Optimization in MPC:

Faster Private Set Intersection based on OT Extension

Zhi Xing

Background

- Private set intersection (PSI)
- Semi-honest adversary model
- Random oracle model
- Oblivious transfer (OT)

Private set intersection

- Two parties P_1 and P_2
- P_1 holds set X, P_2 holds set Y
- Identify $X \cap Y$ without revealing information about elements outside the intersection

Semi-honest adversary model

- The adversary tries to learn as much information as possible
- But it is not able to deviate from the protocol steps

Random oracle model

- A black box that responds to every unique query with a truly random response chosen uniformly from its output domain
- If a query is repeated, it responds the same way every time

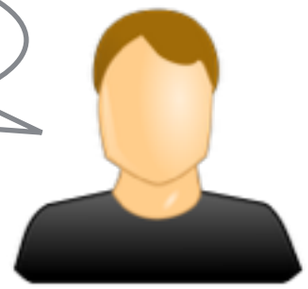
Oblivious transfer (OT)

- The sender wants to transfer k out of N messages to the receiver
- The receiver picks k of the N messages according to a N-bit choice vector
- The sender doesn't know which messages are picked
- The receiver doesn't know the messages other than the ones that are picked

1-out-of-2 OT

send one of m_0, m_1

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



1. generates RSA key (N, e, d) and random values x_0, x_1

2. sends N, e, x_0, x_1

3. generates a random value k

4. sends $v = (x_b + k^e) \bmod N$

5. calculates $k_0 = (v - x_0)^d \bmod N$ and $k_1 = (v - x_1)^d \bmod N$

6. sends $m_0' = m_0 + k_0$ and $m_1' = m_1 + k_1$

7. gets $m_b = m_b' - k$

OT extension

- $\binom{2}{1} - OT_l^m$ (*m* invocations of 1-out-of-2 OT on *l*-bit strings)

  requires 3*m* public-key operations

- 1-out-of-2 OT extension reduces OT_l^m to OT_κ^κ

- Compute the rest using symmetric cryptographic operations

- Computational complexity of OT is reduced => network bandwidth becomes the main bottleneck

send $(x_0^i, x_1^i) \in \{0,1\}^{2l}$ for $1 \le i \le m$

$\binom{2}{1} - OT_l^m$

receive $x_{b[i]}^i \in \{0,1\}^l$ for $1 \le i \le m, b \in \{0,1\}^m$

1. sends $(k_0^j, k_1^j) \in \{0,1\}^{2\kappa}$ for $1 \le j \le \kappa$

2. receives $k_{s[j]}^j$ for $1 \le j \le \kappa$
   using choice vector $s \in \{0,1\}^\kappa$

$\binom{2}{1} - OT_\kappa^\kappa$

3. chooses a random $m \times \kappa$ bit-matrix T
4. computes $v_0^j = t^j \oplus G(k_0^j)$ and $v_1^j = t^j \oplus G(k_1^j) \oplus b$
   where $t^j \in \{0,1\}^m$ denotes the j-th column of T
   and $G : \{0,1\}^\kappa \to \{0,1\}^m$ is a PRG
5. sends (v_0^j, v_1^j) for $1 \le j \le \kappa$
6. generates a $m \times \kappa$ bit-matrix Q
   as $q^j = v_{s[j]}^j \oplus G(k_{s[j]}^j)$
7. computes $y_0^i = x_0^i \oplus H(q_i)$ and $y_1^i = x_1^i \oplus H(q_i \oplus s)$
   where $q_i \in \{0,1\}^\kappa$ denotes the i-th row of Q
   and $H : \{0,1\}^\kappa \to \{0,1\}^l$ is a CRF
8. sends (y_0^i, y_1^i) for $1 \le i \le m$
9. receives $x_{b[i]}^i = y_{b[i]}^i \oplus H(t_i)$ for $1 \le i \le m$

Other OT extensions

- 1-out-of-*N* OT extensions
- Random OT extensions
  - Use basic OT (asymmetric cryptographic operations) to establish oblivious “symmetric keys”
  - Compute the rest of the protocol using symmetric cryptographic operations
  - Examples in the proposed protocol

Classification of PSI Protocols

- Naive solution
  - cryptographic hash functions
  - not secure if the input domain is small or has low entropy

- Public-key-based PSI
  - based on commutative property

- Circuit-based PSI
  - sort-compare-shuffle
  - Yao's garbled circuit protocol
  - Goldreich-Micali-Wigderson (GMW) protocol

- OT-based PSI
  - Bloom-filter-based
  - Set inclusion with hashing

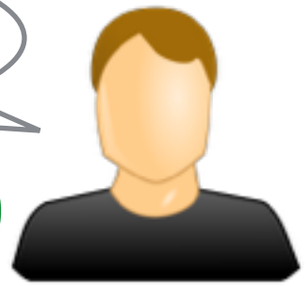
The proposed OT-based PSI

- Basic private equality test (PEQT) protocol
- Private set inclusion protocol
- The proposed OT-based PSI protocol

Basic PEQT protocol

check if my σ-bit $x = y$

P_1



check if my σ-bit $y = x$

P_2

1. uses $\binom{2}{1} - OT_l^\sigma$ to transfer uniformly distributed random l-bit strings (s_0^i, s_1^i) for $1 \leq i \leq \sigma$

random $\binom{2}{1} - OT$ extension

2. uses y as choice vector and obtains $s^i_{y[i]}$ for $1 \leq i \leq \sigma$

3. computes and sends $m_{P_1} = \bigoplus_{i=1}^{\sigma} s^i_{x[i]}$

4. computes $m_{P_2} = \bigoplus_{i=1}^{\sigma} s^i_{y[i]}$ and decides $x = y$ iff $m_{P1} = m_{P2}$

The basic PEQT protocol

- Can be improved by using base-N representation of inputs and $\binom{N}{1} - OT$ extension

- If $N = 2^{\eta}$, x and y are σ-bit, let $t = \sigma / \eta$, then x and y will be cut into t blocks of η bits:
  $x = x[1] \| x[2] \| \ldots \| x[t]$, $y = y[1] \| y[2] \| \ldots \| y[t]$

- $\binom{N}{1} - OT_l^t$ can be used to send l-bit strings $(s_0^i, \ldots, s_{N-1}^i)$

Private set inclusion protocol

- Check whether y equals to any of the values in $X = \{x_1, \ldots, x_{n_1}\}$
- $\binom{N}{1} - OT^t_{n_1 l}$ is used to transfer $n_1 l$-bit strings
- In the i-th transfer, N random strings $(s^i_0, \ldots, s^i_{N-1})$ are sent and $s^i_{y[i]}$ is received
- For $1 \leq i \leq t$, $s^i_{y[i]}$ is divided into n_1 substrings of length l, one for each element in X

Private set inclusion protocol

- P_1 computes and sends $m_{P_1}[j] = \oplus_{i=1}^{t} s^i_{x_j[i]}[j]$ for $1 \leq j \leq n_1$ which are compared to P_2's $m_{P_2} = \oplus_{i=1}^{t} s^i_{y[i]}$
- Can be improved by sending only the random seeds of length l, and generating the rest $(n_1 - 1)l$ part using PRG
- Same amount of data transfer as for single PEQT

OT-based PSI protocol

- Run private set inclusion protocol for each $y \in Y$

Hashing Schemes

- Pair-wise comparison for sets of size n has $O(n^2)$ complexity
- This can be improved by hashing elements into bins using a publicly known hash function, so only elements mapped to the same bin are compared
- If use n bins for n elements, the number of elements in each bin is $O(1)$ and the overall complexity is $O(n)$
- However, number of elements in each bin shouldn't be revealed, so dummy items are added to each bin, which increase complexity a little bit

Hashing Schemes

- Simple hashing
  - one hash function
  - ignores collisions

- Balanced allocations
  - two hash functions
  - maps an element to the less populated bin

- Cuckoo hashing
  - two hash functions
  - maps an element to a bin and relocates the other element if there's a collision

Evaluations

Type	Symm. Security Parameter κ	80-bit					128-bit					Asymptotic
	Set Size n	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	
Public-Key	**DH-based FFC [32]**	**0.4**	**1.6**	**6.2**	**24.7**	**98.8**	4.8	19.1	76.5	306.0	1,224.1	$2n$ asym
	DH-based ECC [32]	0.7	2.8	11.0	44.1	177.5	**1.6**	**6.5**	**26.1**	**104.2**	**416.2**	$2n$ asym
	RSA-based [16]	0.5	2.0	7.9	31.3	124.9	7.7	31.0	124.3	497.2	1,982.1	$2n$ asym
Circuit [30]	**Yao [8,31]**	1.2	5.7	27.7	128.2	-	**1.6**	**6.3**	**28.4**	**129.1**	-	$12n\sigma \log_2 n$ sym
	GMW [1]	1.9	8.6	35.2	161.9	806.5	2.6	12.8	58.9	276.4	1,304.2	$30n\sigma \log_2 n$ sym
	Vector-MT GMW §3.2	**1.2**	**5.1**	**21.2**	**100.3**	**462.7**	1.9	7.8	36.5	168.9	762.4	$18n\sigma \log_2 n$ sym
OT	Garbled Bloom Filter [19]	0.3	0.9	3.9	16.1	71.9	0.6	2.0	8.5	37.1	154.4	$4.32n\kappa$ sym
	Random Garbled Bloom Filter §4.3	0.15	0.5	2.0	8.1	34.3	0.27	1.0	4.1	16.7	67.6	$3.6n\kappa$ sym
	Set Inclusion §5 + Hashing §6	**0.13**	**0.2**	**0.8**	**3.3**	**13.5**	**0.26**	**0.3**	**0.9**	**3.7**	**13.8**	$0.75n\sigma$ sym

Table 5: Runtimes in seconds for PSI protocols with one thread over Gigabit LAN ($\sigma = 32$: bit size of set elements, asym: public-key operations, sym: symmetric cryptographic operations).

Evaluations

Type	Symm. Security Parameter κ	80-bit					128-bit					Asymptotic
	Set Size n	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	
Public-Key	DH-based FFC [32]	0.4	1.5	6.0	24.0	96.0	1.1	4.5	18.0	72.0	288.0	$3n\rho$
	DH-based ECC [32]	**0.1**	**0.2**	**1.0**	**3.8**	**15.0**	**0.1**	**0.4**	**1.5**	**6.0**	**24.0**	$3n\varphi$
	RSA-based [16]	0.3	1.1	4.3	17.3	69.0	0.8	3.1	12.5	50.0	200.0	$2n\rho + 2n\kappa$
Circuit [30]	Yao [8,31]	28.1	135.0	630.0	2,880.0	12,960.0	45.0	216.0	1,008.0	4,608.0	20,736.0	$9n\kappa\sigma\log_2 n$
	GMW [1]	31.3	150.0	700.0	3,200.0	14,400.0	50.0	240.0	1,120.0	5,120.0	23,040.0	$10n\kappa\sigma\log_2 n$
	Vector-MT GMW §3.2	**18.8**	**90.0**	**420.0**	**1,920.0**	**8,640.0**	**30.0**	**144.0**	**672.0**	**3,072.0**	**13,824.0**	$6n\kappa\sigma\log_2 n$
OT	Garbled Bloom Filter [19]	3.4	13.5	54.0	216.0	864.0	7.6	30.2	121.0	483.8	1,935.4	$2.88n\kappa(\kappa+\lambda)$
	Random GBF §4.3	1.1	4.5	18.1	72.6	290.4	2.9	11.6	46.2	184.9	739.7	$1.44n\kappa^2 + n(\lambda + 2\log_2 n)$
	Set Inclusion §5 + Hashing §6	**0.2**	**0.8**	**3.3**	**13.4**	**54.3**	**0.3**	**1.2**	**4.8**	**19.4**	**78.3**	$0.5n\kappa\sigma + 6n(\lambda + 2\log_2 n)$

Table 6: Communication complexity in MB for PSI protocols. ($\sigma = 32$: bit size of set elements, security parameters $\kappa, \lambda, \rho, \varphi$ as defined in §2.1). Numbers are computed from the asymptotic complexity given in the last column.

Evaluations

Type	**Network (Bandwidth (Mbit/s) / Latency (ms))**	**Gigabit LAN** (1,000 / 0.2)	**802.11g WiFi** (54 / 0.2)	**Intra-country WAN** (25 / 10)	**Inter-country WAN** (10 / 50)	**HSDPA** (3.6 / 500)
Public-Key	DH-based ECC [32]	104.2	104.8	107.6	111.8	115.9
Circuit [30]	Yao [8,31]	129.1	779.5	1,735.5	4,631.8	11,658.6
	Vector-MT GMW §3.2	168.9 (11.3)	370.5 (18.1)	770.4 (27.5)	1,936.5 (67.2)	5,310.9 (170.2)
OT	Random Garbled Bloom Filter §4.3	16.6	37.2	70.8	164.9	445.0
	Set Inclusion §5 + Hashing §6	**3.7**	**5.0**	**8.8**	**22.8**	**77.5**

Table 7: Runtimes in seconds for PSI protocols with one thread in different network scenarios for $n = 2^{16}$ elements, $\sigma = 32$: bit size of elements, and $\kappa = 128$-bit security (cf. Tab. 2); online time for Vector-MT GMW in ().

Q & A

Thank you.