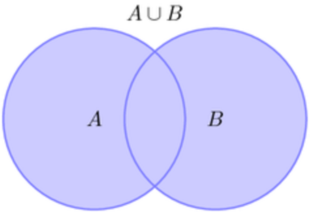
Linear Private Set Union from Multi-Query Reverse Private Membership Test



Yu Chen
Shandong University

joint work[1] with Cong Zhang, Weiran Liu, Min Zhang and Dongdai Lin

[1] USENIX Security 2023: Linear Private Set Union from Multi-Query Reverse Private Membership Test. Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, Dongdai Lin.

Outline

Outline

MPC with Semi-Honest Security

$P_1(r_1)$ ... $P_i(r_i)$ ... $P_n(r_n)$

x_1 ... x_i ... x_n

Π for $f(x_1, \ldots, x_n)$

$\text{output}_1(x_1, \ldots, x_n)$ ... $\text{output}_i(x_1, \ldots, x_n)$ ... $\text{output}_n(x_1, \ldots, x_n)$

$f_1(x_1, \ldots, x_n)$ $f_i(x_1, \ldots, x_n)$ $f_n(x_1, \ldots, x_n)$

P_i's view includes
private input
random tape
incoming messages

MPC with Semi-Honest Security

$P_1(r_1)$ ... $P_i(r_i)$ ... $P_n(r_n)$

$x_1 \downarrow$ ... $x_i \downarrow$ ... $x_n \downarrow$

Π for $f(x_1, \ldots, x_n)$

$\mathsf{output}_1(x_1, \ldots, x_n)$ ... $\mathsf{output}_i(x_1, \ldots, x_n)$ ... $\mathsf{output}_n(x_1, \ldots, x_n)$

$f_1(x_1, \ldots, x_n)$ $f_i(x_1, \ldots, x_n)$ $f_n(x_1, \ldots, x_n)$

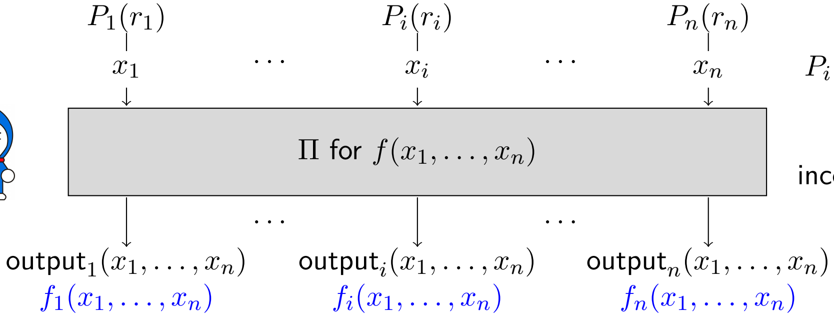
P_i's view includes
private input
random tape
incoming messages

- all P_i are semi-honest (honest but curious)
- P_i learns no more information other than his output and private input

Definition 1 (Semi-honest Security)

Π securely realizes probabilistic f in the presence of semi-honest adversaries if there exists a PPT simulator Sim such that for all inputs $x_1, \ldots, x_n$ and all $i \in [n]$:
$(\mathsf{View}_{P_i}(x_1, \ldots, x_n), \mathsf{output}(x_1, \ldots, x_n)) \approx_{c,s} (\mathsf{Sim}(i, x_i, f_i(x_1, \ldots, x_n)), f(x_1, \ldots, x_n))$

MPC with Semi-Honest Security



P_i's view includes
private input
random tape
incoming messages

- all P_i are semi-honest (honest but curious)
- P_i learns no more information other than his output and private input

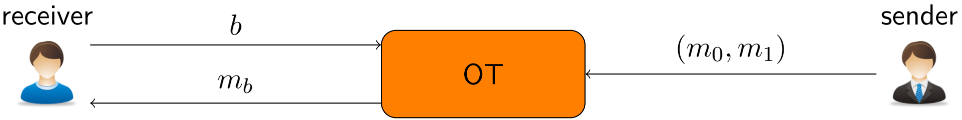
Definition 1 (Semi-honest Security)

Π securely realizes deterministic f in the presence of semi-honest adversaries if there exists a PPT simulator Sim such that for all inputs $x_1, \ldots, x_n$ and all $i \in [n]$:

$$\mathsf{View}_{P_i}(x_1, \ldots, x_n) \approx_{c,s} \mathsf{Sim}(i, x_i, f_i(x_1, \ldots, x_n))$$

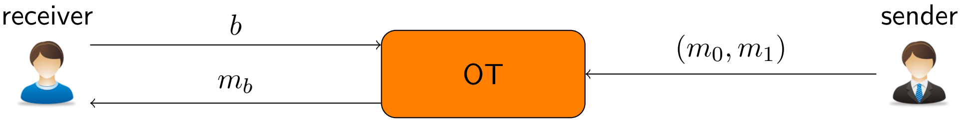
Oblivious Transfer

1-out-of 2 OT [Rab05] enables the receiver learns only one messages from sender, while sender learns nothing.



Oblivious Transfer

1-out-of 2 OT [Rab05] enables the receiver learns only one messages from sender, while sender learns nothing.

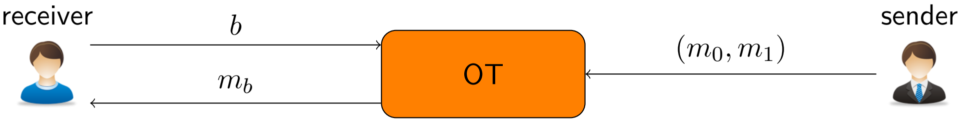


OT is complete for MPC [Kil88].

- Private-information retrieval (PIR) is weaker than OT: it only cares privacy of receiver

Oblivious Transfer

1-out-of 2 OT [Rab05] enables the receiver learns only one messages from sender, while sender learns nothing.



OT is complete for MPC [Kil88].

- Private-information retrieval (PIR) is weaker than OT: it only cares privacy of receiver

OT does not belong to Minicrypt $\rightsquigarrow$ expensive public-key operations are unavoidable, while real applications need a large number of OT

- [IKNP03] proposed Ishai-Kilian-Nissim-Petrank OT extension: generate many OT efficiently from $O(\kappa)$ number of base OT $\Rightarrow$ OTe is cheap

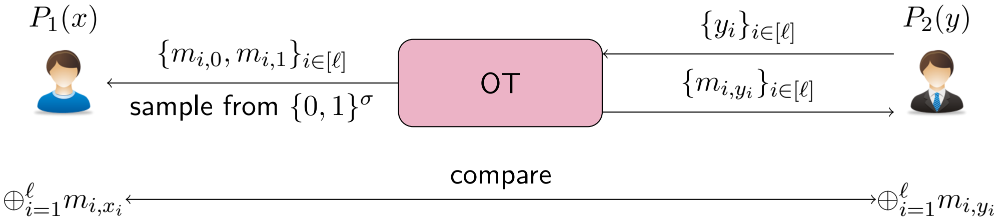
Private Equality Test Protocol

PEQT [PSSZ15] enables P_1 and P_2 check if their ℓ-bits elements x and y are equal.

Private Equality Test Protocol

PEQT [PSSZ15] enables P_1 and P_2 check if their ℓ-bits elements x and y are equal.

[PSSZ15] showed how to build PEQT by invoking 1-out-of-2 random OT ℓ times



Oblivious Pseudorandom Functions

OPRF [FIPR05] enables server obtain a key k and client evaluate obliviously.

PRF $F : K \times D \to R$

Oblivious Pseudorandom Functions

OPRF [FIPR05] enables server obtain a key k and client evaluate obliviously.

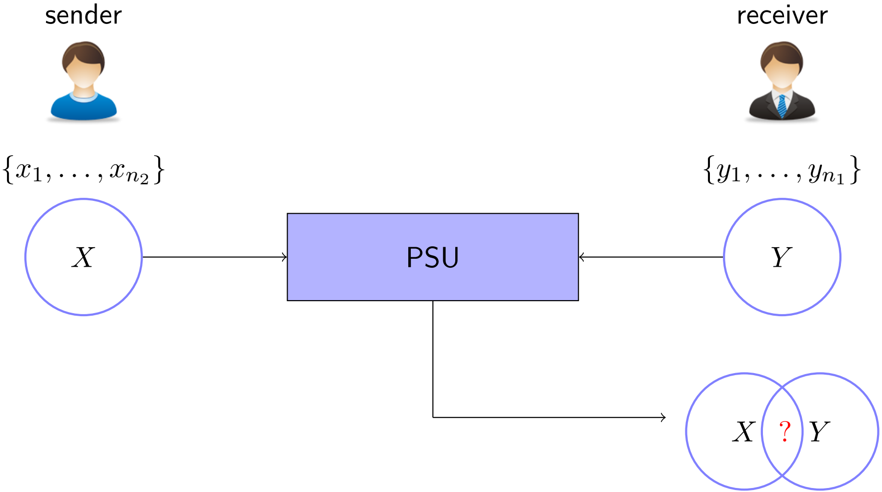
PRF $F : K \times D \to R$

OPRF is a powerful tool in MPC (see [CHL22] for a good survey)

- many variants: batch/programmable/permuted/distributed OPRF
- fast construction from OT or VOLE

Outline

Private Set Union



In this work, we focus on the balanced setting, i.e., $n_1 \approx n_2$. For simplicity, we assume $n_1 = n = n_2$ hereafter.

Applications of PSU

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

According to the underlying techniques, existing PSU protocols can be divided into two categories:

- Public-key techniques (e.g. AHE) [KS05, Fri07, DC17]
  - Pros. good asympotic complexity: "almost" linear computation/communication complexity
  - Cons. poor concrete efficiency: $O(\lambda)$ AHE operations per set element
- Symmetric-key techniques coupled with OT [KRTW19, GMR+21, JSZ+22]
  - Pros. (i) good concrete efficiency: running time is several orders of magnitude faster than AHE-based protocols;
  - Cons. poor asympotic complexity: communication/computation complexity are superlinear

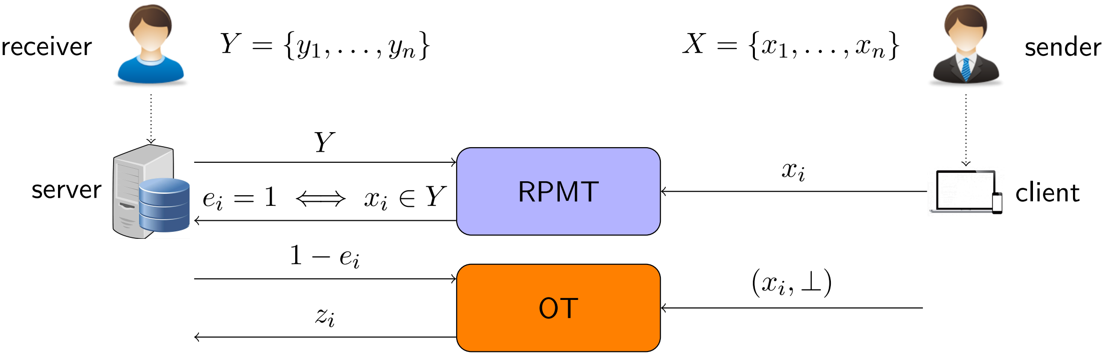
Protocols based on symmetric-key techniques are plausibly quantum secure.

Motivation

Can we attain the-best-of-two-worlds: designing PSU protocols with optimal linear complexity and good concrete efficiency?

Outline

Review of KRTW (Kolesnikov-Rosulek-Trieu-Wang) Protocol



$$z_i = \begin{cases} x_i & e_i = 0 \\ \perp & e_i = 1 \end{cases}$$

repeat the 1-vs-many PSU n times independently

Zoom In of the Sub-protocol RPMT

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

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

$F : K \times D \to \{0,1\}^\ell$

indication string of Y

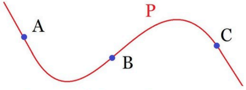
$s \xleftarrow{\mathsf{R}} \mathbb{F}_{2^\ell}$

k ← **OPRF** ← x

OPRF → $F_k(x)$

$P \leftarrow \text{Interpolate}\{(y_i, s \oplus F_k(y_i))_{i\in[n]}\}$ → P

Lagrange Interpolation



Polynomial through any points

s → **PEQT** ← s^* ← $s^* \leftarrow P(x) \oplus F_k(x)$

$b := s \stackrel{?}{=} s^*$ ← **PEQT**

Zoom In of the Sub-protocol RPMT

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

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

$F : K \times D \to \{0,1\}^\ell$

indication string of Y

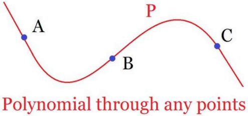
$s \xleftarrow{\mathsf{R}} \mathbb{F}_{2^\ell}$

$\xleftarrow{k}$ **OPRF** $\xleftarrow{x}$

OPRF $\xrightarrow{F_k(x)}$

$P \leftarrow$ Interpolate$\{(y_i, s \oplus F_k(y_i))_{i \in [n]}\}$ $\xrightarrow{P}$

Lagrange Interpolation



$\xrightarrow{s}$ **PEQT** $\xleftarrow{s^*}$ $s^* \leftarrow P(x) \oplus F_k(x)$

$\xleftarrow{b := s \overset{?}{=} s^*}$ **PEQT**

Usage of OPRF. Without OPRF masking, s^* reveals too much information about $x \rightsquigarrow$ compromise sender's privacy

Correctness and Security Analysis

Correctness. Consider the following two cases:

- If $x \in Y \Rightarrow s^* = \underline{P(y_i)} \oplus F_k(y_i) = \underline{s \oplus F_k(y_i)} \oplus F_k(y_i) = s$.
- If $x \notin Y \Rightarrow F_k(x)$ is pseudorandom. Via real-or-random argument, We conclude that for PPT receiver and sender, $\Pr[s^* = F_k(x) \oplus P(x) = s] \leq 1/2^\ell$ in computational sense.

$$x \in Y \iff s = s^*$$

Security. Follows the semi-honest security of OPRF and PEQT.

Complexity Analysis

OPRF and PEQT are fast cryptographic protocols

The computation bottleneck lies at polynomial interpolation of *arbitrary* n points

- trivial algorithm using Langrange formula requires $O(n^2)$
- fast algorithm using FFT requires $O(n \log^2 n)$

The communication bottleneck lies at the representation of degree-n polynominal

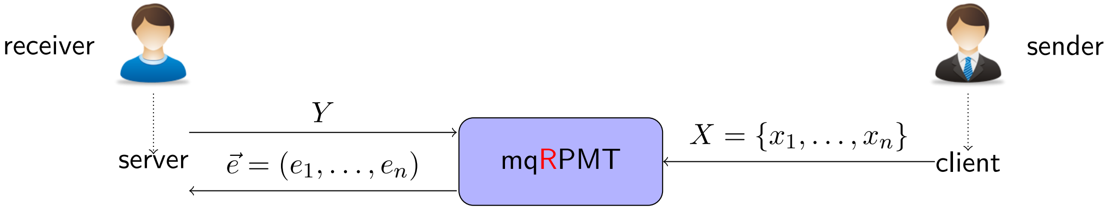
- $O(n)$ field elements in $\mathbb{F}_{2^\ell}$

In sum, KRTW protocol has $O(n^2 \log^2 n)$ computation complexity and $O(n^2)$ communication complexity[2]

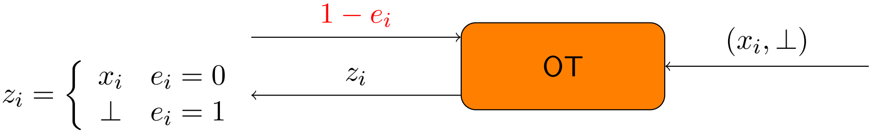
[2] In [KRTW19], hash-to-bin technique was used to reduce complexity. However, Jia et al. [JSZ$^+$22] pin-pointed that the improved protocol is not secure.

Outline

PSU from mqRPMT



yields PSU coupled with OT (flipping $\vec{e}$): receiver obtains $X \backslash Y$



How to Batch mqRPMT to Build Efficient mqRPMT

Root of inefficiency for KRTW protocol

1. degree-n polynominal interpolation is heavy
2. have to repeat polynominal interpolation n times, while batch the basic RPMT protocol is not trivial:
    - sender learns the purported indication string s^* in clear $\Rightarrow$ direct reusing P let sender be able to decide if $\underline{x_i \in Y \wedge x_j \in Y}$ by computing and compairing $s^* \rightsquigarrow$ compromise receiver's privacy

How to Batch mqRPMT to Build Efficient mqRPMT

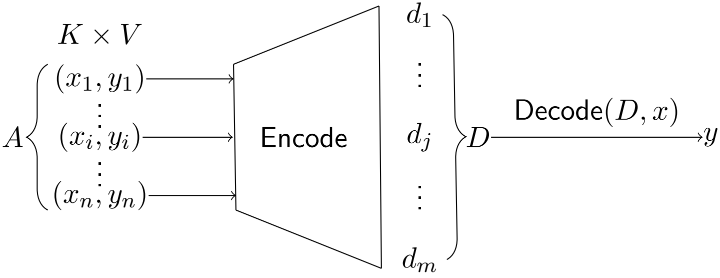
Root of inefficiency for KRTW protocol

1. degree-n polynominal interpolation is heavy
2. have to repeat polynominal interpolation n times, while batch the basic RPMT protocol is not trivial:
   - sender learns the purported indication string s^* in clear $\Rightarrow$ direct reusing P let sender be able to decide if $\underline{x_i \in Y \wedge x_j \in Y}$ by computing and compairing $s^* \rightsquigarrow$ compromise receiver's privacy

Our idea is based on two key observations.

- 1st Observation. Polynomial interpolation plays the role of oblivious key-value store.
- 2nd Observation. The usage of OPRF is three-fold:
  - receiver uses OPRF to derive n pseudorandom one-time pads, then encrypts the same s^* into n ciphertexts under these one-time pads.
  - sender uses OPRF to decrypt a ciphertext obliviously.
  - OPRF infuses polynomial interpolation with randomness to ensure the correctness.

Oblivious Key-Value Store



rate n/m
1 is optimal

Correctness. For any $A = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and any $x_i \in \{x_1, \dots, x_n\}$:

$$\Pr[\text{Decode}(D, x_i) = y_i] \geq 1 - \text{negl}(\lambda), \text{ where } D \leftarrow \text{Encode}(A).$$

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

$$\text{Decode}(D, x) \approx_s U_V, \text{ where } D \leftarrow \text{Encode}(A).$$

Obliviousness. For any $(x_1^0, \dots, x_n^0) \neq (x_1^1, \dots, x_n^1)$:

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

OKVS Off-the-Shelf

Table: Comparison of Different OKVS

scheme	rate	encoding	decoding	randomness	obliviousness
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 # [key-value pairs]. λ is the statistical security parameter (e.g. $\lambda = 40$).

Drop-in replacement of polynomial interpolation with better OKVS will improve efficiency immediately.

How to Batch?

Rough idea to bypass the root of efficiency

- switch the role of decryption: let receiver decrypt ciphertexts then match the results with the indication string.

The idea is problematic since it is insecure even against a semi-honest receiver.

- receiver records the correspondence between y_i and $\mathsf{OKVS}(y_i) \rightsquigarrow$ receiver learns sender's private input x by simple look-up when $x \in Y$, rather than merely the fact that $x \in Y$.

We overcome this difficulty in two steps:

1. re-factor the functionality of OPRF to encryption and oblivious decryption functionality.
2. merge the oblivious decryption functionality and PEQT into a new functionality called vector oblivious decryption-then-matching (VODM) functionality.

Encryption Scheme

SKE/PKE scheme consists of three PPT algorithms:

- KeyGen(1^κ): output a secret key k or a keypair (pk, sk).
- Encrypt$(pk/k, m)$: output a ciphertext c of m.
- Decrypt$(sk/k, c)$: decrypt c to recover m.

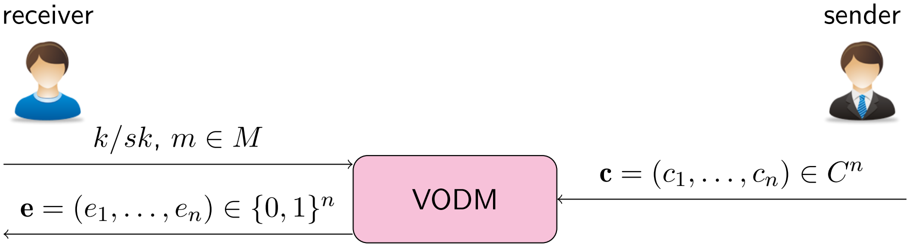
Single-message multi-ciphertext pseudorandomness. For any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, its advantage is negl$(\kappa)$.

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

- Single-message multi-ciphertext pseudorandomness is a mild property satisfied by most IND-CPA secure SKE/PKE, such as PRF-based SKE, ElGamal PKE based on DDH and Regev's PKE based on LWE.

Vector Oblivious Decrypt-then-Match (VODM)

VODM w.r.t. encryption scheme $(\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ is defined as below:



$$e_i = \begin{cases} 1 & \text{if Decrypt}(k/sk, c_i) = m \\ 0 & \text{if Decrypt}(k/sk, c_i) \neq m \end{cases}$$

mqRPMT from OKVS+Encryption+VODM

receiver

$Y = (y_1, \ldots, y_n)$

sender

$X = (x_1, \ldots, x_n)$

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

$s \xleftarrow{\mathsf{R}} M$

$\{c_i \leftarrow \mathsf{Encrypt}(k/pk, s)\}_{i=1}^n$

$\xrightarrow{D \leftarrow \mathsf{Encode}(\{y_i, c_i\}_{i=1}^n)}$

$k/sk,\ m \in M$ → VODM

VODM ← $\{s_i^* \leftarrow \mathsf{Decode}(D, x_i)\}_{i=1}^n)$

$\mathbf{e} = (e_1, \ldots, e_n) \in \{0,1\}^n$ ← VODM

Outline

Our Focus

Choose/Design appropriate primitives to realize the framework.

1. OKVS: any off-the-shelf OKVS is fine.
2. Encryption scheme: the ones satisfy single-message multi ciphertext pseudorandomness.
3. VODM: design w.r.t. the chosen encryption scheme

We only need to foucs on step 2 and 3.

mqRPMT from SKE and Generic 2PC

receiver

$Y = (y_1, \ldots, y_n)$

sender

$X = (x_1, \ldots, x_n)$

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

$s \xleftarrow{\mathsf{R}} M$, $\{z_i \leftarrow \mathsf{Encrypt}(k, s)\}_{i=1}^n$

$\xrightarrow{\mathsf{Encode}((y_1, z_1), \ldots, (y_n, z_n)) \to D}$ $\{\mathsf{Decode}(D, x_i) \to c_i\}_{i=1}^n$

$\xrightarrow{k,\ m \in M}$

2PC
GC or GMW

$\xleftarrow{\mathbf{c} = (c_1, \ldots, c_n)}$

$\xleftarrow{\mathbf{e} = (e_1, \ldots, e_n) \in \{0,1\}^n}$

$b_i \leftarrow \mathsf{Decrypt}(k, c_i) \stackrel{?}{=} m$

- SKE: choose LowMC for small circuit size
- generic 2PC: choose garbled circuit or GMW

mqRPMT from Rerandomizable PKE

receiver

$Y = (y_1, \ldots, y_n)$

sender

$X = (x_1, \ldots, x_n)$

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

$s \xleftarrow{\mathsf{R}} M$, $\{z_i \leftarrow \mathsf{Encrypt}(pk, s)\}_{i=1}^n$

$$m_i \leftarrow \mathsf{Decrypt}(sk, c_i') \xleftarrow[]{} \; \xrightarrow{\mathsf{Encode}((y_1, z_1), \ldots, (y_n, z_n)) \to D} \{\mathsf{Decode}(D, x_i) \to c_i\}_{i=1}^n$$

$$e_i := m_i \stackrel{?}{=} m \quad \xleftarrow{c_i' \leftarrow \mathsf{ReRand}(pk, c_i)}$$

- re-randomizable PKE: exponent ElGamal, Regev's PKE

mqRPMT from Rerandomizable PKE

receiver  $Y = (y_1, \ldots, y_n)$

sender  $X = (x_1, \ldots, x_n)$

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

$s \xleftarrow{\text{R}} M$, $\{z_i \leftarrow \mathsf{Encrypt}(pk, s)\}_{i=1}^n$

$\mathsf{ReRand}(pk, c) \to c'$

$\mathsf{Decrypt}(sk, c') = m = \mathsf{Decrypt}(sk, c)$

$c' \approx_s \mathsf{Encrypt}(pk, m)$

$$\mathsf{Encode}((y_1, z_1), \ldots, (y_n, z_n)) \to D \quad \xrightarrow{\hspace{2em}} \quad \{\mathsf{Decode}(D, x_i) \to c_i\}_{i=1}^n$$

$$m_i \leftarrow \mathsf{Decrypt}(sk, c_i') \quad \xleftarrow{\hspace{2em}} \quad c_i' \leftarrow \mathsf{ReRand}(pk, c_i)$$

$$e_i := m_i \stackrel{?}{=} m$$

- re-randomizable PKE: exponent ElGamal, Regev's PKE

mqRPMT from Rerandomizable PKE

receiver

$Y = (y_1, \ldots, y_n)$

sender

$X = (x_1, \ldots, x_n)$

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

$s \xleftarrow{\mathsf{R}} M$, $\{z_i \leftarrow \mathsf{Encrypt}(pk, s)\}_{i=1}^n$

$\mathsf{Encode}((y_1, z_1), \ldots, (y_n, z_n)) \rightarrow D$

$\{\mathsf{Decode}(D, x_i) \rightarrow c_i\}_{i=1}^n$

$c_i' \leftarrow \mathsf{ReRand}(pk, c_i)$

$m_i \leftarrow \mathsf{Decrypt}(sk, c_i')$

$e_i := m_i \stackrel{?}{=} m$

c_i' s.t. $e_i = 1$ do not leak information since receiver knows s
c_i' s.t. $e_i = 1$ do leak extra information
but such leakage is not harmful for PSU since receiver eventually learns $x_i \notin Y$

- re-randomizable PKE: exponent ElGamal, Regev's PKE

Outline

Retrospect of the Generic Framework

Previous two mqRPMT instantiations achieve linear complexity and enjoy good concrete efficiency.

Can we further generalize the framework?
Can we improve the efficiency of concrete instantiations?

High level idea underlying our mqRPMT design

1. receiver creates a membership relation R for his set Y s.t. $\mathsf{R}(x) = 1 \iff x \in Y$.
2. receiver encrypts elements in Y w.r.t. R and sends the "encoding" of resulting ciphertexts to the sender.
3. sender is able to retrieve the ciphertext of his elements.
4. perform oblivious decrypt-then-match

We realize the right encryption scheme needed is membership encryption (ME).

- ME for set X encrypts an element x into a ciphertext, which decrypts to "1" if $x \in X$ and to "0" (intuitively).

Membership Encryption

Definition 2 (Membership Encryption (Symmetric ME))

ME for set X consists of three PPT algorithms (with X as an implicit input):

- KeyGen(1^κ): outputs a key k.
- 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.
- Dec(k, c): outputs "1" indicates c is an encryption of some $x \in X$ and "0" if not.

Correctness. $\forall x \in X$, $\Pr[\mathsf{Dec}(k, c = \mathsf{Enc}(k, x)) = 1] = 1$, $k \leftarrow \mathsf{KeyGen}(1^\kappa)$.

Consistency. $\forall x \notin X$, $\Pr[\mathsf{Dec}(k, c) = 0] \geq 1 - \varepsilon(\kappa)$: $k \leftarrow \mathsf{KeyGen}(1^\kappa)$, $c \xleftarrow{\mathsf{R}} C$.

Multi-element pseudorandomness. $\forall$ distinct $x_1, \ldots, x_n \in X$

$$\{\mathsf{Enc}(k, x_i)\}_{i \in [n]} \approx_c U_{C^n},\ k \leftarrow \mathsf{KeyGen}(1^\kappa)$$

Symmetric ME naturally extends to the public-key setting:

- KeyGen outputs (pk, sk), in which pk is used to encrypt and sk is used to decrypt.

Generic Construction of ME

The essence of ME is to encrypt element's membership relation, rather than the element itself.

- Membership relation can be created by designing a mapping H from elements to X. Basically, there are two extreme cases of mapping.
  - lossy mapping: select a single indication string s as the characteristic of X, then map all elements to s, i.e., $\mathsf{H} : x_i \to s$.
  - injective mapping: select n indication strings s_i as the characteristic of X, then map elements to distinct indication strings, i.e., $\mathsf{H} : x_i \to s_i$.

We then present various constructions of ME by mixing encryption schemes and membership mapping.

ME from Probabilistic Encryption and Lossy Mapping

ME from probabilistic SKE and lossy mapping.

- $\mathsf{KeyGen}(1^\kappa)$: runs $\mathsf{SKE.KeyGen}(1^\kappa) \to k_{\mathsf{ske}}$, picks $s \xleftarrow{\mathsf{R}} M$, sets $\mathsf{H} : X \to s$, outputs $k = (k_{\mathsf{ske}}, \mathsf{H})$.
- $\mathsf{Enc}(k, x)$: parses $k = (k_{\mathsf{ske}}, \mathsf{H})$, outputs $c \leftarrow \mathsf{SKE.Enc}(k_{\mathsf{ske}}, \mathsf{H}(x))$.
- $\mathsf{Dec}(k, c)$: parses $k = (k_{\mathsf{ske}}, \mathsf{H})$, outputs '1' iff $\mathsf{SKE.Dec}(k_{\mathsf{ske}}, c) = s$.

ME from probabilistic PKE and lossy mapping.

- $\mathsf{KeyGen}(1^\kappa)$: runs $\mathsf{PKE.KeyGen}(1^\kappa) \to (pk_{\mathsf{pke}}, sk_{\mathsf{ske}})$, picks $s \xleftarrow{\mathsf{R}} M$, sets $\mathsf{H} : X \to s$, outputs $pk = pk_{\mathsf{pke}}$ and $sk = (sk_{\mathsf{pke}}, \mathsf{H})$.
- $\mathsf{Enc}(pk, x)$: parses $pk = pk_{\mathsf{pke}}$, outputs $c \leftarrow \mathsf{PKE.Enc}(pk_{\mathsf{pke}}, \mathsf{H}(x))$.
- $\mathsf{Dec}(sk, c)$: parses $sk = (sk_{\mathsf{pke}}, \mathsf{H})$, outputs '1' iff $\mathsf{PKE.Dec}(sk_{\mathsf{pke}}, c) = s$.

Lemma. If SKE/PKE satisfies single-message multi-ciphertext pseudorandomness, then the ME construction satisfies multi-element pseudorandomness with consistency error $1/|M|$.

Discussion

The above ME constructions are exactly the backbones of two instantiations of mqRPMT.

ME requires multi-element pseudorandomness

- the use of lossy mapping inherently stipulates that the accompanying encryption schemes must be probabilistic to satisfy single-message multi-ciphertext pseudorandomness
  $\rightsquigarrow$ ciphertext expansion is unavoidable
  $\Rightarrow$ the size of OKVS increases

> Observation: if adopting injective mapping, then ME can be built from deterministic encryption schemes satisfying multi-message multi-ciphertext pseudorandomness.

ME from Deterministic Encryption and Injective Mapping

ME from deterministic SKE and injective mapping.

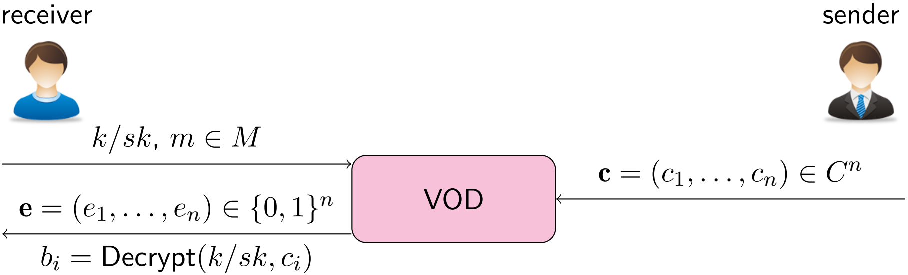
- KeyGen(1^κ): picks $k_{\mathsf{ske}} \xleftarrow{\mathsf{R}} K$, picks $s \xleftarrow{\mathsf{R}} M$, sets $\mathsf{H} : x_i \rightarrow i$, outputs $sk = (k_{\mathsf{ske}}, \mathsf{H})$.
- Enc(k, x): outputs $c \leftarrow \mathsf{SKE.Enc}(k_{\mathsf{ske}}, \mathsf{H}(x))$.
- Dec(sk, c): outputs '1' iff $\mathsf{SKE.Dec}(k_{\mathsf{ske}}, c) \in [n]$, where $n = |X|$.

Lemma: If SKE/PKE satisfies multi-message multi-ciphertext pseudorandomness, then the ME construction satisfies multi-element pseudorandomness with consistency error $n/|M|$.

- SKE candidate: PRP-based construction such as AES $\rightsquigarrow$ compact ciphertext
- PKE candidate: unclear for the time being (deterministic PKE?)

Vector Oblivious Decrypt

Since the decryption result of ME is only 1-bit to indicate membership, thus the accompanying VODM can be simplified to VOD.



Outline

Performance

n	Protocol	Comm. (MB)					Running time (s)															
		$\mathcal{R}$		$\mathcal{S}$		total	LAN				1Gbps				100Mbps				10Mbps			
							$T=1$		$T=8$		$T=1$		$T=8$		$T=1$		$T=8$		$T=1$		$T=8$	
		setup	online	setup	online		setup	online	setup	online	setup	online	setup	online	setup	online	setup	online	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

- communication: $3.7 - 14.8\times$ reduction depending on set sizes
- running time: $1.2 - 12\times$ speed-up depending on network enviroments and set sizes

Thanks for Your Attention!

Any Questions?

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

code: http://github.com/alibaba-edu/mpc4j
eprint: https://eprint.iacr.org/2022/358

Reference I

[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. In *USENIX Security 2023*, pages 301–318, 2023.

[BS05] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 236–252. Springer, 2005.

[CHL22] Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. Sok: Oblivious pseudorandom functions. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022*, pages 625–646. IEEE, 2022.

[DC17] Alex Davidson and Carlos Cid. An efficient toolkit for computing private set operations. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017*, volume 10343 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2017.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *CCS 2013*, pages 789–800, 2013.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005.

[Fri07] Keith B. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007*, volume 4521 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2007.

Reference II

[GMR+21] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In *Public-Key Cryptography - PKC 2021*, volume 12711 of *Lecture Notes in Computer Science*, pages 591–617. Springer, 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 *Advances in Cryptology - CRYPTO 2021*, volume 12826 of *Lecture Notes in Computer Science*, pages 395–425. Springer, 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 *IEEE Cybersecurity Development, SecDev 2016*, pages 75–76. IEEE Computer Society, 2016.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[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 2022*, 2022.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 20–31. ACM, 1988.

[KRTW19] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. Scalable private set union from symmetric-key techniques. In *Advances in Cryptology - ASIACRYPT 2019*, volume 11922 of *Lecture Notes in Computer Science*, pages 636–666. Springer, 2019.

Reference III

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2005.

[LV04] Arjen K. Lenstra and Tim Voss. Information security risk assessment, aggregation, and mitigation. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 391–401. Springer, 2004.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from paxos: Fast, malicious private set intersection. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 12106 of *Lecture Notes in Computer Science*, pages 739–767. Springer, 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 515–530. USENIX Association, 2015.

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. 2005. `http://eprint.iacr.org/2005/187`.

[RR22] Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In *ACM CCS 2022*, 2022.