Linear Private Set Union from Multi-Query Reverse Private Membership Test

Cong Zhang[1,2] Yu Chen[3] Weiran Liu[4] Min Zhang[3] Dongdai Lin[1,2]

SKLOIS,IIE,CAS

School of Cyber Security, UCAS

Shandong University

Alibaba Group

August 2023

USENIX Security 2023

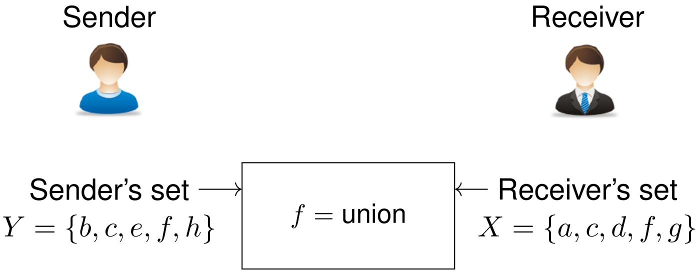
Outline

1. Background
2. KRTW Revisit
3. Multi-Query RPMT
4. Instantiation of mq-RPMT
5. Implement

Outline

1. Background
2. KRTW Revisit
3. Multi-Query RPMT
4. Instantiation of mq-RPMT
5. Implement

Private Set Union



Sender

Receiver

Sender's set

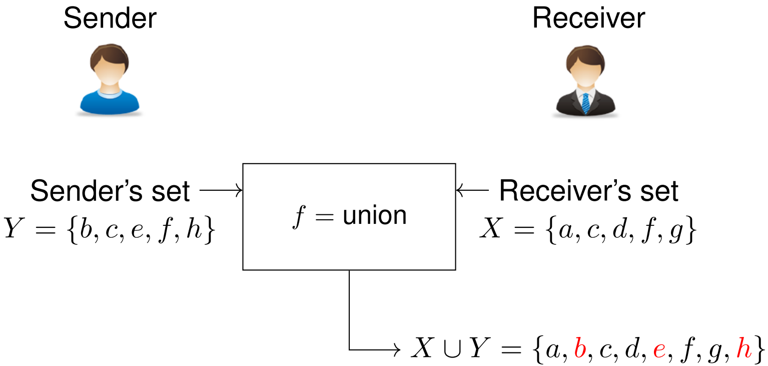
$Y = \{b, c, e, f, h\}$

$f =$ union

Receiver's set

$X = \{a, c, d, f, g\}$

Private Set Union



Sender

Receiver

Sender's set

$Y = \{b, c, e, f, h\}$

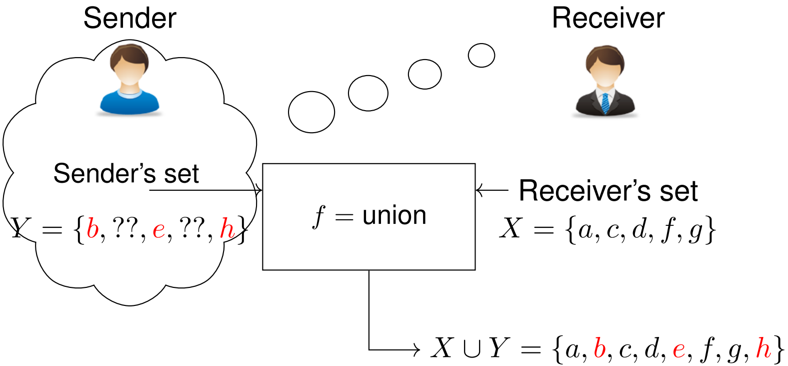
$f =$ union

Receiver's set

$X = \{a, c, d, f, g\}$

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

Private Set Union



Sender

Receiver

Sender's set

$Y = \{b, ??, e, ??, h\}$

$f =$ union

Receiver's set

$X = \{a, c, d, f, g\}$

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

Applications

- 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

There are two known approaches for constructing PSU:

1. Public-key techniques, e.g. additively homomorphic encryption (AHE) : [KS05, Fri07, DC17]
    - Pros
        - Can achieve linear communication complexity.
        - Can achieve "almost" linear computation complexity.
    - Cons
        - Computation is expensive. Have to perform a non-constant number of AHE operations on each set element.
        - Inefficient.
2. Symmetric-key techniques in combination with OT : [KRTW19, GMR$^+$21, JSZ$^+$22]
    - Pros
        - Computation is cheap.
        - Running time is several orders of magnitude faster than AHE-based constructions.
    - Cons
        - Communication complexity is nonlinear.
        - Computation complexity is nonlinear.

Motivation

Can we construct efficient PSU protocols with linear complexity?

Our Result

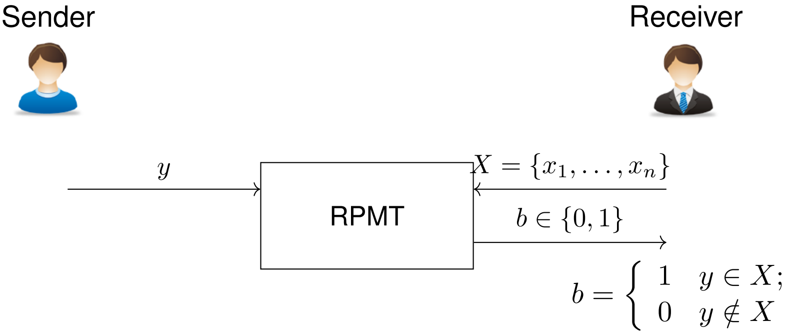
We focus on semi-honest setting. We propose a new framework for constructing PSU protocols and instantiate it based on different encryption schemes, they are:

1. A symmetric-key-based PSU protocol
   - Linear computation and communication complexity.
   - Only symmetric operations are used (except base OT).
2. A public-key-based PSU protocol
   - Linear computation and communication complexity.
   - The lowest concrete communication.

Outline

1. Background
2. KRTW Revisit
3. Multi-Query RPMT
4. Instantiation of mq-RPMT
5. Implement

Reverse Private Membership Test (RPMT)



Sender

Receiver

y

RPMT

$X = \{x_1, \dots, x_n\}$

$b \in \{0, 1\}$

$$b = \begin{cases} 1 & y \in X; \\ 0 & y \notin X \end{cases}$$

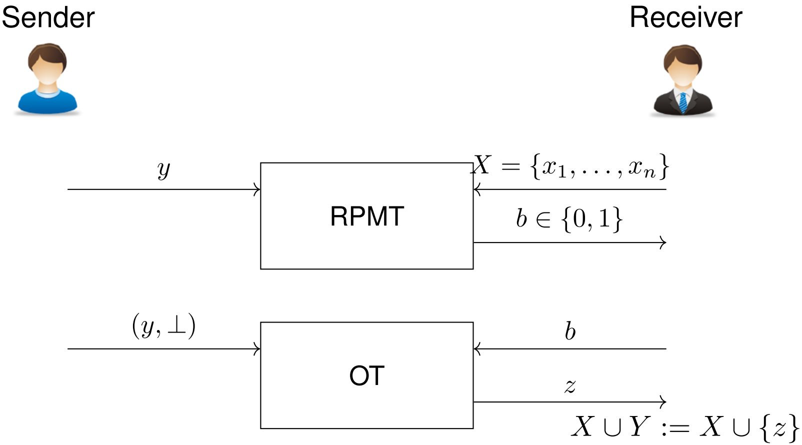
Learns nothing about X.

Learns nothing about which is the sender's item y.

Computation complexity of RPMT in [KRTW19]: $O(n \log^2 n)$.
Communication complexity of RPMT in [KRTW19]: $O(n)$.

KRTW19 Revisit

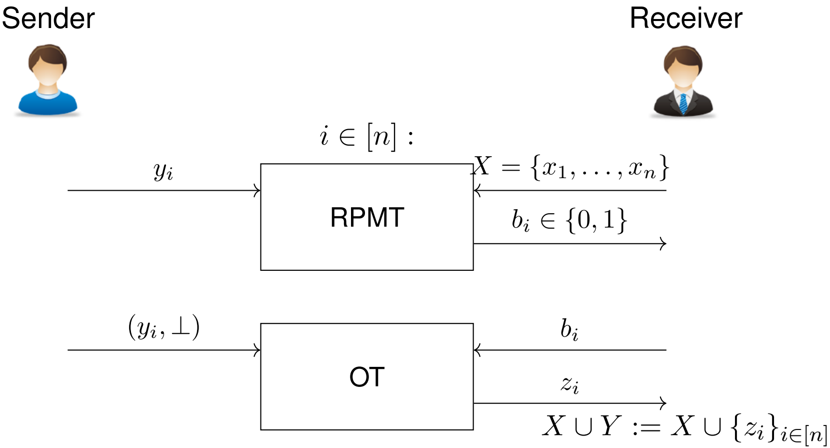
For a special case, the sender has only one item y in its set Y,



Computation complexity: $O(n \log^2 n)$.
Communication complexity: $O(n)$.

KRTW19 Revisit

Independent n times:



Sender

Receiver

$i \in [n]$:

y_i

RPMT

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

$b_i \in \{0, 1\}$

$(y_i, \perp)$

OT

b_i

z_i

$X \cup Y := X \cup \{z_i\}_{i \in [n]}$

Computation complexity: $O(n^2 \log^2 n)$

Communication complexity: $O(n^2)$

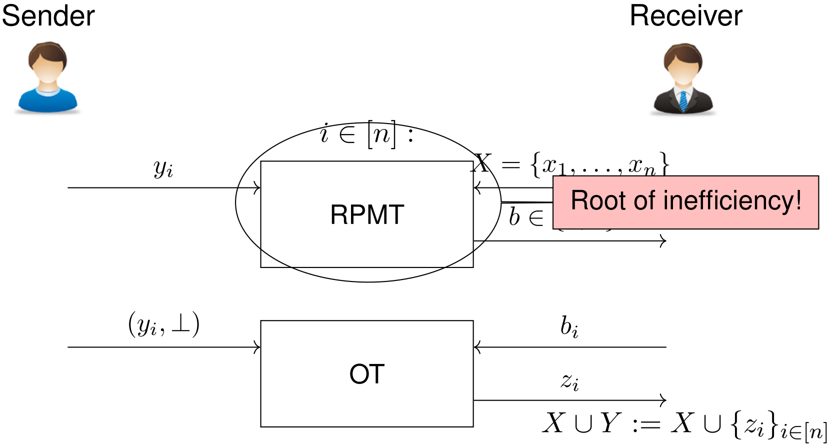
Hash to bin

$O(n \log n \log \log n)$

$O(n \log n)$

KRTW19 Revisit

Independent n times:



Sender

Receiver

$i \in [n]$:

y_i

RPMT

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

$b \in$

Root of inefficiency!

$(y_i, \perp)$

OT

b_i

z_i

$X \cup Y := X \cup \{z_i\}_{i \in [n]}$

Computation complexity: $O(n^2 \log^2 n)$ Hash to bin $O(n \log n \log \log n)$

Communication complexity: $O(n^2)$ $O(n \log n)$

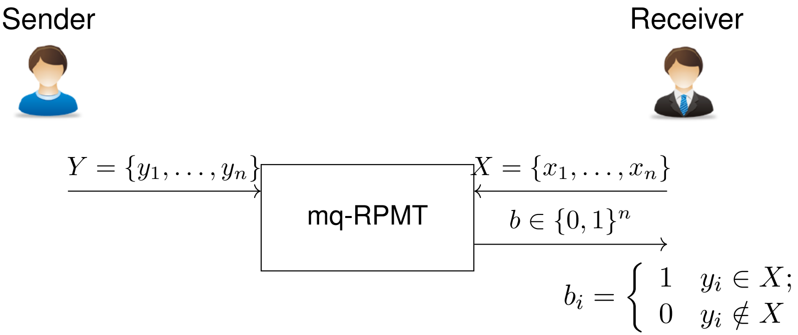
KRTW19 Revisit

Can we query multiple times in an RPMT instance?

Outline

1. Background
2. KRTW Revisit
3. Multi-Query RPMT
4. Instantiation of mq-RPMT
5. Implement

Definition of mq-RPMT



Sender

Receiver

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

mq-RPMT

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

$b \in \{0,1\}^n$

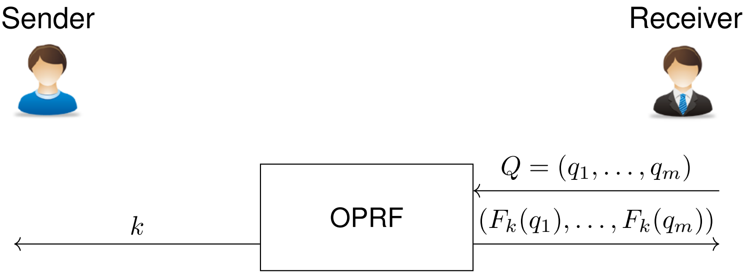
$$b_i = \begin{cases} 1 & y_i \in X; \\ 0 & y_i \notin X \end{cases}$$

Learns nothing about X.

Learns nothing about which is the sender's item y_i.

Our expectations:
Computation complexity: $O(n)$.
Communication complexity: $O(n)$.

Oblivious PRF (OPRF)



Sender

Receiver

$Q = (q_1, \ldots, q_m)$

OPRF

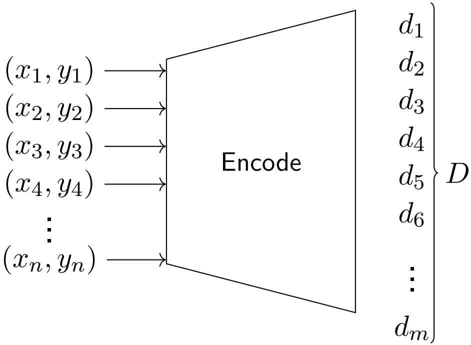
k

$(F_k(q_1), \ldots, F_k(q_m))$

Learns nothing about Q.

Learns nothing about k

Oblivious Key-Value Store



Rate:n/m
optimal is 1.

Encode Complexity:
complexity of Encode algorithm.

Decode Complexity:
complexity of Decode algorithm.

- Encode$((x_1, y_1), \ldots, (x_n, y_n)) \rightarrow D$
- Decode$(D, x) \rightarrow y$

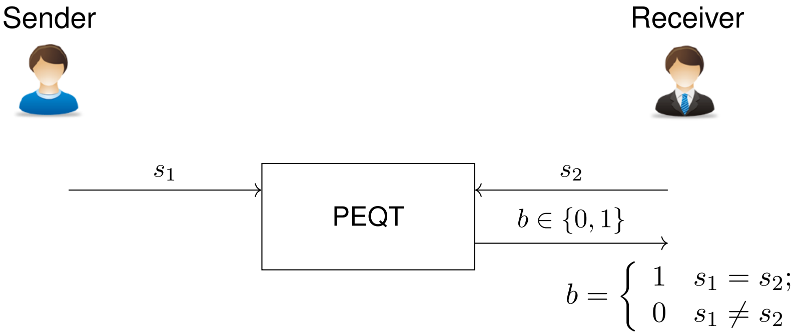
Oblivious Key-Value Store

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$^{\text{New!}}$ [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).

Private Equality Test (PEQT)



Sender

Receiver

s_1

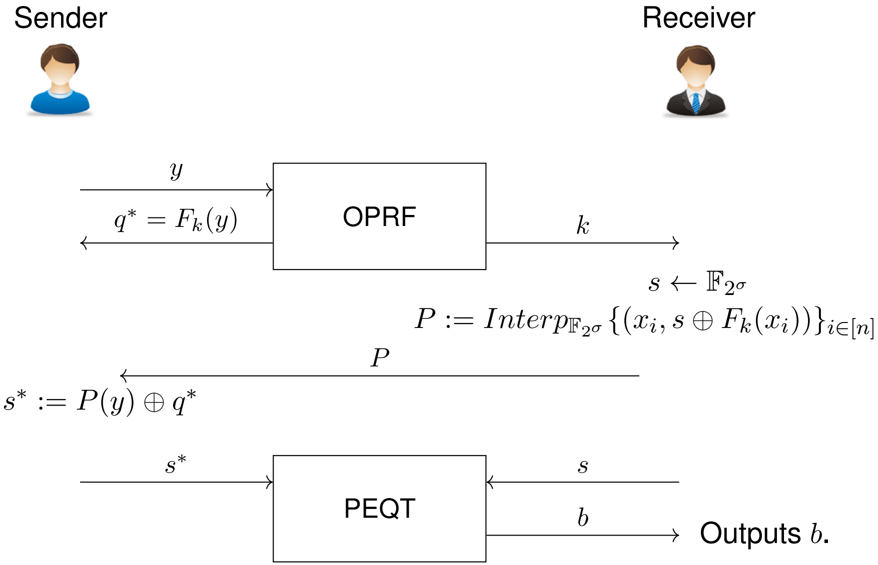
PEQT

s_2

$b \in \{0, 1\}$

$$b = \begin{cases} 1 & s_1 = s_2; \\ 0 & s_1 \neq s_2 \end{cases}$$

Zoom in on the original RPMT



Sender

Receiver

y

$q^* = F_k(y)$

OPRF

k

$s \leftarrow \mathbb{F}_{2^\sigma}$

$P := Interp_{\mathbb{F}_{2^\sigma}}\{(x_i, s \oplus F_k(x_i))\}_{i\in[n]}$

P

$s^* := P(y) \oplus q^*$

s^*

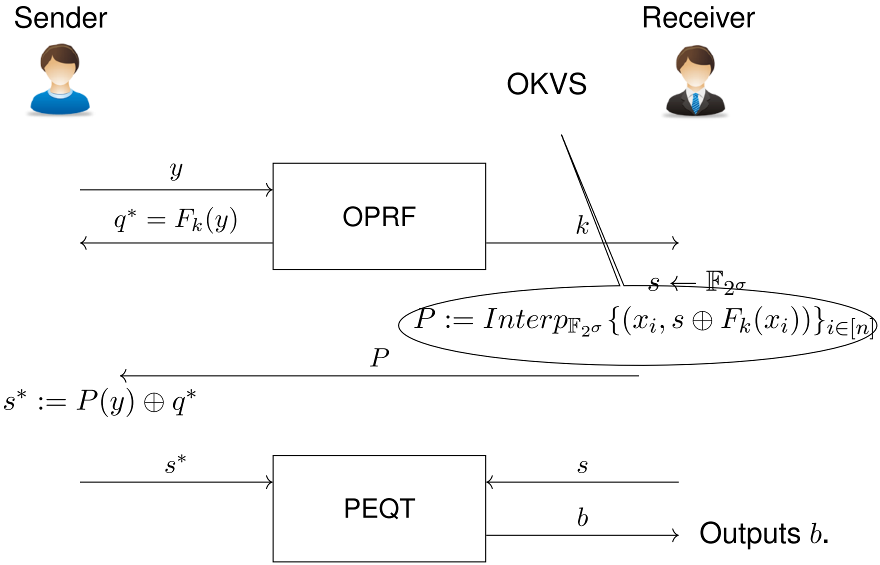
PEQT

s

b

Outputs b.

Zoom in on the original RPMT



Sender

Receiver

OKVS

y

$q^* = F_k(y)$

OPRF

k

$s \leftarrow \mathbb{F}_{2^\sigma}$

$P := Interp_{\mathbb{F}_{2^\sigma}}\{(x_i, s \oplus F_k(x_i))\}_{i \in [n]}$

P

$s^* := P(y) \oplus q^*$

s^*

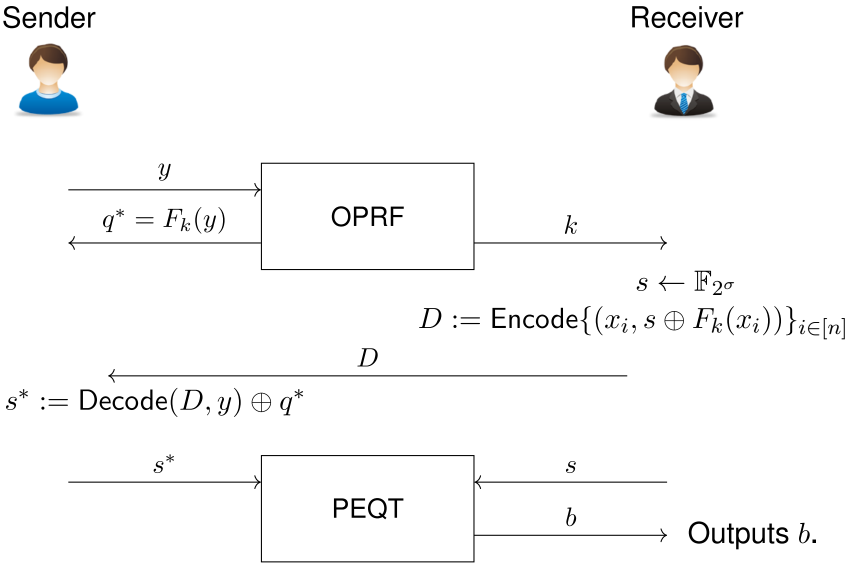
PEQT

s

b

Outputs b.

More efficient OKVS



Sender

Receiver

y

$q^* = F_k(y)$

OPRF

k

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D :=$ Encode$\{(x_i, s \oplus F_k(x_i))\}_{i \in [n]}$

D

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

s^*

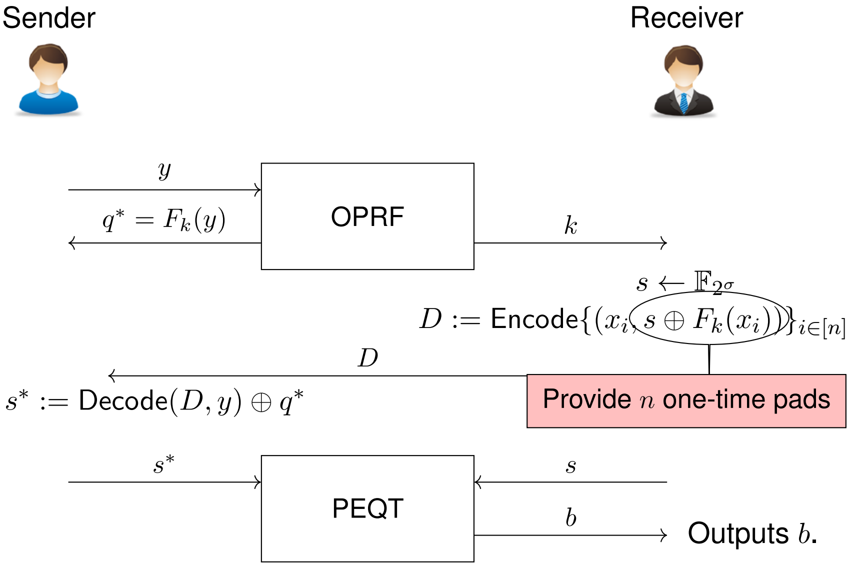
PEQT

s

b

Outputs b.

Usage of OPRF



Sender

Receiver

y

$q^* = F_k(y)$

OPRF

k

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \text{Encode}\{(x_i, s \oplus F_k(x_i))\}_{i\in[n]}$

D

$s^* := \text{Decode}(D, y) \oplus q^*$

Provide n one-time pads

s^*

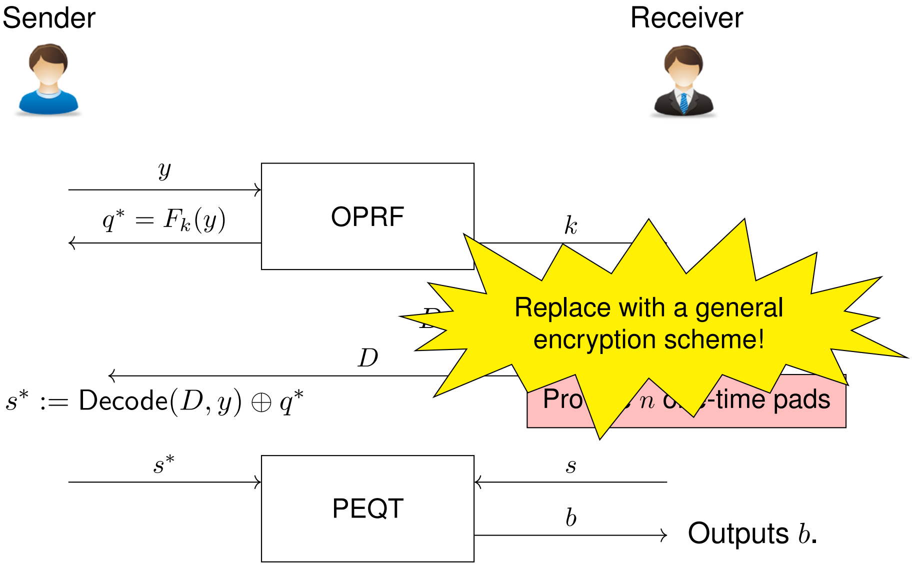
PEQT

s

b

Outputs b.

Usage of OPRF



Sender

Receiver

y

$q^* = F_k(y)$

OPRF

k

D

D

$s^* := \text{Decode}(D, y) \oplus q^*$

Replace with a general encryption scheme!

Pro n o -time pads

s^*

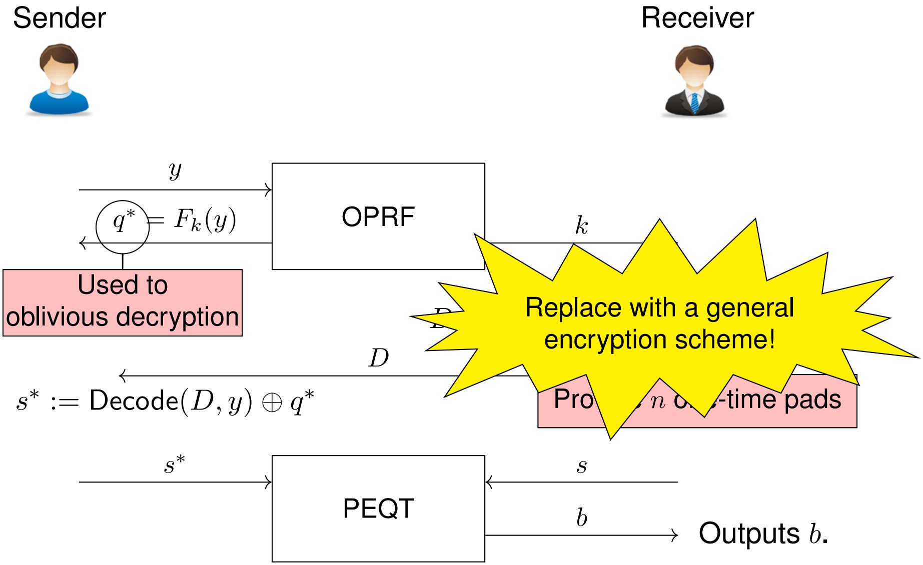
PEQT

s

b

Outputs b.

Usage of OPRF



Sender

Receiver

y

OPRF

$q^* = F_k(y)$

k

Used to oblivious decryption

D

D

Replace with a general encryption scheme!

Pro n o -time pads

$s^* := \text{Decode}(D, y) \oplus q^*$

s^*

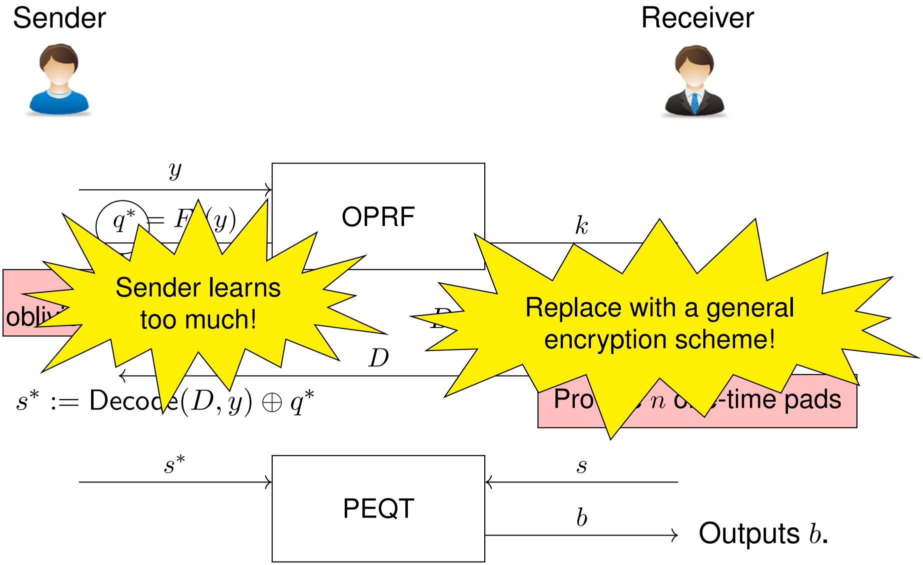
PEQT

s

b

Outputs b.

Usage of OPRF



Sender

Receiver

y

OPRF

$q^* = F(y)$

k

obliv

D

D

$s^* := \text{Decode}(D, y) \oplus q^*$

Pro n o -time pads

s^*

PEQT

s

b

Outputs b.

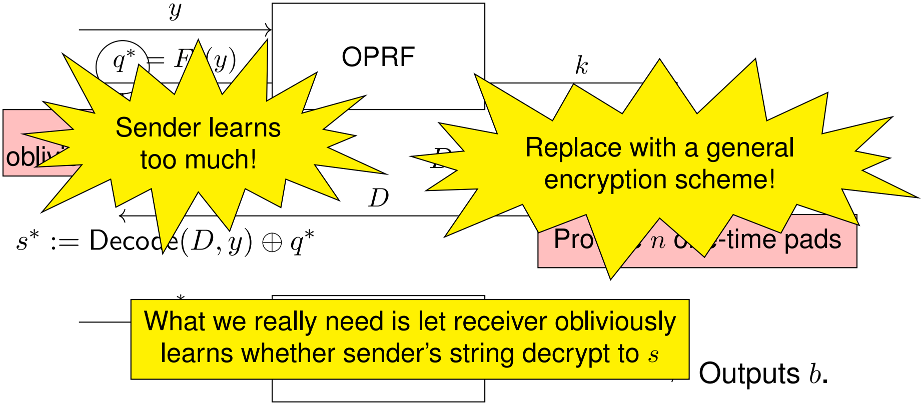
Sender learns too much!

Replace with a general encryption scheme!

Usage of OPRF

Sender

Receiver



y

OPRF

k

$q^* = F_k(y)$

obliv

D

D

$s^* := \text{Decode}(D, y) \oplus q^*$

Pro n o -time pads

*

Outputs b.

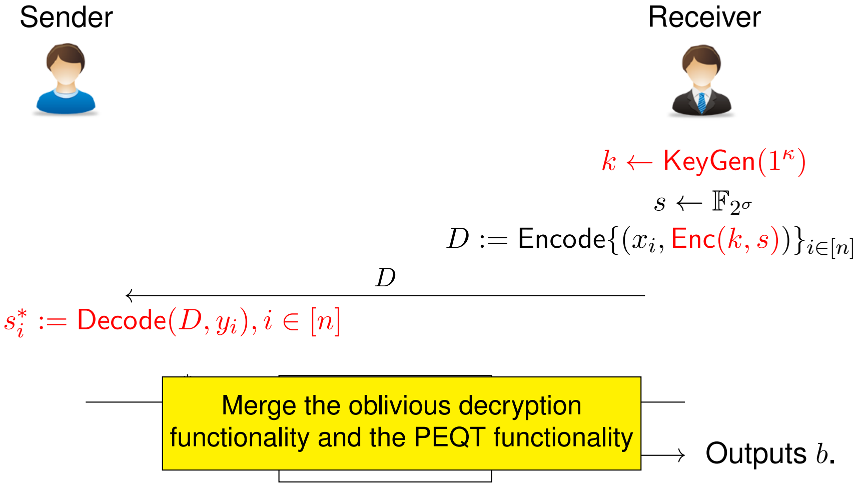
Sender learns too much!

Replace with a general encryption scheme!

What we really need is let receiver obliviously learns whether sender's string decrypt to s

Multi-Query RPMT

Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

$$\xleftarrow{\quad D \quad}$$

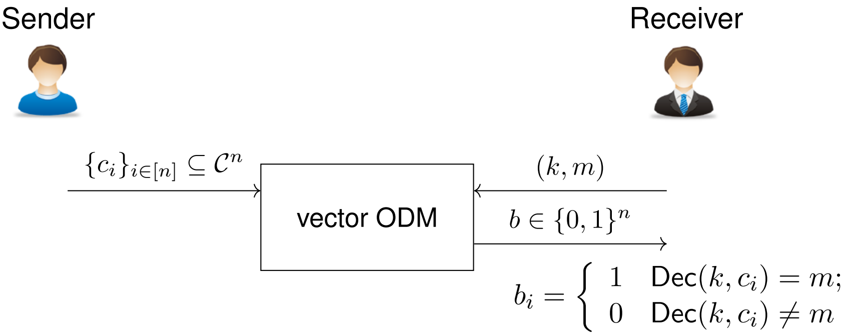
$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

Merge the oblivious decryption functionality and the PEQT functionality

Outputs b.

Vector Oblivious Decryption-then-Matching (vector ODM)

Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme.



Sender

Receiver

$\{c_i\}_{i\in[n]} \subseteq \mathcal{C}^n$

vector ODM

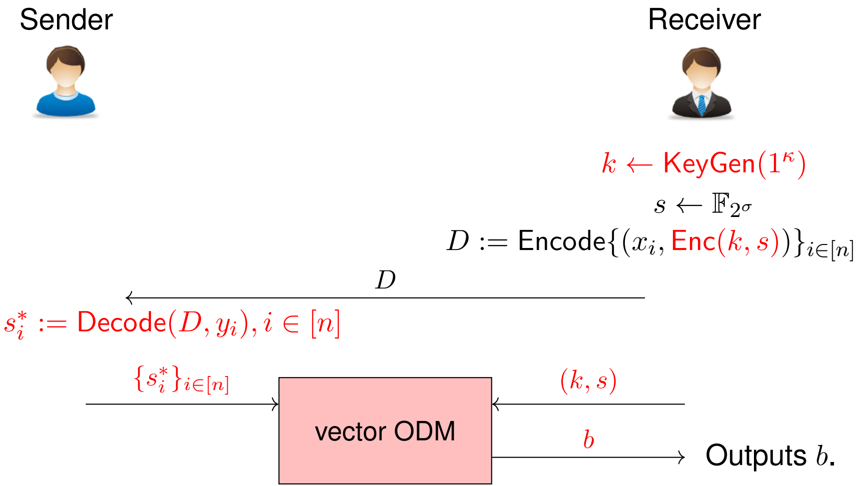
(k, m)

$b \in \{0,1\}^n$

$$b_i = \begin{cases} 1 & \mathsf{Dec}(k, c_i) = m; \\ 0 & \mathsf{Dec}(k, c_i) \neq m \end{cases}$$

Multi-Query RPMT

Let (Setup, KeyGen, Enc, Dec) be an encryption scheme.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i\in[n]}$

D

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\{s_i^*\}_{i\in[n]}$

(k, s)

vector ODM

b

Outputs b.

Outline

1 Background

2 KRTW Revisit

3 Multi-Query RPMT

4 Instantiation of mq-RPMT

5 Implement

SKE-based Instantiation

- $\mathsf{Setup}(1^\kappa) \to pp$.
- $\mathsf{KeyGen}(pp) \to k$.
- $\mathsf{Enc}(k, m) \to c$.
- $\mathsf{Dec}(k, c) \to m/\perp$.

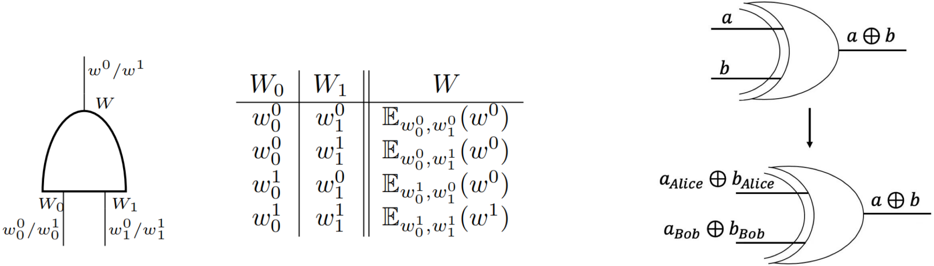
Security. For our purpose, we require a case-tailored security notion called *single-message multi-ciphertext pseudorandomness*. Formally, a SKE scheme is single-message multi-ciphertext pseudorandom if for any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\mathsf{Adv}_{\mathcal{A}}(1^\kappa) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ k \leftarrow \mathsf{KeyGen}(pp); \\ (m, state) \leftarrow \mathcal{A}_1(pp, k); \\ \beta \leftarrow \{0,1\}; \\ c^*_{i,0} \leftarrow \mathsf{Enc}(k, m), c^*_{i,1} \leftarrow C, \text{ for } i \in [n]; \\ \beta' \leftarrow \mathcal{A}_2(pp, state, \{c^*_{i,\beta}\}_{i \in [n]}) \end{array}\right] - \frac{1}{2}$$

is negligible in κ.

SKE-based Instantiation

Vector ODM: 2PC, e.g. Garbled Circuit [Yao86], GMW [GMW87].

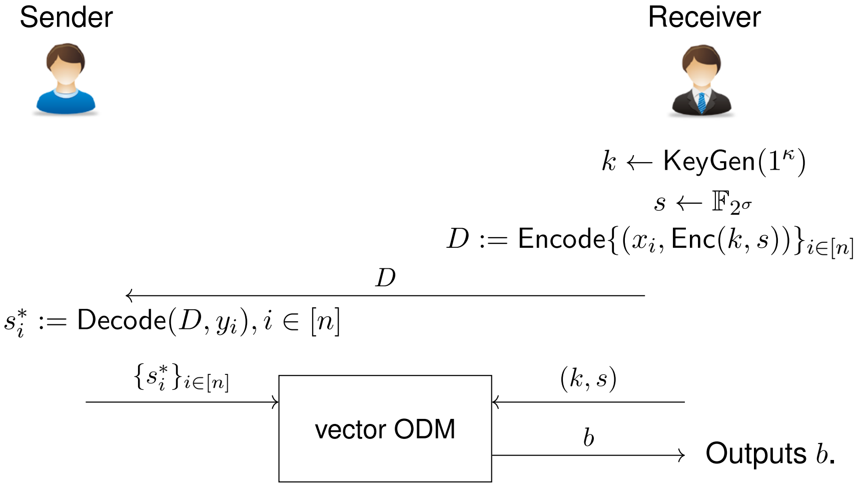


W_0	W_1	W
w_0^0	w_1^0	$\mathbb{E}_{w_0^0,w_1^0}(w^0)$
w_0^0	w_1^1	$\mathbb{E}_{w_0^0,w_1^1}(w^0)$
w_0^1	w_1^0	$\mathbb{E}_{w_0^1,w_1^0}(w^0)$
w_0^1	w_1^1	$\mathbb{E}_{w_0^1,w_1^1}(w^1)$

Figure: Garbled Circuit (left) and GMW (right)

Multi-Query RPMT

Let (Setup, KeyGen, Enc, Dec) be a SKE.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

D

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\{s_i^*\}_{i \in [n]}$

vector ODM

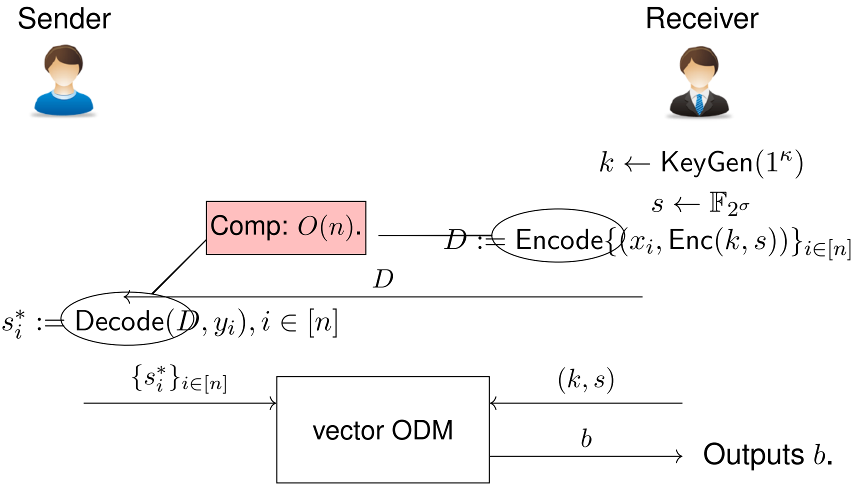
(k, s)

b

Outputs b.

Multi-Query RPMT

Let (Setup, KeyGen, Enc, Dec) be a SKE.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

Comp: $O(n)$.

D

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\{s_i^*\}_{i \in [n]}$

(k, s)

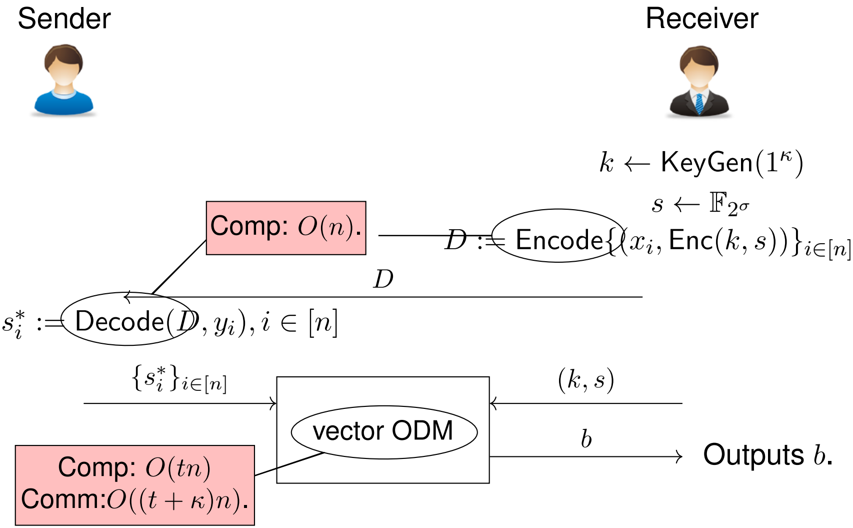
vector ODM

b

Outputs b.

Multi-Query RPMT

Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a SKE.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

Comp: $O(n)$.

$D := \text{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

D

$s_i^* := \text{Decode}(D, y_i), i \in [n]$

$\{s_i^*\}_{i \in [n]}$

(k, s)

vector ODM

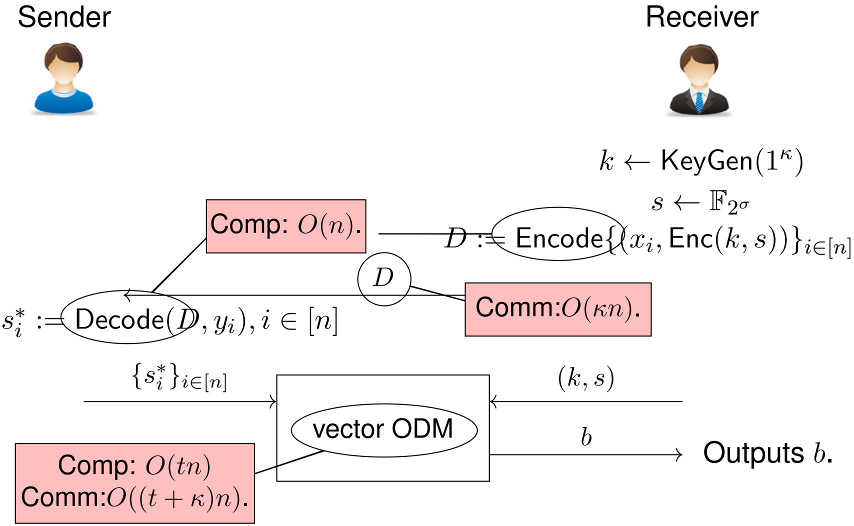
b

Outputs b.

Comp: $O(tn)$
Comm:$O((t + \kappa)n)$.

Multi-Query RPMT

Let (Setup, KeyGen, Enc, Dec) be a SKE.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

Comp: $O(n)$.

D

Comm:$O(\kappa n)$.

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\{s_i^*\}_{i \in [n]}$

(k, s)

vector ODM

b

Outputs b.

Comp: $O(tn)$
Comm:$O((t + \kappa)n)$.

PKE-based Instantiation

A re-randomizable PKE (ReRand-PKE) scheme is a tuple of five algorithms:

- $\mathsf{Setup}(1^\kappa) \to pp$.
- $\mathsf{KeyGen}(pp) \to (pk, sk)$.
- $\mathsf{Enc}(pk, m) \to c$.
- $\mathsf{Dec}(sk, c) \to m/\perp$.
- $\mathsf{ReRand}(pk, c) \to c'$.

Indistinguishability. For any $pp \leftarrow \mathsf{Setup}(1^\kappa)$, any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, and any $m \in M$, the distribution $c_0 \leftarrow \mathsf{Enc}(pk, m)$ and the distribution $c_1 \leftarrow \mathsf{ReRand}(pk, c_0)$ are identical.

PKE-based Instantiation

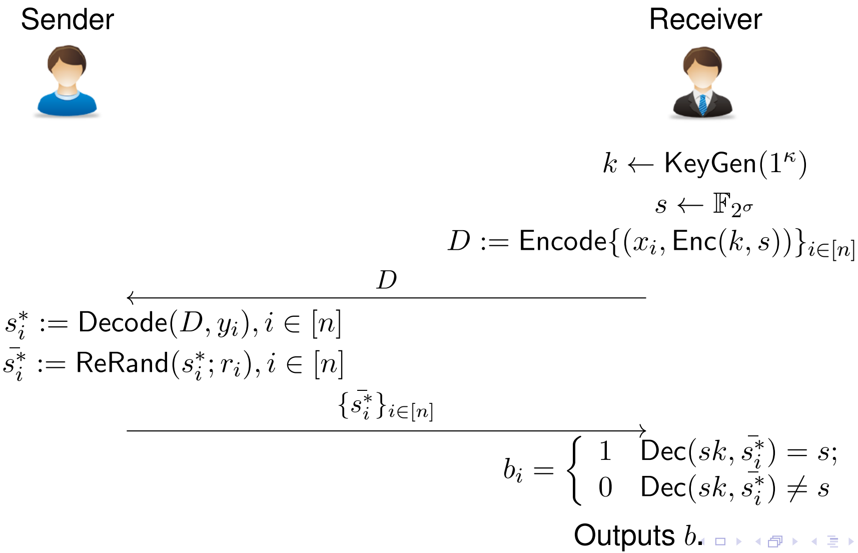
Security. For our purpose, we require a case-tailored security notion called *single-message multi-ciphertext pseudorandomness*. Formally, a PKE scheme is single-message multi-ciphertext pseudorandom if for any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\mathsf{Adv}_{\mathcal{A}}(1^\kappa) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (m, state) \leftarrow \mathcal{A}_1(pp, pk); \\ \beta \leftarrow \{0,1\}; \\ c^*_{i,0} \leftarrow \mathsf{Enc}(pk, m), c^*_{i,1} \leftarrow C, \text{ for } i \in [n]; \\ \beta' \leftarrow \mathcal{A}_2(pp, state, \{c^*_{i,\beta}\}_{i\in[n]}) \end{array}\right] - \frac{1}{2}$$

is negligible in κ.

PKE-based mq-RPMT

Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a ReRand PKE scheme.



Sender

Receiver

$$k \leftarrow \mathsf{KeyGen}(1^\kappa)$$

$$s \leftarrow \mathbb{F}_{2^\sigma}$$

$$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i\in[n]}$$

$$\xleftarrow{\quad D \quad}$$

$$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$$

$$\bar{s}_i^* := \mathsf{ReRand}(s_i^*; r_i), i \in [n]$$

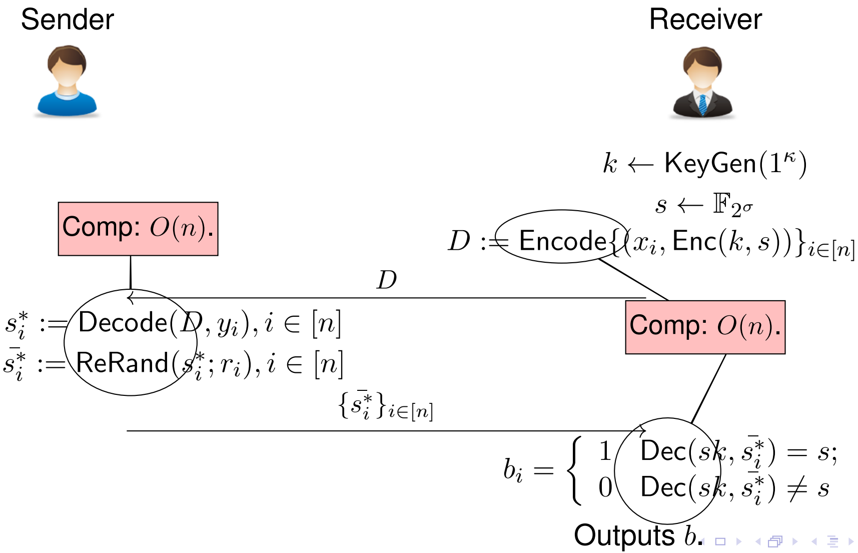
$$\xrightarrow{\quad \{\bar{s}_i^*\}_{i\in[n]} \quad}$$

$$b_i = \begin{cases} 1 & \mathsf{Dec}(sk, \bar{s}_i^*) = s; \\ 0 & \mathsf{Dec}(sk, \bar{s}_i^*) \neq s \end{cases}$$

Outputs b.

PKE-based mq-RPMT

Let (Setup, KeyGen, Enc, Dec) be a ReRand PKE scheme.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^\kappa)$

$s \leftarrow \mathbb{F}_{2^\sigma}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

Comp: $O(n)$.

D

Comp: $O(n)$.

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\bar{s}_i^* := \mathsf{ReRand}(s_i^*; r_i), i \in [n]$

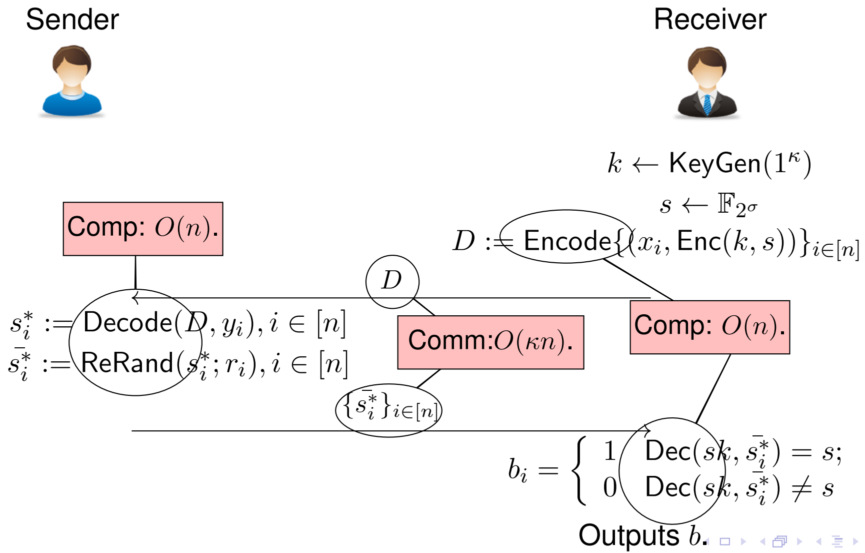
$\{\bar{s}_i^*\}_{i \in [n]}$

$$b_i = \begin{cases} 1 & \mathsf{Dec}(sk, \bar{s}_i^*) = s; \\ 0 & \mathsf{Dec}(sk, \bar{s}_i^*) \neq s \end{cases}$$

Outputs b.

PKE-based mq-RPMT

Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a ReRand PKE scheme.



Sender

Receiver

$k \leftarrow \mathsf{KeyGen}(1^{\kappa})$

$s \leftarrow \mathbb{F}_{2^{\sigma}}$

$D := \mathsf{Encode}\{(x_i, \mathsf{Enc}(k, s))\}_{i \in [n]}$

Comp: $O(n)$.

D

Comm:$O(\kappa n)$.

$s_i^* := \mathsf{Decode}(D, y_i), i \in [n]$

$\bar{s}_i^* := \mathsf{ReRand}(s_i^*; r_i), i \in [n]$

Comp: $O(n)$.

$\{\bar{s}_i^*\}_{i \in [n]}$

$$b_i = \begin{cases} 1 & \mathsf{Dec}(sk, \bar{s}_i^*) = s; \\ 0 & \mathsf{Dec}(sk, \bar{s}_i^*) \neq s \end{cases}$$

Outputs b.

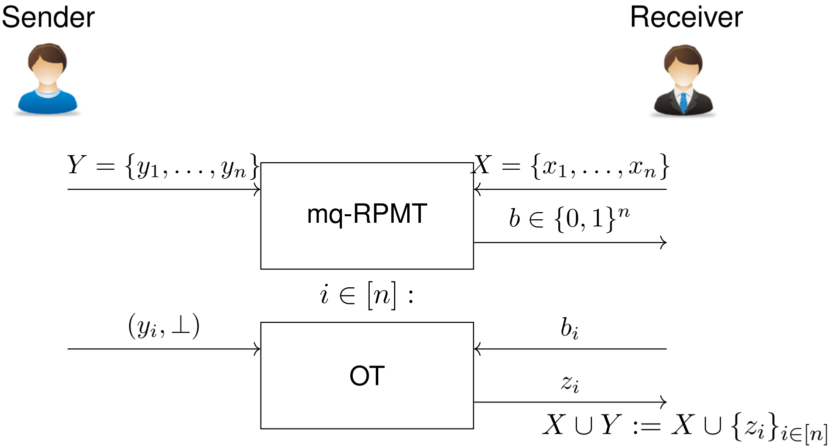
Unification with Membership Encryption

Definition (Membership Encryption)

Membership encryption for set X consists of four polynomial time algorithms satisfying the following properties.

- $\mathsf{Setup}(1^\kappa)$: on input a security parameter κ, outputs public parameters pp, which include the ciphertext space C.
- $\mathsf{KeyGen}(pp, X)$: on input public parameters pp and $X \subseteq \{0,1\}^*$, outputs a key k.
- $\mathsf{Enc}(k, x)$: on input a key k and an element $x \in X$, outputs a ciphertext $c \in C$. For uttermost generality, the behavior of Enc on $x \notin X$ is unspecified. Looking ahead, such treatment suffices for the construction of mq-RPMT protocol.
- $\mathsf{Dec}(k, c)$: on input a key k and a ciphertext $c \in C$, outputs “1” indicating c is an encryption of an element x in X and “0” if not.

Final PSU



Sender

Receiver

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

mq-RPMT

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

$b \in \{0,1\}^n$

$i \in [n]$:

$(y_i, \perp)$

OT

b_i

z_i

$X \cup Y := X \cup \{z_i\}_{i \in [n]}$

Computation complexity: $O(n)$.
Communication complexity: $O(\kappa n)$.

Outline

1. Background
2. KRTW Revisit
3. Multi-Query RPMT
4. Instantiation of mq-RPMT
5. Implement

Implement

n	Protocol	Comm. (MB)					Running time (s)															
		$\mathcal{R}$		$\mathcal{S}$		total	LAN				1Gbps				100Mbps				10Mbps			
		setup	online	setup	online		$T=1$		$T=8$		$T=1$		$T=8$		$T=1$		$T=8$		$T=1$		$T=8$	
							setup	online	setup	online	setup	online	setup	online	setup	online	setup	online	setup	online	setup	online
2^{14}	KRTW	0.02	4.17	0.01	29.63	33.8	0.07	3.5	0.03	1.07	0.49	16.13	0.37	14.06	0.83	27.36	0.72	24.66	0.81	55.9	0.73	55.32
	GMRSS	0.02	5.89	0.02	7.96	13.85	0.1	1.01	0.04	0.42	0.66	1.96	0.46	1.28	1	3.53	0.91	2.97	1.06	14.44	0.93	13.97
	JSZDG-R	0.01	4.65	0.01	5.63	10.28	0.07	1.81	0.02	0.52	0.27	2.65	0.23	1.34	0.49	4.19	0.41	2.66	0.45	12.08	0.37	10.63
	SKE-PSU	0.01	3.16	0	3.36	6.52	0.03	0.65	0.02	0.29	0.12	6.76	0.11	6.48	0.21	12.66	0.19	12.09	0.2	15.62	0.19	15.59
	PKE-PSU	0.01	1.16	0	1.59	2.75	4.6	2.37	4.58	1.07	4.78	2.63	4.75	1.34	4.92	3.02	4.9	1.77	4.99	4.43	4.91	3.79
	PKE-PSU*	0.01	2.16	0	2.9	5.05	4.6	1.96	4.6	0.59	4.75	2.36	4.76	1	4.95	2.76	4.91	1.54	4.92	5.72	4.93	5.31
2^{16}	KRTW	0.02	17.64	0.01	122.05	139.69	0.07	12.57	0.03	3.76	0.46	26.27	0.39	20.96	0.82	40.09	0.73	36.3	0.81	163.48	0.75	161.63
	GMRSS	0.02	25.95	0.02	34.11	60.06	0.11	4.79	0.04	1.95	0.64	6.61	0.48	4.25	1.11	12.67	0.92	9.78	1.04	60.75	0.94	57.5
	JSZDG-R	0.01	20.75	0.01	24.74	45.49	0.07	7.5	0.02	2.25	0.3	9.29	0.2	4.45	0.44	13.78	0.4	8.58	0.47	49.41	0.42	44.58
	SKE-PSU	0.01	12.61	0	13.41	26.03	0.04	2.66	0.02	1.15	0.13	8.66	0.11	7.32	0.2	15.84	0.19	14.39	0.2	31.79	0.19	30.98
	PKE-PSU	0.01	4.62	0	6.37	10.99	4.62	9.75	4.59	4.39	4.82	10.21	4.76	5.22	4.9	10.94	4.91	5.83	5.01	16.38	4.92	13.61
	PKE-PSU*	0.01	8.63	0	11.57	20.19	4.57	7.96	4.6	2.58	4.76	8.68	4.77	3.37	4.93	9.94	4.91	4.65	4.94	21.46	4.93	19.67
2^{18}	KRTW	0.02	69.29	0.01	562.76	632.05	0.08	63.02	0.03	17.67	0.52	85.56	0.39	45.31	0.76	111.14	0.71	113.83	0.84	660.33	0.74	664.93
	GMRSS	0.02	113.7	0.02	145.11	258.81	0.13	20.74	0.03	9.8	0.58	28.62	0.55	16.63	1.09	49.68	0.93	38.82	1.03	251.84	0.97	243.63
	JSZDG-R	0.01	92.67	0.01	107.89	200.56	0.07	41.15	0.03	10.71	0.25	43.17	0.21	16.84	0.42	64.06	0.4	33.8	0.53	221.27	0.39	191.2
	SKE-PSU	0.01	50.34	0	53.51	103.85	0.04	10.78	0.02	4.88	0.12	17.83	0.1	12.32	0.2	28.38	0.18	22.54	0.21	98.96	0.19	95.72
	PKE-PSU	0.01	18.5	0	25.45	43.95	4.6	41.5	4.59	19.82	4.79	42.37	4.75	20.97	4.92	44.8	4.91	23.38	4.92	66.68	4.9	54.39
	PKE-PSU*	0.01	34.5	0	46.26	80.76	4.61	34.63	4.58	12.26	4.78	37.1	4.75	13.99	4.92	40.62	4.92	18.45	4.91	85.31	4.92	79.22
2^{20}	KRTW	0.02	300.14	0.01	2305.8	2605.95	0.11	245.37	0.04	67.97	0.52	281.96	0.38	120.35	0.82	363.95	0.74	361.12	0.84	2643.84	0.75	2638.05
	GMRSS	0.02	493.2	0.02	615.9	1109.1	0.11	100.48	0.04	48.53	0.62	119.98	0.51	75.76	1.11	207.83	0.95	164.25	1.09	1074.33	0.95	1030.3
	JSZDG-R	0.01	405.53	0.01	467.26	872.79	0.08	173.07	0.04	54.41	0.48	184.63	0.2	73.28	0.47	266.51	0.73	146.13	0.47	941.5	0.72	825.16
	SKE-PSU	0.01	200.88	0	213.55	414.43	0.05	44.73	0.03	22.78	0.13	59.65	0.11	35.71	0.2	86.11	0.2	65.18	0.21	378.57	0.4	369.24
	PKE-PSU	0.01	74	0	101.8	175.8	4.65	168.79	4.6	79.95	4.78	169.18	4.79	86.49	4.97	179.58	4.94	96.32	4.97	269.32	4.87	216.19
	PKE-PSU*	0.01	138	0	185	323	4.64	144.24	4.58	50.56	4.75	146.41	4.74	60.5	4.9	161.26	5	76.33	4.99	345	4.9	313.37

Table: Communication cost (in MB) and running time (in seconds) comparing our protocols to KRTW GMRSS, and JSZDG-R. The LAN network has 10 Gbps bandwidth and 0.2 ms RTT latency. Communication cost of $\mathcal{S}/\mathcal{R}$ indicates the outgoing communication from $\mathcal{S}/\mathcal{R}$ to the other party. The best protocol within a setting is marked in blue.

Implement

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

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

eprint: `https://eprint.iacr.org/2022/358`

THANK YOU!

Q & A

Reference

[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.

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

[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.

[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. Cryptology ePrint Archive, Report 2021/243, 2021. https://eprint.iacr.org/2021/243.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC 1987*, 1987.

[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.

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

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from paxos: Fast, malicious private set intersection. In *EUROCRYPT 2020*, 2020.

[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.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.