Private Matching

A survey of private set intersection protocols over the decades for future compute

Prashanthi Ramachandran

CS2952L (Fall 2022): Final Presentation

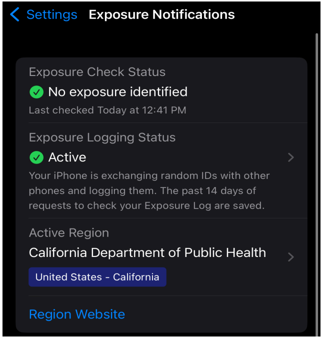
What is PSI?

- Parties Alice and Bob hold sets $A = \{a_1, a_2, \cdots, a_n\}$ and $B = \{b_1, b_2, \cdots, b_m\}$ respectively
- Goal: to design a protocol that computes $A \cap B$
- Constraints:
  - No trusted third party
  - Bob should not learn anything about all $a_i \notin A \cap B$ that he did not already know
  - ...and vice versa

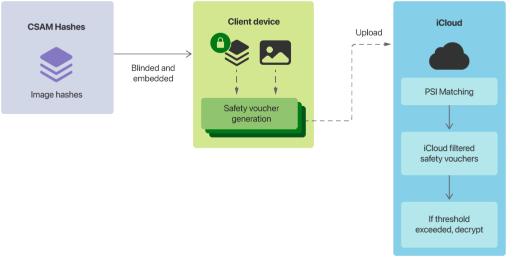
Applications

- Contact Tracing
  - Spread of infections
  - Ad targeting
- Password Monitoring
- Remote diagnostics
- Combining data for useful aggregation
  - Aggregate (sum/cardinality/average) of an attribute
  - Machine Learning or Federated Learning
- Community Discovery

Social Implications



(a) COVID-19 Exposure Tracking



(b) Apple CSAM Detection

Figure: Real-life applications of PSI

Overview

- PSI protocols over the decade
- Efficiency, threat models, and security for downstream computation
- PSI for PPML and PPFL

Major cryptographic primitives

1. Decisional Diffie-Hellman
2. Oblivious Polynomial Evaluation
3. Generic MPC
4. Oblivious Transfer (OT) and Bloom filters
5. OT Encoding

Decisional Diffie-Hellman (DDH)

The DDH Assumption

- Consider a (multiplicative) cyclic group G of order q, and with generator g
- The DDH assumption states that, given g^a and g^b for uniformly and independently chosen $a, b \in \mathbb{Z}_q$, the value g^{ab} "looks like" a random element in G.
- Formally, the following two distributions are computationally indistinguishable:
  - (g^a, g^b, g^{ab}), where a and b are randomly and independently chosen from $\mathbb{Z}_q$
  - (g^a, g^b, g^c), where a, b, and c are randomly and independently chosen from $\mathbb{Z}_q$

Basic Idea (1986)

- Let parties A and B possess secrets S_A and S_B resp. and share a common prime P
- Let S_A and S_B also be generators of the group Z_P
- A chooses a secret number M_A and B chooses M_B

A and B can match their secrets as follows:

1. A sends $S_A^{M_A}$ to B
2. B sends $S_B^{M_B}$ to A
3. A sends $S_B^{M_B M_A}$ to B
4. B sends $S_A^{M_A M_B}$ to A

Is this maliciously secure?

Basic Idea (1986)

1. A sends $S_A^{M_A}$ to B
2. B sends $S_B^{M_B}$ to A
3. A sends $S_B^{M_B M_A}$ to B
4. B sends $S_A^{M_A M_B}$ to A

Is this maliciously secure?
No. But we can use digital signatures!
What about fairness?

Extending to set intersection (1999)

- Let's say party A holds the set $A = x_1, \cdots, x_n$ and a secret value $a \in \mathbb{Z}_p$ and party B holds $B = y_1, \cdots, y_m$ and a secret value $b \in \mathbb{Z}_p$
- A and B can match their sets as follows:
  1. A sends $H(x_1)^a, \cdots, H(x_n)^a \mod p$ (after randomly permuting) to B
  2. B sends $H(y_1)^b, \cdots, H(y_n)^b \mod p$ (after randomly permuting) to A
  3. A sends $H(y_1)^{ab}, \cdots, H(y_n)^{ab} \mod p$ (after randomly permuting) to B
  4. B sends $H(x_1)^{ba}, \cdots, H(x_n)^{ba} \mod p$ (after randomly permuting) to A
  5. Each party can count their matches
- Cardinality is revealed!
  - Conscious optimization choice.
- ZKP for malicious security

More recent work

- DDH and RSA assumption
- Malicious security and client-server model
- Semi-honest PSI-CA using OT

Oblivious Polynomial Evaluation

Basic Idea

1. Parties A and B hold sets $\{a_1, \cdots, a_n\}$ and $\{b_1, \cdots, b_n\}$ respectively.
2. Parties A and B pick random polynomials P_A and P_B of degree n.
3. A obliviously obtains $\{P_B(a_i)\}_{i=1}^{n}$ and computes $\{P_A(a_i) + P_B(a_i)\}_{i=1}^{n}$.
4. Similarly B computes $\{P_A(b_i) + P_B(b_i)\}_{i=1}^{n}$.
5. This way, for a pair a_i and b_j, such that $a_i = b_j$, the sum results in equality of items in the respective computed lists.

n **oblivious evaluations on polynomials of degree** n**!**

Optimizing the basic idea for one-to-many PSI

1. Party A holds an input element x and B holds set $\{b_1, \cdots, b_n\}$
2. B chooses n linear polynomials $\{P_1, \cdots, P_n\}$
3. For every $i \in \{1, \cdots, n\}$, A obliviously computes $P_i(x)$
4. B computes and publishes $\{P_1(b_1), \cdots, P_n(b_n)\}$, permuted randomly
5. A checks if any of the values she computed matches the list B published

Many-to-many from one-to-many

- B computes n^2 polynomials instead of n: $P^B_{i,j}$ for $1 \leq i,j \leq n$
- A obliviously obtains n^2 linear polynomials $P^B_{i,j}(a_i)$ for $1 \leq i,j \leq n$
- A also computes $P^A_{i,j}$ for $1 \leq i,j \leq n$
- B obliviously obtains $P^A_{i,j}(b_j)$ for $1 \leq i,j \leq n$

In this case, if $a_i = b_j$,

$$P^A_{i,j}(a_i) + P^B_{i,j}(a_i) = P^A_{i,j}(b_j) + P^B_{i,j}(b_j)$$

Computation: $O(n)$
Communication: $O(n^2)$

Further optimization

- Using homomorphic encryption, communication can be brought down to $O(n)$ at the cost of slightly higher computation costs (2004).
- Using secret-sharing and Elgamal encryption, this was further optimized (2012, 2019).

Generic MPC

Generic MPC

- Assumption that generic MPC is expensive, not scalable, impractical
- Homomorphic encryption
- Public-key encryption
- Through the late 2000s and 2010s, several efficient maliciously and semi-honest secure PSI protocols based on Yao's GC were proposed.
- **Significant contribution**: Aguiar and Blanton (2012) - huge framework with a variety of private set operations
  - n parties where $n > 2$
  - Very compatible with downstream computation

OT and Bloom Filter

OT and Bloom Filter

- Recent line of work
- Scalability and better privacy guarantees
- **Bloom Filter**
  - linear runtime
  - compact datastructure
  - array of m bits that represent a set S of at most n elements
  - a set of k uniform hash functions $\{h_0, h_1, \cdots, h_{k-1}\}$
  - they uniformly map each element to an index number in $[0, m-1]$

Bloom filter

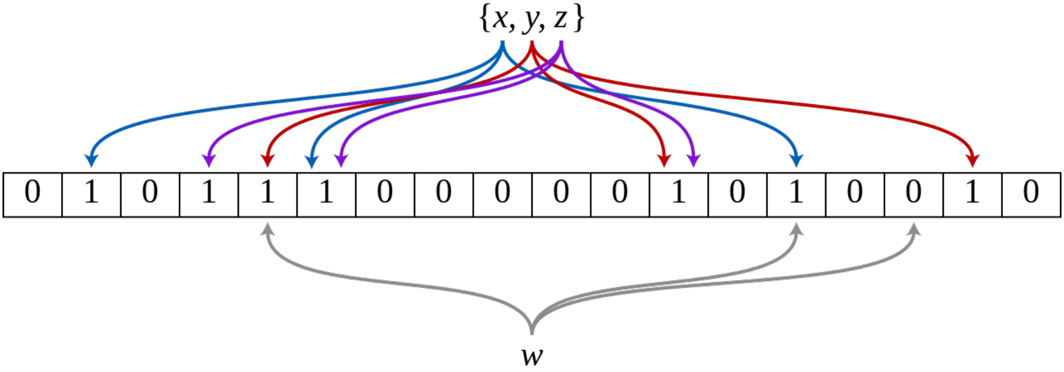


Figure: Bloom filter

Insert or check for element in Bloom filter

- **Insert x:**
  - BF_S is initially all zeroes
  - To insert $x \in S$, set $BF_S(h_i(x)) = 1$ for $\forall i \in \{0, \cdots, k-1\}$
- **Check for membership of y:**
  - If all the bits corresponding to the indices $h_i(y)$ in the Bloom filter are set to 1, i.e., $(BF_S(h_i(y)) == 1)\ \forall 0 \leq i \leq k-1$, then it is possible that $y \in S$
  - If not *all* the bits corresponding to the indices $h_i(y)$ are equal to 1, then we can be sure that $y \notin S$

The probability that a false positive occurs is negligible in k

Garbled Bloom filter

- Each of the m element of the array stores a λ-bit string
- **Insert** $x \in S$**:**
  - To insert $x \in S$, split x into k λ-bit shares (XOR-based SS)
  - We compute k indices for x similar to the usual Bloom filter approach and store one λ-bit share of x in each index
- **Check for membership of** y**:**
  - Get k indices corresponding to y
  - Collect all bitstrings in those locations and check to see if the XOR of all of the bitstrings is equal to y
  - If so, $y \in S$. If not, $y \notin S$.
  - False positive probability is negligible in security parameter λ

Basic Idea (2013)

- The client generates a (m, n, k, H)-BF that encodes its set C and the server generates a (m, n, k, λ, H)-GBF that encodes its private set S
- The client and server participate in a simple OT protocol, where the server sends m pairs of $\lambda-$bit strings $(x_{i,0}, x_{i,1})$ where $x_{i,0}$ is a randomly string and $x_{i,1}$ is $GBF_S[i]$
- As a result, from $0 \leq i \leq m-1$, if $BF_C[i] = 0$, the client gets a random string, but if it is 1, it gets the share stored in $GBF_S[i]$
- C obtains a new GBF, $GBF_{C \cap S}$ as the result of this computation
- The client can then query this new GBF against each one its elements to privately check for membership

Optimizations and SOTA

- In 2016, Rindal and Rosulek pointed out a major security flaw in the maliciously secure protocol based on GBF.
- They also provide full simulation-based security proof for the Bloom-filter-based PSI paradigm.
- New maliciously secure protocols upto 8-75$\times$ faster.

OT Encoding

OTs for PSI (private equality)

- Earliest work: Fagin et. al, 1996 (digital envelopes)
- Semi-honest security

Basic Idea

- Ron and Moshe hold inputs (bitstrings) $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ resp.
- They each sample n pairs of random numbers ($2n$ in total) uniformly from the range $[0, 2^k - 1]$
- Let us denote Ron's random pairs by $(R_1^0, R_1^1), (R_2^0, R_2^1), \cdots, (R_n^0, R_n^1)$ and those of Moche's by $(M_1^0, M_1^1), (M_2^0, M_2^1) \cdots, (M_n^0, M_n^1)$
- Ron then computes:

$$T_R = \sum_{i=1}^{n} R_i^{x_i} \mod 2^k$$

and Moshe computes:

$$T_M = \sum_{i=1}^{n} M_i^{y_i} \mod 2^k$$

Basic Idea (cont.)

- Moshe obliviously obtains those random values sampled by Ron that correspond to each bit from his n-bit input y
- Ron obliviously obtains those random values sampled by Moshe that correspond to each bit from his n-bit input x
- Ron computes:

$$S_R = T_R + \sum_{i=1}^{n} M_i^{x_i} \mod 2^k = \sum_{i=1}^{n} (R_i^{x_i} + M_i^{x_i}) \mod 2^k$$

and Moshe computes:

$$S_M = T_M + \sum_{i=1}^{n} R_i^{y_i} \mod 2^k = \sum_{i=1}^{n} (M_i^{y_i} + R_i^{y_i}) \mod 2^k$$

Basic Idea (cont.)

- Given that Ron and Moche are semi-honest, if $x = y$, then $S_M = T_M$.
- Small error prob.: 2^{-k}.

Optimizations and SOTA

- Several recent works for private set equality/intersection based on OT extension from 1-out-of-2 to 1-out-of-n.

2014 highly-efficient and novel OT-based PSI protocol that builds on top of private equality with very low computation costs and $O(n \log n)$ communication cost.

2017 further optimization of the malicious-secure protocol by approximately $12\times$ in the standard and random-oracle models.

Privacy-Preserving Machine Learning

- Let's say there is a hospital that wants to train a machine learning model for the detection of brain cancer
- A single hospital may not have enough data to train a good model
- ***Need to combine data from different hospitals!***

Combining data from various sources

- Challenging: due to privacy issues, legal constraints, and potential competitive disadvantage
- **Types:**
  - Horizontally
  - Vertically

PPML frameworks

- Work with secure outsourced computation (SOC)
- All stakeholders secret share their data among different servers
- Horizontal combination

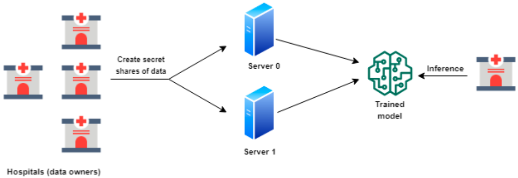


Figure: Secure outsourced computation in PPML

Vertically combining data

- Can they privately match their records?
- How can they use the combined attributes of the matching records to train a model while making sure no party learns anything more than the final model?
- Can private matching be done efficiently?
- How do the input and the output to the machine learning model look?
- Can we use other machine learning techniques and run them with an added layer of privacy to solve this problem?
- Can we use existing highly-efficient secure MPC protocols or cryptographic primitives to make this happen?

Solutions

- PS3I protocol from Private Matching for Compute (Facebook, 2020): highly compatible with Mohassel and Zhang (SecureML (2019))
- Aguiar and Blanton (2012): $n > 2$ parties and SS input-output
- Mohassel, Rindal, and Rosulek, 2019 work with 2-out-of-3 additively secret shared inputs and allows us to perform various SQL-like operations on data from various sources $\rightarrow$ highly compatible with Patra and Suresh, 2020 (BLAZE)
- Hardy et. al 2017 emonstrate the learning of a federated machine learning model where the data is split *vertically* among different data sources and only one data owner has the knowledge of the target variable
  - privacy-preserving entity resolution
  - federated logistic regression