Private Matching

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

Prashanthi Ramachandran

I. Introduction

Private set intersection (PSI) is a common technique in cryptography that deals with the problem of two parties holding sets of items that they wish to compute the intersection of. Formally, if Alice and Bob hold sets $A = \{a_1, a_2, \cdots, a_n\}$ and $B = \{b_1, b_2, \cdots, b_m\}$ respectively, the goal is to design a protocol that computes $A \cap B$. This protocol should ensure that for every $a_i \notin B$, then Bob should not learn anything about it (and vice versa). In other words, Bob should not learn anything from the protocol that he did not already know about elements in A that are not in $A \cap B$ (and vice versa). There also exist private protocols that perform associated set operations such as PSI-CA, PSDiff, PSI-Sum, etc. that compute $|A \cap B|$, $A - B$, and $\sum(A \cap B)$ respectively.

PSI has a variety of applications in the real world such as contact tracing, password monitoring, remote diagnostics, etc. Contact tracing involves tracking the people/devices a person have recently been in close contact with. This information could help companies with ad-targeting and even help us keep track of the spread of an infection, such as COVID-19. PSI allows hospitals/medical institutions or banks and insurance companies to collaborate and combine their data. This would allow these institutions to enrich their data and thereby provide better diagnosis or detect fraudulent transactions respectively. One of the most recent usage of this technique is in Apple's CSAM detection. They compute PSI on the user's private photos and encrypted versions of a database of CSAM images. Apple does not learn anything except for the neural hashes of the intersecting items and the neural hashes do not reveal anything about the items themselves or other items outside of the intersection. Recent works such as [1], [2], [3], [4] are highly efficient and practical for deployment.

In many scenarios, the parties may wish to securely aggregate the intersection data without revealing the data or the identifiers of elements in the intersection set [5]. This aggregation may be something simple—such as the average age of people in the intersection—or something complex—such as a machine learning model that computes whether genes in a certain person's DNA may be associated with the risk of a certain rare health condition. The aggregation in both these examples operate with data of highly sensitive nature. Revealing the intersection in the clear may be a bad idea and also unnecessary, considering that anyway the aggregation is only concerned with the final output. It is useful, therefore, to study two-party private intersection protocols that enable downstream computation without revealing anything about individual elements in the intersection set.

One such crucial downstream task is privacy-preserving machine learning (PPML) which solves the following problem: d data owners wish to collaboratively and securely train a machine learning model such that no party involved in the training or inference learns anything about the dataset in its original form. This is an emerging area in secure multiparty computation that allows secure aggregation of private data of parties without revealing individual data points in the clear. The work in this area is extensive [6], [7], [8], [9], [10], [11], [12] and highly optimized, making it practical for deployment. However, all of these works assume that the data from various sources has already been combined and pre-processed. Without a framework to join different databases, it is unclear how one can train a privacy-preserving ML model keeping privacy and efficiency in mind. Furthermore, the added privacy demand would make compute-intensive machine learning algorithms even more expensive.

Existing protocols for PSI and the downstream task of securely and privately training machine learning models may be incompatible, causing the combined protocol to be highly inefficient. Works such as [13] highly optimize private set intersection at the cost of some leakage. In this project, I will study various existing PSI frameworks and break them down by the cryptographic primitives they use. Further, I will also analyze them keeping the PPML/FL usecase in mind in terms of threat models, security, and efficiency.

II. Private Set Intersection

Over the decades, several PSI protocols have been proposed that use a diverse set of cryptographic primitives. Many of these protocols also perform PSI-C (the cardinality of the intersection of the two private sets) or PSI-sum (the sum of the items in the intersection) and other aggregations. In this section, I have attempted to categorize some PSI protocols (and some PSI-C and PSI-sum protocols) over the years by the cryptographic technique used, the input-output format, no. of parties involved (and the nature of these parties), and their threat models. These have also been summarized in Table I.

A. Diffie-Hellman

Some of the earliest papers such as [14] and [15] were based on the DDH assumption apart from others. [14] builds on the following basic idea. 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. Now, 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

The biggest drawbacks of using the above basic algorithm is that it places a lot of trust on both the parties. Party B could easily send the value it received in Step 3 to party A and make it seem like their values matched, but in reality, the values may not have matched for B's original input. In [14], they use digital signatures to establish the identity of the user and the validity of the messages exchange in steps 3 and 4. The protocol is maliciously secure owing to its usage of digital signatures, but it does not ensure fairness (if party B terminates after Step 3) and does not protect against bad inputs.

The PSI protocol proposed by [15] also builds on top of this idea, but additionally uses a hash function to extend this protocol to multiple items (set of inputs). This protocol ends up revealing the cardinality of PSI, but the authors argue that this was a conscious optimization choice. 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

If Alice and Bob don't randomly permute their sets in the above protocol, then they will be able to learn the common elements. However, this will enable the parties to learn the exact positions of those matches in the other party's original list. They also add a layer of zero knowledge proofs to this protocol to make it maliciously secure.

One major thing to note about [15] is that it assumes a slight variant of the PSI problem we established. Their motivation in this paper was to help the problem of community discovery for good recommendations and ad-targeting while protecting the privacy of members of the community by using shared preferences among them. They use non-interactive oblivious transfer by Bellare and Micali as a primitive. Anyone can post a question (or challenge) Q which is a yes/no question (and can be generalized to a multiple-choice question). Through the security guarantees of the above cryptographic primitives, they ensure that if a person answered "yes", they won't be able to read messages meant for people who answered "no" and vice-versa. They also provide guarantees for authenticating users so that they do not join both sides of the debate using digital signatures. Further, they provide insights into what can be done if merely shared preferences are not valuable enough for the community.

There has also been recent work in PSI that uses DDH including [16] and [17]. [16] additionally depends on RSA assumption and ZKP and is designed as a client-server model. As a result, while it is maliciously secure, fairness guarantees do not hold here, because in the client-server model, only the client wishes to compute the intersection. In the final step, the client is guaranteed an output, as long as the server is able to give a valid zero knowledge proof. On the other hand, [16] works with the semi-honest security model and computes PSI-C as $|A \cap B|$ and PSI-sum as $\sum(A \cap B)$ if A and B are the sets that each party holds respectively. It also uses random OT and Bloom filter as primitives.

B. Oblivious Polynomial Evaluation

The earliest paper that proposes oblivious polynomial evaluation to compute PSI is [18]. This paper proposes protocols to compute one-to-many intersection and extends that to compute list intersection (many-to-many). The core idea behind this protocol is as follows:

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 $a_i = b_j$ results in equality of items in the respective computed lists.

Note that this protocol requires each party to perform n oblivious evaluations on polynomials of degree n. The authors propose a one-to-many intersection protocol that requires the parties to perform n oblivious evaluations on linear polynomials. The above protocol is redesigned as follows:

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

Note that the computation and communication overheads for the above variant is linear in n. This paper also extends the one-to-many protocol to list intersection by requiring B to compute n^2 polynomials instead of n, $P_{i,j}^B$ for $1 \leq i, j \leq n$ and A obliviously computes n^2 linear polynomials $P_{i,j}^B(a_i)$ for $1 \leq i, j \leq n$. A and B do the converse as computing $P_{i,j}^A$ for $1 \leq i, j \leq n$ and B obliviously computes $P_{i,j}^A(b_i)$ for $1 \leq i, j \leq n$. In this case, if $a_i = b_j$, $P_{i,j}^A(a_i) + P_{i,j}^B(a_i) = P_{i,j}^A(b_i) + P_{i,j}^B(b_i)$. This protocol has a computation complexity of n evaluations, but a communication complexity of $O(n^2)$.

A later work [19] further decreases the complexity by achieving a computation cost of $O(n \log \log n)$ and a communication cost of $O(n)$ in both semi-honest and malicious settings. In the semi-honest setting, they require the client $\mathcal{C}$ to define a polynomial P whose roots are her inputs:

$$P(y) = (x_1 - y) \cdot (x_2 - y) \cdot \cdots , (x_n - y)$$

The client sends the server $\mathcal{S}$ homomorphic encryptions of the coefficients of the above polynomial. $\mathcal{S}$ then computes the encryption of $(r \cdot P(y) + y)$ using the homomorphic properties of the encryption scheme. As a natural result, $Enc(r \cdot P(y) + y)$ (where r is a freshly sampled random value) is equal to an encryption of the element itself, if it is part of the intersection set, and random, otherwise. There are certain clever techniques a malicious adversary can use to derive more information about the intersection set given the above protocol. Further, this paper provides a modified protocol that guarantees malicious security for the same setting. Other recent works such as [20] and [21] build on top of this protocol as a primitive. The former optimizes this protocol by using secret-sharing and Elgamal encryption. The latter further optimizes this protocol to a communication complexity of $O(mk^2 \log^2 n + kn)$ and a computation complexity of $O(mnk + mk^2 \log^2 n)$, where m and n are the sizes of the sets of the server and the client respectively and k is a security parameter.

C. *Generic MPC*

In the literature for private set intersection, solutions involving homomorphic encryption or public key cryptography have been introduced and optimized extensively. This have been due to the apparent assumption that generic MPC protocol may be incompatible or impractical for set operation on a large scale. Two major contributions in the literature have actually contested this assumption by proposing and highly optimizing generic secure multiparty computation primitives to solve this problem, namely [22] and [23]. Other papers that work with generic MPC to compute PSI include [24], [16], and [25] (all maliciously secure) and those that compute PSI-CA include [26], [27], [28], [29], and [30] (semi-honest secure).

[23] proposes 3 different semi-honest secure protocols for PSI based on Yao's garbled circuits. Yao's garbled circuit protocol takes in bit inputs and performs bit-wise operations on them depending on the computation. In this paper, they proposed a Bitwise-AND protocol, a pairwise private compare protocol ($\theta(n^2)$), and sort-compare-shuffle protocols ($\theta(n \log n)$). They describe how to use these foundational protocols to compute the set intersection of two parties with sets of bit string elements for inputs. The idea behind this work is have each party sort their input lists individually. Then, they privately merge their sorted lists into one big sorted list. Further, they obliviously compare adjacent elements using their pairwise private compare protocol. This paper assumes that the input sets do not contain any duplicate elements.

[22], on the other hand, is a huge framework with a variety of capabilities. This work performs secure multiparty computation (works with parties > 2) to compute multiple set operations, including PSI, PSU (private set union), PSDiff (private set difference), PER (private element reduction), PSR (private subset relationship), PSI-CA, PSU-CA, PSDiff-CA, PER-CA (cardinality), PMI (private multiset intersection), PMU (private multiset union), PMDiff (private multiset difference), PSE (private set equality), PSuR (superset relationship), PMI-CA, PMU-CA, and PMDiff-CA.

They use cryptographic building blocks, such as oblivious sorting, comparisons, and prefix AND. This is a unique piece of work that works with $n > 2$ parties for PSI (and associated operations) computation. This framework is highly compatible with secure outsourced computation and further compute (PPML, for instance) as each private value is secret-shared between all the n parties, $P_1, \cdots, P_n$ at every stage. All these protocols have a communication and computation complexity of $O(m \log m)$ for sets or multisets of size m.

D. *Oblivious Transfer and Bloom Filter*

This approach is fairly recent in the cryptography world and is primarily motivated by the need for efficiency and scalability as databases increase in size. In [31], they introduce *oblivious bloom filters* for enhancing the scalability and the privacy guarantees of the PSI protocol. Bloom filters have a linear runtime and easily parallelizable operations. A Bloom filter is a compact datastructure that can be used for testing probablistic set membership. It is essentially an array of m bits that represent a set S of at most n elements. It is composed of a set of k uniform hash functions $\{h_0, h_1, \cdots, h_{k-1}\}$. These functions uniformly map each element to an index number in $[0, m-1]$.

To insert an element $x \in S$ into an initially empty (all zeroes) Bloom filter, we hash the element using the k hash functions and obtain k indices in the range $[0, m-1]$. In the Bloom filter BF_S, we set the bits at all these indices to 1, i.e., $BF_S(h_i(x)) = 1 \ \forall 0 \leq i \leq k-1$. Similarly, to check if an element y is in S, we check to see if all the bits corresponding to the indices $h_i(y)$ in the Bloom filter are set to 1, i.e., $(BF_S(h_i(y)) == 1) \ \forall 0 \leq i \leq k-1$. If not *all* the bits corresponding to the indices $h_i(y)$ are equal to 1, then we can be sure that $y \notin S$. In other words, the probability of a false negative is 0. However, because of the possibility of collisions in the Bloom filter for multiple elements in the set, if *all* the bits corresponding to the indices $h_i(x)$ for an element x are equal to 1, it is *possible* that $x \in S$, but we cannot be sure. Owing to some theoretical guarantees, the authors claim that the probability that a false positive occurs is negligible in k.

In this paper, they introduce a variant of Bloom filters known as *Garbled Bloom filters* (GBF). While there is very little functional difference between a regular Bloom filter and a GBF, at a low-level, the difference is that a GBF uses $\lambda-$bit strings (where λ is a security parameter). To insert an element $x \in S$ in a GBF, they split x into k $\lambda-$bit shares (using an XOR-based secret sharing scheme). As was standard, we also break x down into k index numbers and store one $\lambda-$bit share each in $h_i(x)$. To check the membership of an element y in S, we collect all the bitstrings in the k locations corresponding to $h_i(y)$ and XOR them to see if they reconstruct y. If they do, we can be sure that $y \in S$. If they do not, we would know that $y \notin S$. This way the false positive and false negative probabilities for GBF are negligible in λ.

The idea behind oblivious Bloom Intersection protocol is as follows: 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.

[32] point out a major bug in the malicious-secure protocol of [31] and provide full simulation-based security proof for the Bloom-filter-based PSI paradigm. They also focus on malicious security and propose a new optimized protocol (8-75× faster) through several techniques including the removal of polynomial interpolation completely. [17] further improve upon these protocols using a combination of DDH computational assumption-based techniques (like some older work) and random OT and encrypted Bloom filters to compute PSI-CA in the semi-honest 2-party setting.

E. Oblivious Transfer Encoding

This basic approach to using OTs for PSI (private equality) was first proposed in 1996 by Fagin et. al [33]. They attempt to solve the private comparison problem. Their solutions to private equality are most relevant for our purposes. They propose semi-honest adversary secure a solution to compare two n-bit values values $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ using 1-out-of-2 OT.

They require the two parties involved, Ron and Moshe with inputs $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ resp. to 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:

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

Then, Moshe obliviously obtains those random values sampled by Ron that correspond to each bit from his n-bit input y. Similarly, Ron obliviously obtains those random values sampled by Moshe that correspond to each bit from his n-bit input x. Now, 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:

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

Given that Ron and Moche are semi-honest, if $x = y$, then $S_M = T_M$. This way they can check for the equality of two private numbers. There is a small error probability (2^{-k}), i.e., the probability that Ron and Moshe conclude that $x = y$ since $S_M = T_M$, but actually $x \neq y$, but that is negligible for a sufficiently large and positive k. However, this protocol will not work in the malicious adversary setting. This private equality protocol can be extended to multiple elements of a set if we can use an OT extension protocol.

The authors of [1] not only provide a detailed comparison of some the state-of-the-art PSI protocols at the time, but also propose several optimizations, particularly in the circuit- and Bloom filter-based protocols. They achieve this by incorporating techniques from recently improved OT extension protocols. They also propose a 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. [34] further optimizes the malicious-secure protocol by approximately $12\times$ in the standard and random-oracle models.

Technique	Reference	Year	No. of parties	Primitives	Input	Output	Threat Model
DDH	[14]	1986	2	DDH, digital signatures	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$A \cap B$	Malicious
	[15]	1999	2	DDH, ZKP	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$\|A \cap B\|$	Malicious
	[16]	2010	2 (client-server)	DDH, ZKP, RSA assump.	Client holds C = $\{(c_1, \sigma_1), \cdots, (c_v, \sigma_v)\}$ Server holds S = $\{s_1, \cdots, s_w\}$ where σ_i is a signature	C outputs $C \cap U$	Malicious
	[17]	2019	2	DDH	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$\|A \cap B\|$ and $\sum(A \cap B)$	Semi-honest
	[5]	2020	2	add. homomorphic enc., random oracle model	Private-ID: Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$ PS3I: Alice holds A = $\{(c_1, v_{c,1}), \cdots, (c_n, v_{c,n})\}$ Bob holds B = $\{(v_1, v_{p,1}), \cdots, (v_n, v_{p,n})\}$	Private-ID: $\|A \cap B\|$ PS3I: A and B get additive secret shares of $\|A \cap B\|$	Semi-honest
Oblivious Polynomial Evaluation	[18]	1999	2	OT	Client holds C = $\{c\}$ Server holds S = $\{s_1, \cdots, s_w\}$ (or) Client holds C = $\{c_1, \cdots, c_v\}$ Server holds S = $\{s_1, \cdots, s_w\}$	$0/1(c \in S)$ (or) $C \cap S$	Malicious
	[19]	2004	2 (client-server)	FHE	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_m\}$	$A \cap B$	Malicious, semi-honest
	[20]	2009	2 (client-server)	FHE, Shamir SS, Elgamal enc.	Client holds C = $\{c_v, \cdots, c_v\}$ Server holds S = $\{s_1, \cdots, s_w\}$	C outputs $C \cap S$	Malicious
	[21]	2019	2 (client-server)	FHE, OLE, crypto commitments	Client holds C = $\{c_1, \cdots, c_v\}$ Server holds S = $\{s_1, \cdots, s_w\}$	C outputs $C \cap S$	Malicious
Generic MPC	[22]	2011	n > 2	Ob. sorting, Bitwise pvt op., OPRFs, Shamir SS	$P_1, \cdots, P_n$ hold secret shares of the inputs	Various set operations (see text)	Malicious, semi-honest
	[23]	2012	2	Garbled circuits, Ob. sorting, Bitwise pvt op., OT ext.	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$A \cap B$	Semi-honest
OT and Bloom Filters	[31]	2013	2 (client-server)	Shamir SS OT	Client holds C = $\{c_1, \cdots, c_v\}$ Server holds S = $\{s_1, \cdots, s_w\}$	C outputs $C \cap S$	Semi-honest
	[32]	2016	2 (client-server)	Shamir SS OT	Client holds C = $\{c_1, \cdots, c_v\}$ Server holds S = $\{s_1, \cdots, s_w\}$	C outputs $C \cap S$	Malicious
	[17]	2019	2	DDH Random OT Enc. Bloom filters	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$\|A \cap B\|$	Semi-honest
OT Encoding	[33]	1996	2	1-out-of-2 OT	Alice holds an n-bit value x = $x_1, x_2, \cdots, x_n$ Bob holds an n-bit value y = $y_1, y_2, \cdots, y_n$	$x == y$	Semi-honest
	[1]	2014	2	OT ext. Random oracle GMW	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_m\}$	$A \cap B$	Malicious, semi-honest
	[34]	2017	2	[1]	Alice holds A = $\{x_1, \cdots, x_n\}$ Bob holds B = $\{y_1, \cdots, y_n\}$	$A \cap B$	Malicious
Mixture	[35]	2020	3	2-out-of-3 SS Cuckoo Hashing	P_0, P_1, P_2 hold 2-out-of-3 secret shares of databases X and Y	P_0, P_1, P_2 obtain 2-out-of-3 secret shares of the result of SQL agg. op. on X and Y	Semi-honest

TABLE I: Summary of private set intersection protocols over the years

III. Privacy-Preserving Machine Learning

Let us 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. Thus, to train a good machine learning model, data from multiple hospitals is required. However, combining data from different sources remains a challenge due to multiple reasons. Privacy preserving machine learning via secure Multiparty Computation (MPC) allows many entities to train a model on joint data without revealing any information about the data. Privacy preserving models should be built keeping in mind the following factors:

1) **Privacy concerns and regulations:** Medical institutions are legally and ethically bound by the security and privacy of senstive patients data. The privacy-preserving model should ensure the security and privacy of patient data.
2) **Competitive advantage:** Since multiple hospitals need to be involved, this is an important factor that plays in when data is procured for the training process. The privacy-preserving model should be trained such that it does not give any hospital a competitive advantage over another hospital.
3) **Data sovereignty and jurisdiction:** Every hospital is subjected to the laws within the nation with respect to patient data and how they choose to use it.

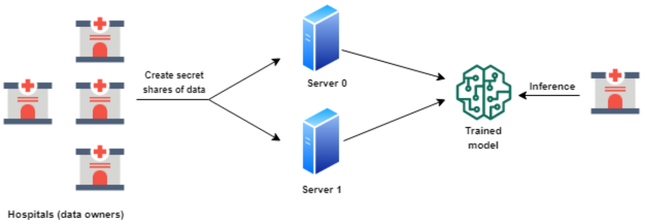

Fig. 1: Secure Outsourced Computation in Privacy-Preserving Machine Learning

There are primarily two ways data can be combined before we can train the models:

- **Horizontally:** If party P_0 and party P_1 hold datasets D_0 and D_1 respectively, we wish to combine the datasets such that the new dataset $D = D_0 + D_1$.
- **Vertically:** If party P_0 and party P_1 hold datasets D_0 and D_1 respectively, we wish to get the inner join of D_0 and D_1.

Several recent PPML frameworks including [6], [7], [9], [36], [37], [38], [8] are highly optimized and practical for deployment. Most of these papers work with a technique called secure outsourced computation for preserving the privacy of the sensitive data. Secure outsourced computation (SOC) allows to outsource computation to a set of powerful cloud-servers and avail their services on a pay-per-use basis, while guaranteeing the privacy of training and classification. As Figure 1 shows, each data owner “secret-shares" their data among two or more servers that will perform secure multiparty computation on the combined set of data to train a model. This way, they privately combined their data horizontally for training.

However, vertically combining data (inner join) from different sources still remains a big challenge because this would require the parties to determine the intersection of their datasets before combining on the data points in the intersection.

A. PSI for PPML

Let us consider a scenario where there are two institutions that collected different types of data from people. For instance, let us say the Nike Fitness app (with a dataset N) and Apple’s Health app (with a dataset A) wish to jointly train a machine learning model to improve fitness predictions for their users. However, Nike and Apple may each be bounded by different user privacy policies that dictate how they use data. Now, there are several questions that arise:

- 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?

The PS3I protocol described in [5] attempts to solve exactly this problem. They allow parties to privately compute the intersection of their private sets by outputting a secret share of the elements in the intersection. This secret-shared data can then be easily plugged into any of the PPML frameworks described above. A good combination of frameworks would be [5] and [9] as the former’s output format (additive secret shares) match the input format of the latter, and both these frameworks work with the two-party semi-honest setting.

[22] is another PSI protocol that we have discussed in Section 2.3 that works with similar techniques. At every stage in this protocol, the values are secret-shared among the n participating parties and this is highly compatible with the recent PPML frameworks discussed above. Further, this paper also works with generic MPC protocols such as garbled circuits which make them compatible with any further private computations.

[35] works with 2-out-of-3 additively secret shared inputs and allows us to perform various SQL-like operations on data from various sources. It also outputs a 2-out-of-3 additively secret shared database to each of the 3 servers, which is highly compatible with the input-output formats of [6]. The private bit/single element matching protocols described in [18], [33] also pertain very well to the Private Compare protocol described in [11].

A very interesting line of work for this problem is that of [39] that does not bother combining data from various sources through the approaches I have described above. In this work, they demonstrate 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. Furthermore, in their setting, the entities are not even linked across data sources. They describe a three-party end-to-end framework for privacy-preserving entity resolution (to match identifiers across vertically-split data) and federated logistic regression in the cross-feature space.

IV. Conclusion

In this project, I have studied and compared various existing PSI frameworks through the years. I have categorized them into five different types of frameworks based on the cryptographic technique/assumptions they primarily rely on. I have also intensively described each cryptographic primitive or assumption that these PSI protocols are based on to set up the contextual background for the state-of-the-art. I have also described the setting, threat model, and security-privacy guarantees of these protocols. Further, I have reported the input and output formats of these protocols so that the distinctions between these protocols through the years is clear. Moreover, I have analyzed the efficiency of the protocols through the years and given a thorough account of the optimizations over time.

Finally, I described Privacy-Preserving Machine Learning and set up the ground for PSI in this context. I then described the existing PSI protocols that can be adapted in existing PPML frameworks and demonstrated the need for more compatible and efficient PSI protocols for further compute.

References

[1] B. Pinkas, T. Schneider, and M. Zohner, “Faster private set intersection based on OT extension,” in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 797–812. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas

[2] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, “Phasing: Private set intersection using permutation-based hashing,” in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 515–530. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas

[3] B. Pinkas, T. Schneider, and M. Zohner, “Scalable private set intersection based on ot extension,” *ACM Trans. Priv. Secur.*, vol. 21, no. 2, jan 2018. [Online]. Available: https://doi.org/10.1145/3154794

[4] P. Mohassel, P. Rindal, and M. Rosulek, “Fast database joins and psi for secret shared data,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1271–1287. [Online]. Available: https://doi.org/10.1145/3372297.3423358

[5] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, “Private matching for compute,” Cryptology ePrint Archive, Paper 2020/599, 2020, https://eprint.iacr.org/2020/599. [Online]. Available: https://eprint.iacr.org/2020/599

[6] A. Patra and A. Suresh, “BLAZE: Blazing Fast Privacy-Preserving Machine Learning,” 2020, https://eprint.iacr.org/2020/042.

[7] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, “ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction,” 2019, https://eprint.iacr.org/2019/429.

[8] P. Mohassel and P. Rindal, “Aby3: A mixed protocol framework for machine learning,” Cryptology ePrint Archive, Paper 2018/403, 2018, https://eprint.iacr.org/2018/403. [Online]. Available: https://eprint.iacr.org/2018/403

[9] P. Mohassel and Y. Zhang, “SecureML: A System for Scalable Privacy-Preserving Machine Learning,” 2017, https://eprint.iacr.org/2017/396.

[10] Facebook, “Crypten: A research tool for secure machine learning in pytorch.”

[11] S. Wagh, D. Gupta, and N. Chandran, “SecureNN: Efficient and Private Neural Network Training,” 2018, https://eprint.iacr.org/2018/442.

[12] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, “Private machine learning in tensorflow using secure computation,” *arXiv preprint arXiv:1810.08130*, 2018.

[13] A. Groce, P. Rindal, and M. Rosulek, “Cheaper private set intersection via differentially private leakage,” Cryptology ePrint Archive, Paper 2019/239, 2019, https://eprint.iacr.org/2019/239. [Online]. Available: https://eprint.iacr.org/2019/239

[14] C. Meadows, “A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party,” in *1986 IEEE Symposium on Security and Privacy*, 1986, pp. 134–134.

[15] B. A. Huberman, M. Franklin, and T. Hogg, “Enhancing privacy and trust in electronic communities,” in *Proceedings of the 1st ACM Conference on Electronic Commerce*, ser. EC ’99. New York, NY, USA: Association for Computing Machinery, 1999, p. 78–86. [Online]. Available: https://doi.org/10.1145/336992.337012

[16] E. D. Cristofaro, J. Kim, and G. Tsudik, “Linear-complexity private set intersection protocols secure in malicious model,” Cryptology ePrint Archive, Paper 2010/469, 2010, https://eprint.iacr.org/2010/469. [Online]. Available: https://eprint.iacr.org/2010/469

[17] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, and M. Yung, “On deploying secure computing: Private intersection-sum-with-cardinality,” Cryptology ePrint Archive, Paper 2019/723, 2019, https://eprint.iacr.org/2019/723. [Online]. Available: https://eprint.iacr.org/2019/723

[18] M. Naor and B. Pinkas, “Oblivious transfer and polynomial evaluation,” in *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, ser. STOC ’99. New York, NY, USA: Association for Computing Machinery, 1999, p. 245–254. [Online]. Available: https://doi.org/10.1145/301250.301312

[19] M. J. Freedman, K. Nissim, and B. Pinkas, “Efficient private matching and set intersection,” in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3027. Springer, 2004, pp. 1–19. [Online]. Available: https://iacr.org/archive/eurocrypt2004/30270001/pm-eurocrypt04-lncs.pdf

[20] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, “Efficient robust private set intersection,” *Int. J. Appl. Cryptol.*, vol. 2, no. 4, p. 289–303, jul 2012. [Online]. Available: https://doi.org/10.1504/IJACT.2012.048080

[21] S. Ghosh and T. Nilges, “An algebraic approach to maliciously secure private set intersection,” vol. 11478, pp. 154–185, 2019.

[22] M. Blanton and E. Aguiar, “Private and oblivious set and multiset operations,” Cryptology ePrint Archive, Paper 2011/464, 2011, https://eprint.iacr.org/2011/464. [Online]. Available: https://eprint.iacr.org/2011/464

[23] Y. Huang, D. Evans, and J. Katz, “Private set intersection: Are garbled circuits better than custom protocols?” in *Network and Distributed System Security Symposium*, 2012.

[24] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, “Secure efficient multiparty computing of multivariate polynomials and applications,” vol. 6715, 06 2011, pp. 130–146.

[25] C. Hazay and K. Nissim, “Efficient set operations in the presence of malicious adversaries,” Cryptology ePrint Archive, Paper 2009/594, 2009, https://eprint.iacr.org/2009/594. [Online]. Available: https://eprint.iacr.org/2009/594

[26] L. Kissner and D. Song, “Privacy-preserving set operations,” in *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, ser. CRYPTO’05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 241–257. [Online]. Available: https://doi.org/10.1007/11535218_15

[27] J. Vaidya and C. Clifton, “Secure set intersection cardinality with application to association rule mining,” *J. Comput. Secur.*, vol. 13, no. 4, p. 593–622, jul 2005.

[28] G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhury, and C. P. Rangan, "Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security," in *Cryptology and Network Security*, 2009.

[29] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," Cryptology ePrint Archive, Paper 2011/141, 2011, https://eprint.iacr.org/2011/141. [Online]. Available: https://eprint.iacr.org/2011/141

[30] Y. Sang and H. Shen, "Efficient and secure protocols for privacy-preserving set operations," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, nov 2009. [Online]. Available: https://doi.org/10.1145/1609956.1609965

[31] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: An efficient and scalable protocol," Cryptology ePrint Archive, Paper 2013/515, 2013, https://eprint.iacr.org/2013/515. [Online]. Available: https://eprint.iacr.org/2013/515

[32] P. Rindal and M. Rosulek, "Improved private set intersection against malicious adversaries," Cryptology ePrint Archive, Paper 2016/746, 2016, https://eprint.iacr.org/2016/746. [Online]. Available: https://eprint.iacr.org/2016/746

[33] R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it," *Commun. ACM*, vol. 39, no. 5, p. 77–85, may 1996. [Online]. Available: https://doi.org/10.1145/229459.229469

[34] P. Rindal and M. Rosulek, "Malicious-secure private set intersection via dual execution," Cryptology ePrint Archive, Paper 2017/769, 2017, https://eprint.iacr.org/2017/769. [Online]. Available: https://eprint.iacr.org/2017/769

[35] P. Mohassel, P. Rindal, and M. Rosulek, "Fast database joins and psi for secret shared data," Cryptology ePrint Archive, Paper 2019/518, 2019, https://eprint.iacr.org/2019/518. [Online]. Available: https://eprint.iacr.org/2019/518

[36] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 707–721.

[37] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "Aby2.0: Improved mixed-protocol secure two-party computation," Cryptology ePrint Archive, Paper 2020/1225, 2020, https://eprint.iacr.org/2020/1225. [Online]. Available: https://eprint.iacr.org/2020/1225

[38] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *arXiv preprint arXiv:2004.02229*, 2020.

[39] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.