Efficient Private Multiset ID Protocols

Cong Zhang[1,2] Weiran Liu[3] Bolin Ding[3] Dongdai Lin[1,2]

SKLOIS,IIE,CAS

School of Cyber Security, UCAS

Alibaba Group

November 20, 2023

ICICS 2023

Outline

1. Background
2. Preliminaries
3. Deterministic-Value (Oblivious) Programmable PRF
4. Construction of PMID
5. Implementation

Outline

1. Background
2. Preliminaries
3. Deterministic-Value (Oblivious) Programmable PRF
4. Construction of PMID
5. Implementation

Private Computation on Database

Alice

user_id	user_name	age	sex
1	Tom	38	F
2	Jerry	27	M
3	Lucy	32	F

Table A

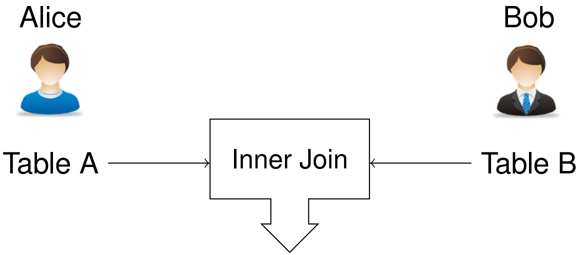
Bob

user_id	prod_id	prod_name	price
1	0003	Fish Toy	$3.49
3	0001	Teddy Bear	$11.99
4	0005	Raggedy Ann	$4.99
5	0006	Rabbit Toy	$3.49

Table B

Private Inner Join

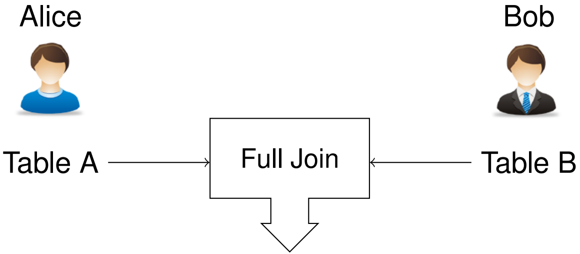
```
SELECT A.user_id,A.user_name B.prod_name
FROM A INNER JOIN B
ON A.user_id = B.user_id
```



user_id	user_name	prod_name
1	Tom	Fish Toy
3	Lucy	Teddy Bear

Private Full Join

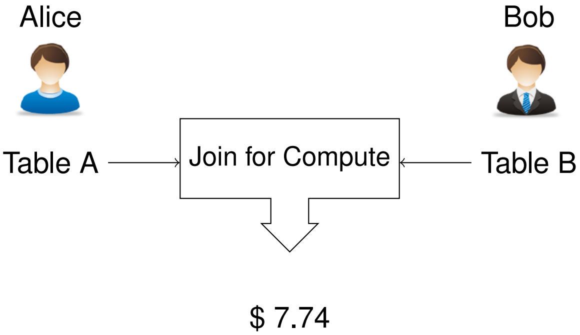
```
SELECT A.user_id,A.user_name B.prod_name
FROM A FULL OUTER JOIN B
ON A.user_id = B.user_id
```



user_id	user_name	prod_name
1	Tom	Fish Toy
2	Jerry	-
3	Lucy	Teddy Bear
4	-	Raggedy Ann
5	-	Rabbit Toy

Private Join for Compute

SELECT AVG(B.price)
FROM A INNER JOIN B
ON A.user_id = B.user_id
WHERE A.age > 30



Known Solutions

A natural idea for solving above problems is to use Private Set Operation (PSO) protocols:

- Private Set Intersection (PSI) [HFH99, FNP04, PSZ14, KKRT16, PRTY19, CM20] for Private Inner Join.
- Private Set Union (PSU) [KS05, Fri07, DC17, KRTW19, JSZ$^+$22, ZCL$^+$23] for Private Full Join.
- Private Set Intersection Cardinality/Sum (PSI-CA/PSI-Sum) [IKN$^+$17, IKN$^+$20] for Private Join for Computing *Linear Functions*.
- Circuit-based PSI/PSU (Circuit PSI/PSU) [HEKM11, HEK12, BA12] : [KS05, Fri07, DC17] for Private Join for Computing any desired functions.

Known Solutions

A natural idea for solving above problems is to use Private Set Operation (PSO) protocols:

- Private Set Intersection (PSI) [HFH99, FNP04, PSZ14, KKRT16, PRTY19, CM20] for Private Inner Join.
- Private Set Union (PSU) [KS05, Fri07, DC17, KRTW19, JSZ$^+$22, ZCL$^+$23] for Private Full Join.
- Private Set Intersection Cardinality/Sum (PSI-CA/PSI-Sum) [IKN$^+$17, IKN$^+$20] for Private Join for Computing *Linear Functions*.
- Circuit-based PSI/PSU (Circuit PSI/PSU) [HEKM11, HEK12, BA12] : [KS05, Fri07, DC17] for Private Join for Computing any desired functions.

☺ The Efficiency is high.

☹ Difficult to unify them in a variety of application scenarios.
Do not support multiset.

Private ID (PID)

Alice

Bob

Alice's set → PID ← Bob's set

$Y = \{b, c, e, f, h\}$ $X = \{a, c, d, f, g\}$

$$R^* = \{\text{id}_a, \text{id}_b, \text{id}_c, \text{id}_d, \text{id}_e, \text{id}_f, \text{id}_g, \text{id}_h\}$$

$$ID_Y = \{\text{id}_b, \text{id}_c, \text{id}_e, \text{id}_f, \text{id}_h\} \qquad ID_X = \{\text{id}_a, \text{id}_c, \text{id}_d, \text{id}_f, \text{id}_g\}$$

Private ID (PID)

Alice

Bob

Alice's set $\longrightarrow$ PID $\longleftarrow$ Bob's set

$Y = \{b, c, e, f, h\}$ $\qquad$ $X = \{a, c, d, f, g\}$

$$R^* = \{\text{id}_a, \text{id}_b, \text{id}_c, \text{id}_d, \text{id}_e, \text{id}_f, \text{id}_g, \text{id}_h\}$$

$$ID_Y = \{\text{id}_b, \text{id}_c, \text{id}_e, \text{id}_f, \text{id}_h\} \qquad ID_X = \{\text{id}_a, \text{id}_c, \text{id}_d, \text{id}_f, \text{id}_g\}$$

☺ Support a unified method to construct all PSO protocols.

☹ Do not support multiset.

Motivation

In most analytical workloads, such as the decision support benchmark TPC-DS [PSKL02], the majority of joins are *key-foreign key joins*.

Alice

user_id	user_name	age	sex
1	Tom	38	F
2	Jerry	27	M
3	Lucy	32	M

Table A

Bob

user_id	prod_id	prod_name	price
1	0003	Fish Toy	$3.49
1	0001	Teddy Bear	$11.99
4	0005	Raggedy Ann	$4.99
4	0006	Rabbit Toy	$3.49

Table B

Motivation

In most analytical workloads, such as the decision support benchmark TPC-DS [PSKL02], the majority of joins are *key-foreign key joins*.

Alice

user_id	user_name	age	sex
1	Tom	38	F
2	Jerry	27	M
3	Lucy	32	M

Table A

Bob

user_id	prod_id	prod_name	price
1	0003	Fish Toy	$3.49
1	0001	Teddy Bear	$11.99
4	0005	Raggedy Ann	$4.99
4	0006	Rabbit Toy	$3.49

Table B

Can we construct an efficient PID protocol in which the inputs of the parties are multiset?

Private Multiset ID (PMID)

Alice

Bob

Alice's multiset $\longrightarrow$ PMID $\longleftarrow$ Bob's multiset

$Y = \{b, c, c, e, e\}$ $\qquad$ $X = \{a, a, c, c, d\}$

$$R^* = \{\mathrm{id}_a^{(1)}, \mathrm{id}_a^{(2)}, \mathrm{id}_b, \mathrm{id}_c^{(1)}, \mathrm{id}_c^{(2)}, \mathrm{id}_c^{(3)}, \mathrm{id}_c^{(4)}, \mathrm{id}_d, \mathrm{id}_e^{(1)}, \mathrm{id}_e^{(2)}\}$$

$$ID_Y = \{\mathrm{id}_b, \mathrm{id}_c^{(1)}, \mathrm{id}_c^{(2)}, \mathrm{id}_c^{(3)}, \mathrm{id}_c^{(4)}, \mathrm{id}_e^{(1)}, \mathrm{id}_e^{(2)}\} \; ID_X = \{\mathrm{id}_a^{(1)}, \mathrm{id}_a^{(2)}, \mathrm{id}_c^{(1)}, \mathrm{id}_c^{(2)}, \mathrm{id}_c^{(3)}, \mathrm{id}_c^{(4)}, \mathrm{id}_d\}$$

Motivation

user_id	user_name	age	sex
1	Tom	38	F
2	Jerry	27	M
3	Lucy	32	F

Table A

PMID	user_id
uid_1_1	1
uid_1_2	1
uid_2_1	2
uid_3_1	3
uid_4_1	
uid_4_2	

PMID	user_id	user_name	age	sex
uid_1_1	1	Alice	38	F
uid_1_2	1	Alice	38	F
uid_2_1	2	Bob	27	M
uid_3_1	3	Carol	32	M
uid_4_1	null	null	null	null
uid_4_2	null	null	null	null

Table A with PMID as UID

$7.74

user_id	prod_id	prod_name	price
1	0003	Fish Toy	$3.49
3	0001	Teddy Bear	$11.99
4	0005	Raggedy Ann	$4.99
5	0006	Rabbit Toy	$3.49

Table B

PMID	user_id
uid_1_1	1
uid_1_2	1
uid_2_1	
uid_3_1	
uid_4_1	4
uid_4_2	4

PMID	user_id	prod_id	prod_name	price
uid_1_1	1	0003	Fish Toy	$3.49
uid_1_2	1	0001	Teddy Bear	$11.99
uid_2_1	null	null	null	null
uid_3_1	null	null	null	null
uid_4_1	4	0005	Raggedy Ann	$4.99
uid_4_2	4	0006	Rabbit Toy	$3.49

Table B with PMID as UID

Outline

1 Background

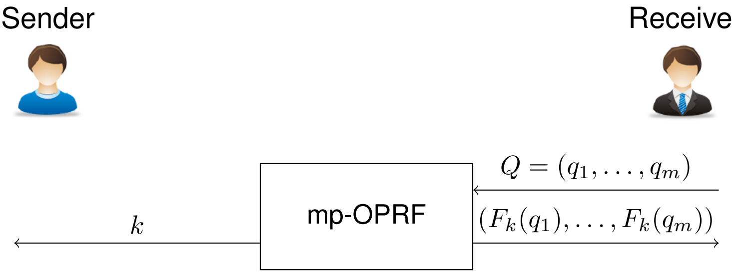
2 Preliminaries

3 Deterministic-Value (Oblivious) Programmable PRF

4 Construction of PMID

5 Implementation

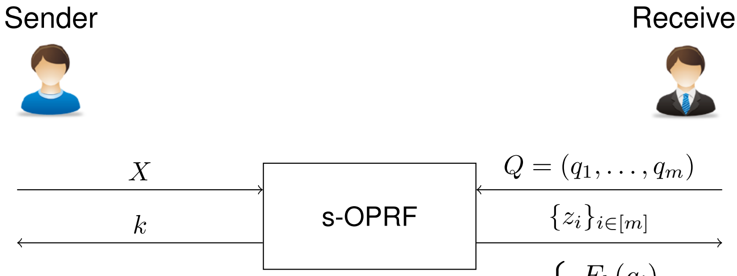
Multi-Point Oblivious PRF (mp-OPRF)



Learns nothing about Q.

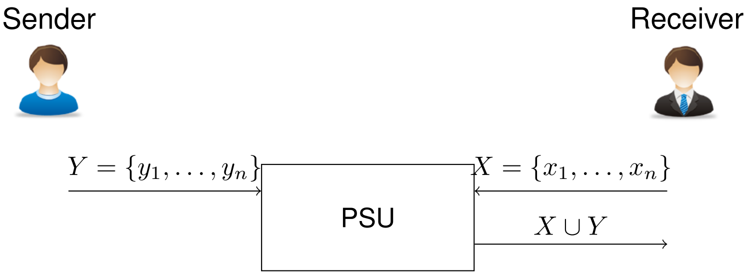
Learns nothing about k

Sloppy Oblivious PRF(s-OPRF)

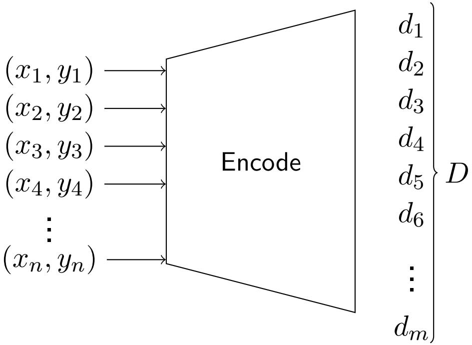


$$z_i = \begin{cases} F_k(q_i) & q_i \in X; \\ \text{random value} & q_i \notin X \end{cases}$$

Private Set Union (PSU)



Oblivious Key-Value Store



Rate:n/m
optimal is 1.

Encode Complexity:
complexity of Encode algorithm.

Decode Complexity:
complexity of Decode algorithm.

- $\text{Encode}((x_1, y_1), \ldots, (x_n, y_n)) \to D$
- $\text{Decode}(D, x) \to y$

Correctness. For all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys:

$$(x, y) \in A \text{ and } \perp \neq D \leftarrow \text{Encode}_H(A) \Longrightarrow \text{Decode}_H(D, x) = y$$

Oblivious Key-Value Store

Obliviousness. For all distinct $\{x_1^0, \ldots, x_n^0\}$ and $\{x_1^1, \ldots, x_n^1\}$, if Encode_H does not output $\perp$ for $\{x_1^0, \ldots, x_n^0\}$ or $\{x_1^1, \ldots, x_n^1\}$, the distribution of $\{D | y_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}_H((x_1^0, y_1), \ldots, (x_n^0, y_n))\}$ is computationally indistinguishable to the distribution of $\{D | y_i \leftarrow \mathcal{V}, i \in [n], \mathsf{Encode}_H((x_1^1, y_1), \ldots, (x_n^1, y_n))\}$.
A key-value store is an oblivious key-value store (OKVS) if it satisfies the obliviousness property.
In our application, we instead require OKVS to satisfy the following *partial obliviousness* property since our application will always leak some values.
Partial Obliviousness. For $t \in [n]$, and some fixed key-value pairs $\{(x_i, y_i)\}_{i \in [t]}$, for all distinct $\{x_{t+1}^0, \ldots, x_n^0\}$ and all distinct $\{x_{t+1}^1, \ldots, x_n^1\}$, if Encode_H does not output $\perp$, then the following distributions are computationally indistinguishable:

$$\{D | y_i \xleftarrow{R} \mathcal{V}, i \in [t+1, n], \mathsf{Encode}_H((x_1, y_1), \ldots, (x_t, y_t), (x_{t+1}^0, y_{t+1}), \ldots, (x_n^0, y_n))\}$$
$$\{D | y_i \xleftarrow{R} \mathcal{V}, i \in [t+1, n], \mathsf{Encode}_H((x_1, y_1), \ldots, (x_t, y_t), (x_{t+1}^1, y_{t+1}), \ldots, (x_n^1, y_n))\}$$

We note that when $t = 0$, this property is equal to the standard Obliviousness, and when $t = n$, the two distributions are identical.

Oblivious Key-Value Store

Table: A comparison between the different OKVS schemes.

scheme	rate	encoding	decoding
Interpolation polynomial	1	$O(n \log^2 n)$	$O(\log n)$
Garbled Bloom Filter[DCW13]	$O(1/\lambda)$	$O(\lambda n)$	$O(\lambda)$
Garbled Cuckoo Table [PRTY20]	0.4	$O(\lambda n)$	$O(\lambda)$
3H-GCT [GPR$^+$21]	0.81	$O(\lambda n)$	$O(\lambda)$
RR22 [RR22]	0.81	$O(\lambda n)$	$O(\lambda)$
RB-OKVS [BPSY23]	0.97	$O(\lambda n)$	$O(\lambda)$

n is the number of key-value pairs, λ is a statistical security parameter (e.g.,λ = 40).

Outline

1. Background
2. Preliminaries
3. Deterministic-Value (Oblivious) Programmable PRF
4. Construction of PMID
5. Implementation

Programmable PRF (PPRF)

Programmable PRF (PPRF) [KMP+17] is a special PRF with the additional property that on a certain "programmed" set of inputs the function outputs "programmed" values. A programmable PRF consists of the following algorithms:

- $\mathsf{KeyGen}(1^\kappa, \mathcal{P}) \to (k, \mathsf{hint})$: Given a security parameter and set of points $\mathcal{P} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ with distinct x_i-values, generates a PRF key k and (public) auxiliary information hint.
- $F(k, \mathsf{hint}, x) \to y$: Evaluates the PRF on input x, giving output y.

Correctness. A programmable PRF satisfies correctness if $(x, y) \in \mathcal{P}$, and $(k, \mathsf{hint}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathcal{P})$, then $F(k, \mathsf{hint}, x) = y$.

Security. For security, considering the following experiment:

$\mathsf{Exp}^{\mathcal{A}}(X, Q, \kappa)$:
for each $x_i \in X$, choose random $y_i \leftarrow \mathcal{V}$
$(k, \mathsf{hint}) \leftarrow \mathsf{KeyGen}(1^\kappa, \{(x_i, y_i) | x_i \in X\})$
return $\mathcal{A}(\mathsf{hint}, \{F(k, \mathsf{hint}, q) | q \in Q\})$

We say that a PPRF is (n, μ)-secure if for all $|X_0| = |X_1| = n$, all $|Q| = \mu$, and all PPT $\mathcal{A}$:

$$|Pr[\mathsf{Exp}^{\mathcal{A}}(X_0, Q, \kappa) = 1] - Pr[\mathsf{Exp}^{\mathcal{A}}(X_1, Q, \kappa) = 1]| \leq negl(\kappa)$$

Deterministic-Value Programmable PRF (dv-PPRF)

Deterministic-Value Pseudorandomness. For any fixed set of points $\mathcal{P} = \{(x_1, y_1), \dots, (x_t, y_t)\}$, considering the following experiment:

$\mathsf{Exp}^{\mathcal{A}}(\mathcal{P}, X, Q, \kappa)$:
for each $x_i \in X$, choose random $y_i \leftarrow \mathcal{V}$
$(k, \mathsf{hint}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathcal{P} \cup \{(x_i, y_i) | x_i \in X\})$
return $\mathcal{A}(\mathcal{P}, \mathsf{hint}, \{F(k, \mathsf{hint}, q) | q \in Q\})$

We say that a PPRF satisfying (t, n, μ)-deterministic-value pseudorandomness if for all $|X_0| = |X_1| = n - t$, all $|Q| = \mu$ satisfying $Q \cap \{x_1, \dots, x_t\} = \emptyset$ and all PPT $\mathcal{A}$:

$$|Pr[\mathsf{Exp}^{\mathcal{A}}(\mathcal{P}, X_0, Q, \kappa) = 1] - Pr[\mathsf{Exp}^{\mathcal{A}}(\mathcal{P}, X_1, Q, \kappa) = 1]| \leq negl(\kappa)$$

Definition (dv-PPRF)

A Deterministic-Value Programmable PRF (dv-PPRF) is the PPRF scheme satisfying correctness and (t, n, μ)-deterministic-value pseudorandomness.

Construction of dv-PPRF

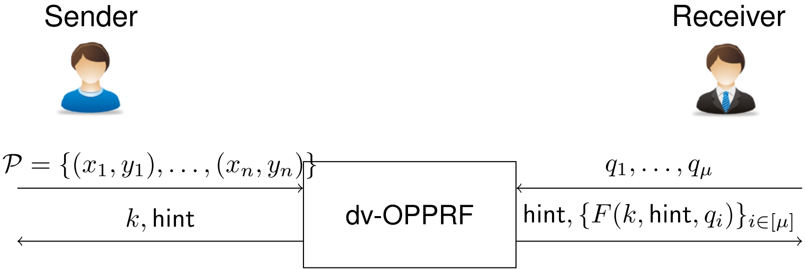
Let $\widehat{F}$ be a PRF and $(\mathsf{Encode}_H, \mathsf{Decode}_H)$ be an OKVS scheme satisfying partial obliviousness. We define it as follows:

- $\mathsf{KeyGen}(1^\kappa, \{(x_1, y_1), \ldots, (x_n, y_n)\})$: Choose a random key k for $\widehat{F}$. Compute an OKVS $D := \mathsf{Encode}_H((x_1, y_1 \oplus \widehat{F}_k(x_1)), \ldots, (x_n, y_n \oplus \widehat{F}_k(x_n)))$. Let hint be D.
- $F(k, \mathsf{hint}, q) = \widehat{F}_k(q) \oplus \mathsf{Decode}_H(\mathsf{hint}, q)$.

Theorem

Assuming the OKVS scheme satisfies partial obliviousness, the above construction is a dv-PPRF.

Deterministic-Value Oblivious Programmable PRF (dv-OPPRF)



Construction of dv-OPPRF

$S(\mathcal{P} = \{(x_1, y_1), \ldots, (x_n, y_n)\})$

$R(Q = \{q_1, \ldots, q_\mu\})$

mp-OPRF

$\xleftarrow{\quad k \quad}$ mp-OPRF $\xleftarrow{\quad Q \quad}$

$\xrightarrow{\{\bar{F}_k(q)\}_{q \in Q}}$

$P \leftarrow \mathsf{Encode}(\{(x_i, \bar{F}_k(x_i) \oplus y_i)\}_{i \in [n]})$

$\xrightarrow{\qquad P \qquad}$

$\mathsf{hint} := P$

Output (k, hint)

$\mathsf{hint} := P$

Output $\{F(k, \mathsf{hint}, q) := \mathsf{Decode}(P, q) \oplus \bar{F}_k(q)\}_{y \in Y}$

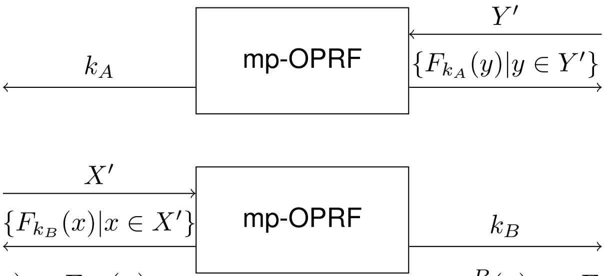
Outline

1. Background
2. Preliminaries
3. Deterministic-Value (Oblivious) Programmable PRF
4. Construction of PMID
5. Implementation

PMID from mp-OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$
$X' := \{x_1, \ldots, x_m\}$

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$
$Y' := \{y_1, \ldots, y_n\}$



$$r^A(x) := F_{k_A}(x) \oplus F_{k_B}(x)$$

$$c_i^{\mathsf{x}} = \begin{cases} \text{random value} & u_i^{\mathsf{x}} = 1; \\ u_i^{\mathsf{x}} & u_i^{\mathsf{x}} \neq 1 \end{cases}$$

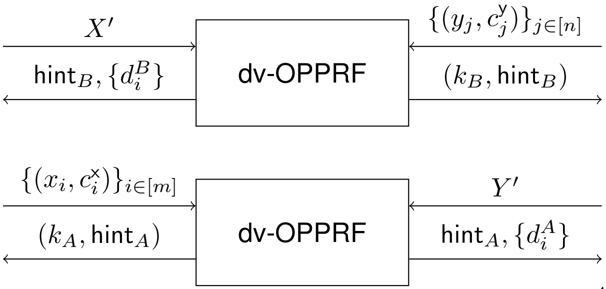
$$r^B(y) := F_{k_A}(y) \oplus F_{k_B}(y)$$

$$c_i^{\mathsf{y}} = \begin{cases} \text{random value} & u_i^{\mathsf{y}} = 1; \\ u_i^{\mathsf{y}} & u_i^{\mathsf{y}} \neq 1 \end{cases}$$

PMID from mp-OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$



$$\bar{u}_i^{\mathsf{x}} = \begin{cases} u_i^{\mathsf{x}} \cdot d_i^B & 1 < d_i^B \le U_y; \\ u_i^{\mathsf{x}} & d_i^B > U_y \end{cases}$$

$$\bar{u}_j^{\mathsf{y}} = \begin{cases} u_j^{\mathsf{y}} \cdot d_j^A & 1 < d_j^A \le U_x; \\ u_j^{\mathsf{y}} & d_j^A > U_x \end{cases}$$

PMID from mp-OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$

$i \in [m]$, $t \in [\bar{u}_i^{\mathsf{x}}]$:

$$\mathsf{id}(x_i^{(t)}) := \bar{H}(r^A(x_i)||t)$$

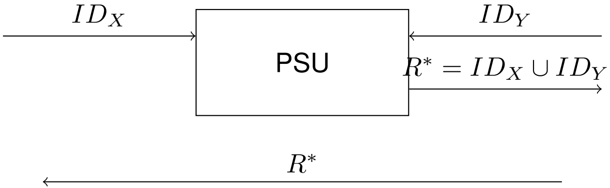
$$ID_X := \{\mathsf{id}(x_i^{(t)}) | i \in [m], t \in [\bar{u}_i^{\mathsf{x}}]\}$$

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$

$j \in [n]$, $t \in [\bar{u}_i^{\mathsf{y}}]$:

$$\mathsf{id}(y_j^{(t)}) := \bar{H}(r^B(y_j)||t)$$

$$ID_Y := \{\mathsf{id}(y_j^{(t)}) | j \in [n], t \in [\bar{u}_j^{\mathsf{y}}]\}$$



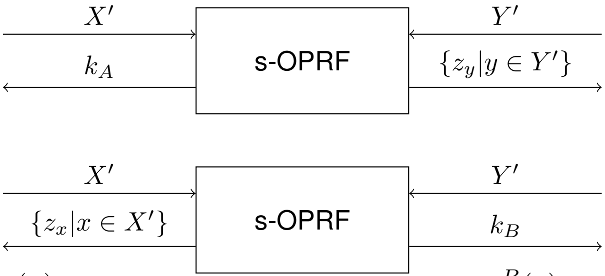
PMID from sloppy OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$

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

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$

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



$$r^A(x) := F_{k_A}(x) \oplus z_x$$

$$c_i^{\mathsf{y}} = \begin{cases} \text{random value} & u_i^{\mathsf{y}} = 1; \\ u_i^{\mathsf{y}} & u_i^{\mathsf{y}} \neq 1 \end{cases}$$

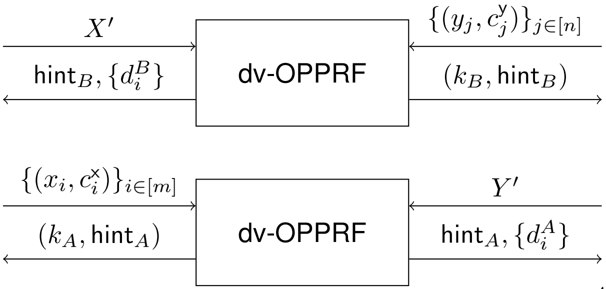
$$r^B(y) := z_y \oplus F_{k_B}(y)$$

$$c_i^{\mathsf{x}} = \begin{cases} \text{random value} & u_i^{\mathsf{x}} = 1; \\ u_i^{\mathsf{x}} & u_i^{\mathsf{x}} \neq 1 \end{cases}$$

PMID from sloppy OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$



$$\bar{u}_i^{\mathsf{x}} = \begin{cases} u_i^{\mathsf{x}} \cdot d_i^B & 1 < d_i^B \leq U_y; \\ u_i^{\mathsf{x}} & d_i^B > U_y \end{cases}$$

$$\bar{u}_j^{\mathsf{y}} = \begin{cases} u_j^{\mathsf{y}} \cdot d_j^A & 1 < d_j^A \leq U_x; \\ u_j^{\mathsf{y}} & d_j^A > U_x \end{cases}$$

PMID from sloppy OPRF

Alice $(X = \{(x_1, u_1^{\mathsf{x}}), \ldots, (x_m, u_m^{\mathsf{x}})\})$

$i \in [m]$, $t \in [\bar{u}_i^{\mathsf{x}}]$:

$\mathsf{id}(x_i^{(t)}) := \bar{H}(r^A(x_i)||t)$

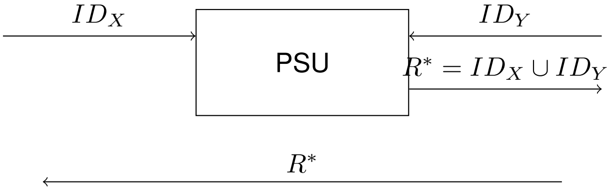
$ID_X := \{\mathsf{id}(x_i^{(t)}) | i \in [m], t \in [\bar{u}_i^{\mathsf{x}}]\}$

Bob $(Y = \{(y_1, u_1^{\mathsf{y}}), \ldots, (y_n, u_n^{\mathsf{y}})\})$

$j \in [n]$, $t \in [\bar{u}_i^{\mathsf{y}}]$:

$\mathsf{id}(y_j^{(t)}) := \bar{H}(r^B(y_j)||t)$

$ID_Y := \{\mathsf{id}(y_j^{(t)}) | j \in [n], t \in [\bar{u}_j^{\mathsf{y}}]\}$



Outline

1. Background
2. Preliminaries
3. Deterministic-Value (Oblivious) Programmable PRF
4. Construction of PMID
5. Implementation

PID Comparisons

Protocols	LAN(s)				WAN(s)				Comm(MB)			
	2^{14}	2^{16}	2^{18}	2^{20}	2^{14}	2^{16}	2^{18}	2^{20}	2^{14}	2^{16}	2^{18}	2^{20}
[BKM$^+$20]	4.33	17.4	69.67	277.56	5.07	19.42	75.56	298.05	3.35	13.41	53.63	214.5
Std-[GMR$^+$21]	1.86	9.03	4.77	217.51	4.85	17.43	76.96	327.49	16.45	70.51	302.3	1284.47
Sloppy-[GMR$^+$21]	1.75	7.82	35.49	162.71	6.02	17.87	73.79	306.53	20.89	87.9	384.28	1602.82
Std-PMID	2.05	9.54	47.56	221.43	5.64	18.41	78.05	326.63	16.45	70.51	302.3	1284.47
Sloppy-PMID	1.75	7.76	35.97	163.73	5.83	18.75	77.88	315.6	20.89	87.9	384.28	1602.82

Table: Communication (in MB) and run time (in seconds) of the private-ID protocol for input set sizes $n = 2^{14}, 2^{16}, 2^{18}, 2^{20}$ executed over a single thread for LAN and WAN configurations.

Scalability and Parallelizability

n	Protocol	Multi-plicity		Comm.(MB)			Running time (s)			
							LAN		WAN	
		U_x	U_y	Alice	Bob	Total	T=1	T=8	T=1	T=8
2^{14}	Sloppy-PMID	1	1	9.31	11.58	20.89	1.75	0.7	5.83	4.35
		1	3	15.82	22.73	38.55	3.47	1.53	9.13	7.35
		3	3	43.1	56.09	99.19	7.88	3.21	19.81	16.24
	Std-PMID	1	1	7.09	9.36	16.46	2.05	0.68	5.64	3.95
		1	3	13.6	20.51	34.11	3.82	1.48	9.23	6.84
		3	3	40.88	53.87	94.75	8.42	3.35	20	15.41
2^{16}	Sloppy-PMID	1	1	39.49	48.41	87.9	7.76	3.02	18.75	14.85
		1	3	68.36	95.44	163.8	15.58	6.66	35.04	26.32
		3	3	187.23	237.51	424.74	37.35	16.26	82.3	63.93
	Std-PMID	1	1	30.8	39.71	70.51	9.54	3.24	18.41	13.44
		1	3	59.67	86.75	146.42	17.73	7.03	34.8	24.04
		3	3	178.54	228.82	407.36	38.38	16.3	82.24	60.5
2^{18}	Sloppy-PMID	1	1	174.82	209.46	384.28	35.97	14.94	77.88	56.76
		1	3	299.02	405.66	704.68	72.33	32.88	144	107.13
		3	3	813.55	1010.59	1824.13	181.58	89.62	345.54	268.1
	Std-PMID	1	1	133.83	168.47	302.3	47.56	15.46	78.05	49.78
		1	3	258.03	364.67	622.7	84.51	32.96	147.63	101.62
		3	3	772.56	969.6	1742.15	195.43	92.1	350.43	261.19
2^{20}	Sloppy-PMID	1	1	733.61	869.21	1602.82	163.73	75.93	315.6	230.64
		1	3	1271.21	1690.33	2961.54	347.49	173.61	608.68	449.01
		3	3	-	-	-	-	-	-	-
	Std-PMID	1	1	574.44	710.03	1284.47	221.43	77.49	326.63	203.64
		1	3	1112.04	1531.16	2643.19	405.15	177.51	628.13	422.77
		3	3	-	-	-	-	-	-	-

Table: Running time (in seconds) of Sloppy-PMID and Std-PMID with set size ($n = m$), number of threads ($T \in \{1, 8\}$) and number of multiplicity ($U \in \{1, 3\}$) in WAN/LAN settings. Cells with "-" denote setting that program out of memory.

THANK YOU!

Q & A

Reference

[BA12] Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. In *ASIACCS 2012*, 2012.

[BKM+20] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private matching for compute. Cryptology ePrint Archive, 2020. `https://ia.cr/2020/599`.

[BPSY23] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Near-optimal oblivious key-value stores for efficient psi, psu and volume-hiding multi-maps. Cryptology ePrint Archive, Paper 2023/903, 2023. USENIX Security 2023.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In *CRYPTO 2020*, 2020.

[DC17] Alex Davidson and Carlos Cid. An efficient toolkit for computing private set operations. In *ACISP 2017*, 2017.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *CCS 2013*, 2013.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, 2004.

[Fri07] Keith B. Frikken. Privacy-preserving set union. In *ACNS 2007*, 2007.

[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.

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

[HEKM11] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security*, 2011.

[HFH99] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Electronic Commerce (EC-99)*, 1999.

[IKN+17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. *IACR Cryptol. ePrint Archive 2017/738*, 2017.

[IKN+20] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *EuroS&P 2020*, 2020.

[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 *31st USENIX Security Symposium (USENIX Security 22)*, pages 2947–2964, Boston, MA, August 2022. USENIX Association.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *CCS 2016*, 2016.

[KMP+17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *CCS 2017*, 2017.

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