - It means that we can construct a PSU protocol combining circuit-PSI and OT

Contents

Experiment Results on Small Sets

- Instantiate mq-ssPMT with the mq-RPMT in [ZCL+23], and omit all the offline costs

Table: The comparison of SOTA and our MPSU protocol in running time (s) in the LAN setting.

	Number Parties k	Protocol	Set Size n 2^4	2^6	2^8	2^{10}
Time	3	[VCE22]	0.56	1.71	4.84	15.36
		Ours	**0.10**	**0.10**	**0.11**	**0.14**
	4	[VCE22]	0.76	2.36	7.64	20.84
		Ours	**0.15**	**0.16**	**0.17**	**0.19**
	5	[VCE22]	1.08	3.50	10.73	26.43
		Ours	**0.22**	**0.22**	**0.23**	**0.24**
	7	[VCE22]	1.84	4.49	15.29	52.82
		Ours	**0.36**	**0.36**	**0.37**	**0.39**
	10	[VCE22]	3.15	9.12	29.65	75.58
		Ours	**0.58**	**0.62**	**0.63**	**0.68**
	Speedup		5×	12×	41×	109×

Table: The comparison of SOTA and our MPSU protocol in communication cost (MB).

	Number Parties k	Protocol	Set Size n 2^4	2^6	2^8	2^{10}
Comm.	3	[VCE22]	0.16	0.56	1.82	5.68
		Ours	**0.15**	**0.16**	**0.28**	**0.96**
	4	[VCE22]	0.25	0.84	2.74	8.52
		Ours	**0.22**	**0.24**	**0.45**	**1.54**
	5	[VCE22]	0.33	1.11	3.65	11.36
		Ours	**0.30**	**0.33**	**0.63**	**2.17**
	7	[VCE22]	0.49	1.67	5.47	17.03
		Ours	**0.45**	**0.52**	**1.04**	**3.63**
	10	[VCE22]	0.74	2.51	8.21	25.55
		Ours	**0.69**	**0.83**	**1.77**	**6.30**
	Speedup		-	3×	4×	4×

Experiment Results on Large Sets

Table: Running time (seconds) of our protocol in LAN and WAN settings. Each party holds n 64-bit elements. The output length of H is $\ell = 64$. Cells with - denotes trials that ran out of memory.

Setting	Number Parties k	Set Size n 2^{14}	2^{16}	2^{18}	2^{20}
LAN	3	0.55	1.79	7.04	29.02
	4	0.60	1.88	7.46	30.28
	5	0.67	2.01	7.92	34.10
	7	0.88	2.71	10.77	45.68
	10	1.41	4.89	19.90	-
WAN	3	3.36	6.64	15.38	51.81
	4	4.14	8.63	20.28	72.61
	5	5.53	10.56	29.35	111.06
	7	6.91	17.21	60.17	227.75
	10	11.08	33.89	127.71	-

References

[BA16] Marina Blanton and Everaldo Aguiar. "Private and oblivious set and multiset operations". In: *Int. J. Inf. Sec.* 15.5 (2016), pp. 493–518.

[BSMD10] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas A. Dimitropoulos. "SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics". In: *USENIX Security*. 2010.

[CO18] Michele Ciampi and Claudio Orlandi. "Combining Private Set-Intersection with Secure Two-Party Computation". In: *SCN*. 2018.

[EB22] Saba Eskandarian and Dan Boneh. "Clarion: Anonymous Communication from Multiparty Shuffling Protocols". In: *NDSS*. 2022.

[Fri07] Keith Frikken. "Privacy-Preserving Set Union". In: *ACNS*. 2007.

[GHJ22] Xuhui Gong, Qiang-Sheng Hua, and Hai Jin. "Nearly Optimal Protocols for Computing Multi-party Private Set Union". In: *IWQoS*. 2022.

[GMR+21] Gayathri Garimella et al. "Private Set Operations from Oblivious Switching". In: *PKC*. 2021.

[HLS+16] Kyle Hogan et al. "Secure Multiparty Computation for Cooperative Cyber Risk Assessment". In: *IEEE Cybersecurity Development*. 2016.

[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 Song. "Privacy-Preserving Set Operations". In: *CRYPTO*. 2005.

[LPR+21] Tancrède Lepoint et al. "Private Join and Compute from PIR with Default". In: *ASIACRYPT*. 2021.

[LV04] Arjen Lenstra and Tim Voss. "Information Security Risk Assessment, Aggregation, and Mitigation". In: *Information Security and Privacy*. 2004.

[PSTY19] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. "Efficient Circuit-Based PSI with Linear Communication". In: *EUROCRYPT*. 2019.

[RR22] Srinivasan Raghuraman and Peter Rindal. "Blazing Fast PSI from Improved OKVS and Subfield VOLE". In: *CCS*. 2022.

[SCK12] Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. "Constant-Round Multi-party Private Set Union Using Reversed Laurent Series". In: *PKC*. 2012.

[VCE22] Jelle Vos, Mauro Conti, and Zekeriya Erkin. *Fast Multi-party Private Set Operations in the Star Topology from Secure ANDs and ORs*. Cryptology ePrint Archive, Paper 2022/721. 2022.

[ZCL+23] Cong Zhang et al. "Linear Private Set Union from Multi-Query Reverse Private Membership Test". In: *USENIX Security*. 2023.

[ZLDL23] Cong Zhang, Weiran Liu, Bolin Ding, and Dongdai Lin. "Efficient Private Multiset ID Protocols". In: *ICICS*. 2023.

[ZMS+21] Shengnan Zhao et al. "Lightweight threshold private set intersection via oblivious transfer". In: *WASA*. 2021.

Thanks for your attention!