IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Secure Multi-Party Computation for Machine Learning: A Survey

**IAN ZHOU[1],(Member, IEEE,), FARZAD TOFIGH[1],(Member, IEEE), MASSIMO PICCARDI[1],(Senior Member, IEEE), MEHRAN ABOLHASAN[1],(Senior Member, IEEE), DANIEL FRANKLIN[1],(Member, IEEE) and JUSTIN LIPMAN[1],(Senior Member, IEEE)**

[1]School of Electrical and Data Engineering, University of Technology Sydney, Sydney, NSW 2007 Australia

Corresponding author: Farzad Tofigh (e-mail: farzad.tofigh@ uts.edu.au).

The authors acknowledge the support of Food Agility CRC Ltd, funded under the Commonwealth Government CRC Program. The financial and in-kind support of Robert Bosch (Australia) Pty Ltd & Robert Bosch GmbH in completing this work is also acknowledged.

**ABSTRACT** Machine learning is a powerful technology for extracting information from data of diverse nature and origin. As its deployment increasingly depends on data from multiple entities, ensuring privacy for these contributors becomes paramount for the integrity and fairness of machine learning endeavors. This review looks into the recent advancements in secure multi-party computation (SMPC) for machine learning, a pivotal technology championing data privacy. We evaluate these applications from various aspects, including security models, requirements, system types, and service models, aligning with the IEEE's recommended practices for SMPC. Broadly, SMPC systems are divided into two categories: homomorphic-based systems, which facilitate computations on encrypted data, ensuring data remains confidential, and secret sharing-based systems, which disseminate data across parties in fragmented shares. Our literature analysis highlights certain gaps, such as security requisites, streamlined information exchange, incentive structures, data authenticity, and operational efficiency. Recognizing these challenges lead to envisioning a holistic SMPC protocol tailored for machine learning applications.

**INDEX TERMS** Multi-party computation, Machine learning, Federated learning, Data privacy, Cryptography, Protocols

## I. INTRODUCTION

IN an ever-advancing world increasingly saturated with data, recent technological developments such as the Internet of Things (IoT) [1], wireless sensor networks, cloud computing, and machine learning provide different means and methods to extract and process these data. Moreover, sensitive information can be derived from the analysis of data. Data and data analysis applications, collectively referred to as machine learning, have significantly impacted many different fields and industries [2]–[4]. The processing of data could involve multiple parties from different backgrounds for collaborative purposes. Organizations and individuals might want their data, method or model to remain private during collaborative operations. A recent 2022 report [5] from IBM has revealed limited data security in many organisations. 83% of these organisations had more than one data breach at an average cost of USD 4.35 million - increasing by 12.7% since 2020. The increasing cost reflects the importance of data security [6].

Secure Multi-Party Computation (SMPC) has gathered significant interest as a technological solution to data privacy and security concerns. SMPC represents a privacy-preserving approach that allows multiple parties to jointly compute the function of $(y_1, y_2, ..., y_i) \leftarrow f(x_1, x_2, ..., x_i)$, where each party $P_i$ provides input $x_i$ and computes $y_i$. Leveraging the power of cryptography, SMPC enables these parties to collectively determine a result from their individual input data without disclosing any of this data to the other participants. Thus, each party's data remains secure and private, yet a meaningful, collective computation can still take place.

While several survey papers on Secure Multi-Party Computation (SMPC) have been published [7]–[11], only a subset offers a comprehensive overview of the field. Furthermore, certain papers, such as [7], [8], fall short of providing a detailed analysis on each topic encompassed within the domain of SMPC. The discussion on future works is also limited. In [8], the authors focused on a problem-oriented approach of analysis. The authors reviewed and organized

the research according to the different problems that can be solved by SMPC. Specifically, the problems mentioned are preserving privacy in cooperative scientific computations, database query, intrusion detection, data mining, geometric computation, statistical analysis, and other common computing operations. Additionally, two-party examples are provided for all of the problems. However, the authors did not elaborate on the technical details of their solutions. In [8], the authors describe their work as: "*...a guideline for the researchers in other computation areas to think about their computation problem from this security perspective...*" Moreover, the discussion on future works is also general and there are no specific recommendations for the SMPC problems defined.

In a more recent work, [7], the authors provide a comprehensive overview on theoretical and practical aspects of SMPC. They offer an overview of the fundamentals of SMPC by defining threats and security requirements and models such as semi-honest adversary model, malicious adversary model, and cover adversary model. This work also outlines the building blocks of SMPC, including garbled circuits, oblivious transfer, and homomorphic encryption. It should be noted, however, that while Secret Sharing, Zero-Knowledge Proof, and Commitment Scheme are recognized as essential components of SMPC, a thorough exploration of these topics is absent. Additionally, the paper investigates machine learning applications within the framework of application-oriented SMPC. Yet, the scope of the review is relatively limited, with only four works undergoing detailed examination, and technical discussions remain somewhat restricted. The survey concludes with a general discussion on potential future developments [7].

The research contributions of Wang et al. [9] and Feng et al. [10] are noteworthy for their comprehensive review on secure multi-party computation protocols and techniques. Wang et al. [9] thoroughly discuss the classification of adversaries within the context of SMPC, suggesting that contemporary adversaries may be overestimated in their capabilities. To this end, they introduce a more restrained adversary model, one that is constrained by the prospective benefits of the executed attacks. Considering the context of rational parties, the paper introduces the concept of rational SMPC. Additionally, with the lens of game theory, the paper presents the works in rational secret sharing schemes, rational multiple function calculation, and rational Byzantine protocol. This perspective offers an alternative interpretation of SMPC where security measures need to account for strategic, rational behavior as well as outright malicious acts.

In [10], a comprehensive review of the building block technologies for SMPC is presented. The paper also includes a section on privacy-preserving machine learning methods and various protocols with different numbers of parties are compared. However, homomorphic encryption is only viewed as a helper method to secret sharing and garbled circuit-based SMPC systems. Systems utilizing homomorphic encryption as a major method to preserve security are not reviewed. Moreover, systems related to federated learning are not considered in [10].

This survey paper aims to provide a study of SMPC applications in the field of machine learning under IEEE recommended practice for secure multi-party computation [12]. The major contributions of this paper are:

1) Outline of machine learning service settings under the IEEE recommended practice for secure multi-party computation.
2) Analysis of recent homomorphic encryption-based SMPC application for machine learning.
3) Overview of recent usage of secret sharing for machine learning.
4) Identifying research gaps and future directions of SMPC for machine learning.

The rest of this paper is organized into: Section II provides background information about SMPC. This includes security models, security requirements, types of SMPC systems and building block technologies for machine learning SMPC applications. Section IV is preliminary information on machine learning, which includes the definition of user models and knowledge of federated learning. Sections V and VI analyze SMPC applications based on homomorphic encryption and secret sharing, respectively. Challenges and barriers of SMPC adoption and implementation guidelines are discussed in sections VIII and IX. Section VII outlines the potential security vulnerabilities of SMPC systems. Section X presents the limitations on the reviewed works followed by Section XII, the conclusion.

## II. SECURE MULTI-PARTY COMPUTATION (SMPC) FUNDAMENTALS

This section delineates the fundamentals of SMPC systems, emphasizing security models, pertinent requirements, and core technologies which underpin these systems. The types of SMPC systems that adhere to the specifications detailed by the IEEE standard are highlighted. Each subsection provides a comprehensive exposition on the core components of SMPC systems.

### A. SECURITY MODELS AND REQUIREMENTS

Secure Multi-Party Computation (SMPC) protocols serve to ensure the privacy and integrity of computations, which involve multiple participants. These protocols are designed to be robust against the intrusion of adversaries who aim to breach the privacy of the participants or disrupt the computation process. Consequently, security models play a pivotal role in defining the capabilities and intentions of potential adversaries.

Security models are often presented as adversary models, that define the behavior and potential threats posed by different kinds of attackers [13]. These models act as the primary foundation for developing security requirements that are intended to protect the SMPC system against such attacks, thereby preserving the privacy and ensuring the participation of honest users.

As presented in Table 1, adversaries can largely be categorized into three primary models [7], [9], [10], [12].

Semi-honest adversaries: These adversaries operate within the confines of the established protocol, ensuring a consistent execution of their roles. However, they will seize any opportunity to glean private information from other participants, given the chance. This model also allows for the possibility of collusion between corrupted parties to extract sensitive information.

Malicious adversaries: This model represents a more potent threat, as these adversaries are not constrained by the protocol's rules. They possess the capability to launching arbitrary attacks aimed at disrupting or breaching the protocol, consequently creating a significant risk to the integrity and privacy of the SMPC system.

Covert adversaries: Positioned as an intermediate model between the semi-honest and malicious adversaries, covert adversaries weigh the benefits and potential ramifications of their actions before launching an attack. This typically results in covert adversaries restricting themselves to less overt forms of attacks. They are commonly found in commercial and political settings, where the potential gain from a successful attack is high, but so too are the consequences of being caught.

The security requirements for an SMPC system are primarily derived from these adversary models. They form the basis of countermeasures designed to detect, prevent, and mitigate potential attacks while ensuring the smooth and secure operation of the SMPC system.

**TABLE 1.** Adversary Models. [7], [9], [10], [12]

| Type of Adversary | Description |
|---|---|
| Semi-honest | Adversaries follow the execution of protocol. However, they want to obtain private information of other parties. Collusion between corrupted parties are possible. |
| Malicious | Adversaries are not bounded by the protocol and will attempt to break the protocol. |
| Covert | Adversaries consider the gain of cheating, probability and loss for being caught. This model reflects most companies and institutions. |

Other than the adversary models, security can also be described by the number of corrupted parties in a scenario. The honest majority setting assumes that over half of the participants are honest. On the other hand, the dishonest majority setting assumes a scenario with greater or equal to half of the total participants being corrupted [10], [12]. The IEEE recommended practice for secure multi-party computation [12] further divides the honest majority setting into the Q2 condition with less than half of the participants corrupted and the Q3 condition with less than one-third of the participants corrupted.

### B. SECURITY REQUIREMENTS BASED ON ADVERSARY MODELS

In the context of the various adversary models, security requirements for SMPC systems are designed to ensure the integrity, confidentiality, and availability of data and computation processes. These requirements, tailored to combat the threats posed by semi-honest, malicious, and covert adversaries, aim to foster a secure computation environment. Here, we discuss these requirements with reference to each adversary model.

**Semi-honest:** Security requirements for combating semi-honest adversaries are mainly centered around privacy and correctness. The privacy requirement ensures that each participant learns nothing more than their designated output, limiting the amount of information semi-honest adversaries can extract from the computation process. Correctness requires the protocol to produce a correct output as long as honest participants are following the protocol, despite any eavesdropping attempts by semi-honest adversaries.

**Malicious:** To protect against malicious adversaries, who can deviate arbitrarily from the protocol, a more robust set of security requirements is necessary. Beyond privacy and correctness, security against malicious adversaries often necessitates a security property called robustness. This property ensures that the computation process can resist any attempts by malicious adversaries to alter the outcome or disrupt the computation. In addition, the requirement of verifiability is often imposed, enabling the system to detect if an adversary is deviating from the protocol and take appropriate actions, such as terminating the computation or excluding the adversary.

**Covert:** Security requirements against covert adversaries need to strike a balance between deterring unauthorized behavior and maintaining computational efficiency. One critical requirement is accountability, which ensures that any deviation from the protocol by a covert adversary can be detected with a high probability. If caught, proof of their misbehavior can be used to penalize them, creating a deterrent for such actions. Also, a fairness requirement may be included, ensuring that no participant can obtain their output before others, discouraging covert adversaries from gaining an unfair advantage.

These security requirements, when properly implemented, can provide a strong defense against adversaries in SMPC systems. They ensure the system can operate as expected, while preserving the privacy, integrity, and availability of the computation process and the associated data. However, achieving these security requirements often involves trade-offs in terms of computational and communication efficiency, which are critical considerations in the design of SMPC systems.

### C. TYPES OF SMPC SYSTEMS AS PER THE IEEE STANDARD

Secure Multi-Party Computation (SMPC) systems are defined by the IEEE standard [12] based on the level of se-

curity provided, the number of participants, and the nature of the computations involved. While there is a diverse range of SMPC systems, we'll focus on three common types in this section: threshold SMPC, general SMPC, and special-purpose SMPC.

Threshold SMPC: These systems allow a certain number (threshold) of participants to jointly compute a function without revealing their private inputs to each other. They provide privacy, correctness, and a guarantee of output but may not always meet fairness or independence of input requirements. Threshold SMPC systems are often used in cryptographic scenarios like key generation and management where a subset of participants is enough to perform the operation.

General SMPC: In contrast to threshold SMPC, general SMPC systems allow any number of participants to jointly compute a function, ensuring that no participant gains additional knowledge beyond their own input and the computed output. The security requirements include privacy, correctness, and potentially fairness and a guarantee of output. These systems are versatile and widely applicable, including scenarios like private set intersection and distributed machine learning.

Special-purpose SMPC: These systems are designed for specific applications or computations. They offer privacy and correctness but might vary in terms of other security requirements based on the application they are designed for. Examples include privacy-preserving data mining, secure auctions, and voting systems.

As for the common security requirements, a framework that ensures the correct functioning and security of SMPC systems is provided:

Privacy: Ensures that each participant's input is kept secret from the other participants. Correctness: Guarantees that the output of the computation is correct provided that the honest participants follow the protocol.

Fairness: Assures that no participant can get their output before others do. Guarantee of Output: Certifies that if the protocol starts, an output will be generated despite the behavior of adversarial participants.

Independence of Inputs: Emphasizes the importance of each participant's ability to independently select their inputs without interference from other participants. Probability to Catch Deviation: An optional requirement that encourages honest participants to identify those that violate the protocol.

Each of these security requirements plays a vital role in defining the behavior of an SMPC system. They ensure the system operates accurately and securely, providing robust defenses against potential adversaries while preserving the privacy and integrity of the computation process and the associated data. However, the trade-offs between these requirements, such as computational efficiency, communication overhead, and the level of provided security, are critical considerations when designing and implementing SMPC systems.

The common security requirements of SMPC systems are privacy, correctness, fairness, and guarantee of output [7], [12]. In [7], independence of input is included to emphasize the importance of private input information. In the IEEE recommended practice [12], the probability to catch deviation is a requirement that asks honest parties to identify corrupted parties that violate the protocol. This requirement is an optional requirement along with fairness and guarantee of output in IEEE recommended practice [7]. Privacy and correctness are instead fundamental requirements. Table 2 presents the definitions of the security requirements.

**TABLE 2.** Security Requirements. [7], [12]

| Requirement | Definition |
|---|---|
| Privacy | Each party must only obtain its own output and information formulated from its own input and output. |
| Correctness | The result received must be correct or lossless compared to the result obtained in plain-text computation. |
| Fairness | If the result is received by corrupt parties, then it should also be received by honest parties. |
| Guarantee of Output | Output delivery to honest parties should not be interrupted by corrupted parties. |
| Independence of Input | The input from a corrupted party should be independent from the input of any honest parties. |
| Probability to Catch Deviation | Honest parties should have a certain probability to catch any corrupted parties on violations of the protocol. |

### D. DEPLOYMENT MODES OF SMPC SYSTEMS

There are many ways to classify SMPC systems. This paper follows the IEEE standards [12], which classifies SMPC systems into three recommended deployment modes (Table 3). SMPC systems are classified according to the distribution or location of computation tasks. The first type of SMPC system is server-side systems. Server-side systems include data providers with limited computational resources. Each of the data providers shares its data with multiple non-colluding servers for further computation. Server-side systems are assumed to operate by the secret sharing scheme (described in the next subsection) to preserve data privacy when sharing data with computation servers. Peer-to-peer SMPC is the second type of SMPC system. Every data provider has enough computation resources for peer-to-peer systems to follow the SMPC protocols. Computation is organized by each of the data providers. Thus, the roles of each data provider could be homogeneous. Finally, in server-aided SMPC systems, data providers perform most of the computations. However, part of the computations (e.g., generation of Beaver triples [1], and weight aggregation in federated learning) is offloaded to a third-party server.

[1]A Beaver triple is a set of three random values (a, b, c) such that the relationship a * b = c holds. These values are generated by a trusted third party and then distributed to the parties involved in the computation.

**TABLE 3.** Types of SMPC Systems. [12]

| SMPC Type | Data Provider Operation | External Server Operation |
|---|---|---|
| Server-side | Data sharing to multiple non-colluding nodes. | Most of the SMPC operations. |
| Peer-to-peer | Most of the SMPC operations. | N/A |
| Server-aided | Most of the SMPC operations. | Some SMPC operation (e.g. generating Beaver triples) performed by an aiding server. |

### *E. BUILDING BLOCKS*

In [7], oblivious transfer, garbled circuit, commitment scheme, zero-knowledge proof, homomorphic encryption and secret sharing are considered as the building block technologies of SMPC. The works reviewed in this paper are mainly based on homomorphic encryption and secret sharing. Therefore, only information on homomorphic encryption and secret sharing is provided in this section.

#### 1) Homomorphic Encryption

Homomorphic encryption is a transformative cryptographic method that facilitates computations on encrypted data without requiring decryption [14]. This concept forms the backbone of many Secure Multi-Party Computation (SMPC) systems as it allows data to be manipulated in its encrypted state, hence preserving privacy throughout the computational process.

In this encryption paradigm, computations performed on ciphertexts yield a result that, once decrypted, matches the result of the same operations conducted on the plaintext. In other words, the encrypted output of a computation is essentially the encryption of the plaintext output. As such, both the input and output remain shielded by encryption throughout the process.

The extent of computation allowed on encrypted data gives rise to three distinct types of homomorphic encryption schemes [14]:

Partially Homomorphic Encryption (PHE): Supports unlimited repetitions of a single operation (either addition or multiplication) on ciphertexts.

Somewhat Homomorphic Encryption (SHE): Allows a limited number of both addition and multiplication operations on ciphertexts.

Fully Homomorphic Encryption (FHE): Enables an unlimited number of both addition and multiplication operations on ciphertexts.

Among various homomorphic encryption schemes, this paper discusses the ElGamal and Paillier schemes.

##### ElGamal encryption scheme

ElGamal encryption scheme, introduced by Taher Elgamal in 1984, is a partially homomorphic encryption scheme that supports unlimited multiplicative operations on ciphertexts [14], [15]. ElGamal encryption consists of three fundamental steps: key generation, encryption, and decryption. A unique aspect of the ElGamal scheme is the concept of proxy re-encryption [16], which enables the transformation of ciphertext encrypted under one key to ciphertext encrypted under a different key, using a proxy key. Key generation starts with a cyclic group $G$ of order $n$ created with a generator $g$. A random integer $x$ is chosen to compute $h = g^x$. In the end, the public key is $(G, n, g, h)$ and the private key is $x$. During encryption, a message $m$ is encrypted starting by choosing a random number $y$ from $1, 2, ..., n-1$. Then a pair of ciphertext is produced as $(c_1, c_2) = (g^y, mh^y)$. The ciphertext pair can be decrypted by the private key $x$ and the original message can be obtained by $m = c_2 c_1^{-x}$. ElGamal is a partially homomorphic encryption scheme that only supports multiplicative operations on encrypted pairs. A variant of the ElGamal scheme is proxy re-encryption [16]. It allows the conversion of an encrypted message to the ciphertext of the same message encrypted by another key. This conversion requires a proxy function and a proxy key generated from the secret keys before and after the conversion. Therefore, this algorithm assumes no collusion between the users and the proxy converter to preserve privacy. As a variant of the ElGamal scheme, proxy re-encryption also allows homomorphic multiplicative operations [16].

##### Paillier cryptosystem

The Paillier cryptosystem, proposed by Pascal Paillier in 1999, is another notable homomorphic encryption scheme [17]. Unlike ElGamal, the Paillier scheme is homomorphic over addition, meaning it allows for unlimited additions of ciphertexts. Similar to ElGamal, Paillier encryption also consists of key generation, encryption, and decryption steps. Large primes $p$ and $q$ are selected to fulfill the condition that $pq$ and $(p-1)(q-1)$ are coprime. Calculate $n = pq$ and the least common multiple of $(p-1)$ and $(q-1)$ as $\lambda$. Then, select a random integer $g$ from multiplicative subgroup of integers modulo $n^2$ and ensure $n$ and $L(g^\lambda \pmod{n^2})$ are coprime where the function $L(u) = (u-1)/n$. $(n, g)$ are the resulting public key and $(p, q)$ are the private key. The message $m$ is encrypted to obtain the ciphertext $c = g^m r^n \pmod{n^2}$. Decryption is performed by $m = L(c^\lambda \pmod{n^2})/L(g^\lambda \pmod{n^2}) \pmod{n}$. The Paillier scheme is homomorphic over additive and multiplicative operations [14].

In both schemes, the combination of these steps ensures that the data remains encrypted at all times, preserving privacy and preventing unauthorized access to sensitive information. Despite their distinct computational capabilities, both ElGamal and Paillier cryptosystems contribute significantly to maintaining privacy in multi-party computation environments.

#### 2) Homomorphic Proxy Re-Encryption (HPRE)

With the advancement of cloud technologies and machine learning, data privacy has become an increasingly significant

concern. Homomorphic Proxy Re-Encryption that combines the features of a homomorphic encryption scheme and proxy re-encryption scheme, offer a solution for this challenge [18], [19].

#### 3) Secret Sharing

Secret sharing is a crucial cryptographic technique employed in SMPC systems, enabling a secret to be divided amongst a group of parties in such a way that only specific subsets or a sufficient number of parties can reconstruct the original secret [20]. The most commonly used secret sharing methodologies in machine learning applications of SMPC are additive, Shamir's, and replicated secret sharing.

Additive secret sharing is a relatively straightforward technique [10]. In this approach, a secret $x$ is shared among $n$ parties by randomly assigning the first $n-1$ shares from a finite field $F$, having a field size of $p$ [21]. The final share is then calculated as $s_n = (x - \sum_{i=1}^{n-1} s_i) \pmod p$. The original secret can be reconstructed if all shares are available: $x = (\sum_{i=1}^{n} s_i) \pmod p$.

Shamir's secret sharing, proposed by Adi Shamir, adds a threshold concept to secret sharing [20]. Here, only a threshold number $t$ of $n$ shares is needed to reconstruct the secret. The process involves creating a polynomial $f$ of degree $k = t-1$ where $f(0) = x$. The remaining coefficients of $f$ are randomly selected from a finite field $F$, and each secret share $s_i$ is computed as $f(a_i)$, with $a_i \in F$. To reconstruct the original secret, Lagrange basis polynomials are computed and polynomial interpolation is used to reconstruct $f$ and extract the original secret $x$ from its constant portion.

Replicated secret sharing follows a slightly different approach, where a secret $x$ is additively split such that $\sum_{Q \in T} s_Q = x$ [22]. Here, $T$ comprises all possible combinations of $t-1$ parties, assuming a $t$ or more than $t$ parties can reconstruct the secret. Each party receives secret shares $s_Q$, where the party is not a member of $Q$. Every party has $\binom{n-1}{t-1}$ shares from $\binom{n}{t-1}$ shares. The secret reconstruction process is similar to additive sharing [22]. Parties of a minimum number $t$ need to collude to complete the full set of shares and calculate the sum.

As previously discussed, additive, Shamir's, and replicated secret sharing each have their particular methods of distributing and reconstructing secrets. While additive operations on shared secrets can be computed locally with the shares stored at each party, multiplicative operations often require additional computation [10]. Mathematical illustrations are provided in Appendix A.

A commonly used method to handle multiplication of shared secrets is through the use of Beaver triples. Beaver triples are precomputed sets of shared secrets $(a, b, c)$, where $a$ and $b$ are random numbers, and $c$ is the product of $a$ and $b$. These triples allow for the multiplication of two shared secrets without any interaction between the parties, which significantly improves efficiency and privacy. However, the generation and distribution of Beaver triples can be costly in terms of computation and communication [23].

For Shamir's secret sharing, the Damgård and Nielsen protocol can be used for multiplicative operations [24]. More complex operations, such as oblivious selection, division, logarithm, inverse square root, and uniformly random factor number, are detailed in [25].

The methodologies discussed here are fundamental for enabling secure computation in SMPC systems, ensuring the privacy of data while enabling collaborative computation.

## III. COMPARISON OF DIFFERENT SMPC APPROACHES AND OTHER PRIVACY-PRESERVING METHODS

This section provides a comparative analysis of different SMPC techniques as well as alternate privacy preservation strategies such as differential privacy, emphasizing their strengths, limitations, and use-cases.

### A. SMPC VIA HOMOMORPHIC ENCRYPTION

Homomorphic encryption-based SMPC offers a unique capability of enabling computations directly on encrypted data, thereby providing high-level protection against data breaches, as sensitive data remains encrypted throughout the computational process. However, fully implementing homomorphic encryption comes at a significant computational cost, often requiring substantial time and computational resources. Hence, it may not be the most viable option for scenarios demanding real-time or near-real-time computational requirements [14].

### B. SMPC THROUGH SECRET SHARING

Secret sharing-based SMPC presents an alternative to homomorphic encryption. It involves dividing data into multiple shares that are distributed among different parties. Computation is executed on these shares, ensuring the initial data remains concealed. This method offers superior computational speeds in contrast to homomorphic encryption but demands increased communication bandwidth due to the necessity of share exchanges between parties [26]. It also calls for enhanced coordination and mutual trust among participating parties.

### C. DIFFERENTIAL PRIVACY: AN ALTERNATE PRIVACY-PRESERVING METHOD

Although not an SMPC method, differential privacy represents another strategy for privacy-preserving computations. It introduces statistical noise to data, thereby safeguarding individual privacy. This technique permits aggregate statistical analyses without violating the privacy of individual data points. While differential privacy offers solid mathematical assurances of privacy and is often more efficient than SMPC methods, it inherently introduces a level of uncertainty into the results due to the added noise. Additionally, it does not provide the same level of security for individual data points as SMPC methods [27]–[29].

## IV. MACHINE LEARNING BACKGROUND

Machine learning is a technology that extracts patterns from data, providing the basis for predictions and decision making [30]. There are primarily three types of machine learning methods: supervised learning, unsupervised learning, and reinforcement learning. While reinforcement learning is a vital part of the machine learning ecosystem, it hasn't been extensively incorporated into SMPC. Therefore, this paper will focus on supervised and unsupervised learning as they pertain to SMPC systems.

Supervised learning involves models that learn from labeled training data. Each input instance in the dataset comes with the corresponding output label, allowing the model to learn to predict outputs for given inputs [31]. Popular algorithms in supervised learning, such as linear regression, decision trees, and support vector machines, find various applications in the context of SMPC, including predicting customer behavior, credit scoring, and medical diagnosis.

Unsupervised learning uses unlabeled data, and the model learns the inherent structure of the data through techniques like clustering or dimensionality reduction. The derived clusters or features often need further interpretation to provide meaningful insights [32]. Common algorithms in unsupervised learning, such as k-means clustering and principal component analysis, are used in applications like market segmentation, anomaly detection, and recommendation systems.

Reinforcement learning is a method where an agent learns to make decisions by interacting with an environment and receiving rewards or punishments [33]. Due to the complexity and large amounts of computation required, as well as privacy concerns related to the iterative and interactive nature of the learning process, it is not yet widely used in SMPC.

### A. USER MODEL

Machine Learning as a Service (MLaaS) refers to a range of services that offer machine learning tools as part of cloud computing services. These include data preprocessing, model training, visualization, and prediction [10].

In an MLaaS framework, according to [34], there are three major roles: the data gatherer, the learner, and the predictor. The data gatherer is responsible for collecting, preprocessing, and sending data for further processing. The learner uses this data to train the model, and the predictor uses the trained model to provide predictions or inferences based on new data queries. The MLaaS framework can be utilized in an SMPC setting, where these roles are performed in a distributed manner across multiple parties, ensuring privacy and reducing computational load on individual parties.

Applications of MLaaS in SMPC include collaborative learning, where multiple entities combine their data to train a common model while preserving the privacy of individual data, and inference services where a provider uses a pre-trained model to provide predictions based on input data from customers.

The roles and service styles of traditional machine learning services are transformed when considering SMPC services, enhancing data and model security. IEEE-defined roles in an SMPC system (Table 4) are distributed according to relevant services (Table 5). There are three primary service models.

1) Online Training: A customer sends a training dataset to cloud computing providers to offload computation. The trained model is returned to the customer. In an SMPC setting, both the training dataset and the model's weights are considered private data of the customer or data provider.
2) Online Inference: This service model involves a customer sending a query to a model provider, who then returns the inference result using their model. It's a common model for health monitoring applications [35]. The privacy of the customer's query data, which might contain sensitive health information, must be ensured. Equally, the model used by the provider should be kept private.
3) Multi-party Training: In this model, multiple parties each possess a private dataset. The result of this collaborative effort is a shared machine learning model that has been trained on all parties' combined data. Here, the privacy of each party's local datasets must be preserved.

A variant of the multi-party training model is federated learning. In this scenario, each party trains a local model on their dataset. The local model, along with the dataset, must be kept private to prevent unauthorized extraction of the local training dataset through model inversion attacks [36]. The distinction between online training and multi-party training lies in the different goals of the data providers. In online training, the data provider aims to offload computation to external servers, while in multi-party training, data providers also strive for higher model accuracy by utilizing private datasets from other data providers.

**TABLE 4.** IEEE Defined Roles in an SMPC System. [12]

| Role | Description |
|---|---|
| Task Initiator | Initiates the SMPC task and distributes roles. |
| Data Provider | The data owner. |
| Algorithm Provider | Provides task algorithm. |
| Coordinator | Distributes task and algorithm to related roles. |
| Computing Provider | Computes SMPC tasks. |
| Result Obtainer | Receives the computation results and transforms to final results. |

### B. FEDERATED LEARNING

Federated Learning (FL) is a machine learning paradigm that was introduced to address privacy and data locality issues that arise during the training of machine learning models. The concept was proposed by Google in 2016 [37] as a solution to train machine learning models across multiple decentralized devices or servers holding local data samples, without exchanging their data [38]–[40]. In FL, the goal is

**TABLE 5.** Machine Learning Service Models in SMPC Roles.

| Service | Task Initiator | Data Provider | Algorithm Provider | Coordinator | Computing Provider | Result Obtainer |
|---|---|---|---|---|---|---|
| Online Training | Same as the data provider. | An entity with data who requires external computation resources. | Same as the data provider. | Same as the computing provider. | External cloud computing service providers. | Same as the data provider. |
| Online Inference | Same as the data provider. | An entity, with a data query, seeking model inference services. | The machine learning model provider. | Same as the algorithm provider. | External cloud computing service providers. | Same as the data provider. |
| Multi-party Training | One of the data providers. | Multiple parties each with their own private dataset. | Agreed between the data providers | One of the data providers. | Could be the data providers or external cloud computing service providers. | Same as the Data Providers. |

to train a globally accurate model by aggregating locally trained models from participants, ensuring all data remains local. Nevertheless, vulnerabilities in the cloud server or between participants may jeopardize the security of this process. There are three types of federated learning: horizontal federated learning, vertical federated learning, and federated transfer learning [38], [41], [42]. SMPC can significantly enhance the privacy and security aspects of these types of federated learning.

Horizontal Federated Learning: This federated learning approach applies when datasets from different parties have the same features but different samples [43]. Parties collaboratively decide on a unified model structure. A collective model is then built by merging locally trained models from each participant. This approach facilitates data parallelism, where multiple instances of the identical model are trained on distinct portions of the training dataset [39]. In this setting, SMPC provides secure aggregation methods that protect the privacy of each party's local model. These techniques ensure that model aggregation is performed securely, thus preserving the privacy of each participant's contributions.

Vertical Federated Learning: In vertical federated learning, participants may have data entries from similar sample spaces, but the features of these samples differ [44]. This scenario allows for an aggregated description of a data entry with different features collected from multiple parties. A supporting operation for vertical federated learning, designed to identify intersections in the sample space, is private set intersection [7]. Through this operation, parties can disclose the intersection of their sets with other parties' sets without revealing additional data. SMPC can support this operation by providing the necessary cryptographic techniques to carry out the private set intersection [45].

Federated Transfer Learning: In a federated transfer learning setting, both feature and sample spaces have minimal overlaps among the parties. Therefore, transfer learning techniques are employed in this context to leverage the limited commonality between datasets [38]. This approach allows parties to improve their local models by learning from datasets that are not identical but are relevant or share some similarities. SMPC can mitigate the risk of model inversion or membership inference attacks in this setting, which is possible when parties exchange their local models for transfer learning. Furthermore, SMPC can maintain the privacy of local models and data during the learning process.

SMPC provides a set of cryptographic techniques that can enhance the security and privacy aspects of federated learning. These techniques ensure that local models and data remain private, mitigating potential threats and risks while enabling parties to collaboratively train a machine learning model [46]–[48].

## V. HOMOMORPHIC ENCRYPTION-BASED SMPC IN MACHINE LEARNING

Secure Multi-Party Computation (SMPC) applications in machine learning can be primarily categorized into two secure data sharing methods: homomorphic encryption and secret sharing. This section examines SMPC applications based on homomorphic encryption.

Homomorphic encryption is a form of encryption technique allowing computations to be carried out on on encrypted data, or ciphertexts. when the computed result is decrypted, it aligns perfectly with the outcome of the same operations executed on the original, unencrypted data (plaintext) [49]. In such scenarios, sensitive data often needs to be disseminated among various stakeholders. Homomorphic encryption ensures that while computations can be performed on this data, its actual content remains concealed, thereby safeguarding user privacy and data confidentiality [14].

Recent research has leveraged homomorphic encryption for secure model aggregation in federated learning. Federated learning involves training machine learning models on numerous decentralized devices, keeping the training data localized [50]–[53]. This collaborative approach allows multiple parties to jointly train models without exposing their training data, proving beneficial for industries handling sensitive data such as healthcare or finance. However, challenges remain, primarily related to computation offloading and security against collusion.

To tackle these challenges, solutions have been proposed

involving server-aided protocols, proxy re-encryption, and the integration of blockchain with federated learning. For instance, in [50], a server-aided protocol based on the El-Gamal encryption scheme is developed to facilitate model weight sharing in horizontal federated learning. An extension of this work deploys proxy re-encryption to enable training on Convolutional Neural Networks (CNNs) - a class of deep learning models most commonly applied to analyzing visual imagery [52]. However, these solutions also bring their own challenges, such as potentially high computational cost and susceptibility to collusion.

Moreover, some studies attempt to offload the computational burden from data providers through server-based training on encrypted data [54], [55]. This involves training Artificial Neural Networks (ANNs) and autoencoders using data encrypted by the Paillier scheme - a partially homomorphic encryption system. The challenge lies in executing complex operations such as division and exponentiation on encrypted data, integral to compute certain machine learning activation functions. To simplify the computation requirement on encrypted data, functional encryption is introduced - an advanced cryptographic technique where decryption keys enable the decryption of specific functions of the data rather than the data itself [55].

Despite the progress in leveraging homomorphic encryption for SMPC in machine learning, the field still faces challenges, as fundamental security requirements often conflict with computation offloading. Looking forward, exploring new cryptographic techniques, machine learning models, and computing paradigms that can balance these competing needs will be pivotal. Furthermore, benchmarking homomorphic encryption against other secure data sharing techniques such as secret sharing will help understand its relative merits and demerits in different application scenarios.

## VI. SECRET SHARING-BASED SMPC IN MACHINE LEARNING

Table 7 highlights recent advancements in machine learning where Secure Multi-Party Computation (SMPC) is utilized through secret sharing. Secret sharing is a method of distributing parts of confidential data to multiple parties to preserve privacy.

The pioneering works employing secret sharing primarily focused on delivering online inference services. For instance, the authors of [56] demonstrated a Sharemind-based additive sharing approach in a case study concerning maritime cargo consolidation. A machine learning model was designed utilizing an open distance table among all feasible ports, effectively reducing the search time. Confidentiality was maintained by secretly sharing the source and destination ports.

In another work [57], basic functionalities such as comparison, equality, bit decomposition, and truncation were implemented in the ring $Z_{2k}$, enabling online inference. The study proposed a two-party scenario consisting of a client, with a data entry, and a server, equipped with a decision tree or Support Vector Machine (SVM) model. Furthermore, they incorporated active checks to guard against potential malicious server activities.

An application in the domain of edge computing was showcased in [64], where coded computation was utilized to lower the computational requirements for running SMPC on edge servers. The system ensured the protection of data collected from end devices during specific computation tasks dictated by a master node. Computation with shared data was performed on multiple edge servers, with the final results reconstructed at the master node.

A multitude of secret sharing-based federated learning applications have been demonstrated in recent research [46], [58], [60]. In studies [46], [60], an innovative approach was undertaken to alleviate communication and computation burdens for shared model weights in peer-to-peer configurations. Rather than disseminating multiple shares to all peers, a select group of peers is nominated to aggregate the model using data shares [60], a method that leads to a reduction in the number and size of messages, as well as the overall execution time.

Researchers in [46] went a step further by curbing the communication cost of secret sharing to a level akin to non-SMPC systems. The technique they proposed involves sharing only the initial layer weights of the neural network in two phases. During the first phase, every participant disperses a share of the weights to all other peers, followed by each participant calculating a sum of the received shares. In the second phase, the sums calculated in the initial stage are transferred to a server for computation of the sum and averaged weights. The weights following the first model layer are aggregated in plain text on the server.

Apart from the notable applications in model aggregation, secret sharing also offers a unique application in private set intersection computations, essential for vertical federated learning. The authors in [58] implemented Shamir's secret sharing scheme to compute the private set intersection, effectively simplifying complex private set interactions to manageable AND operations on bit-vectors.

These applications underline the potential of federated learning in scenarios where multiple parties contribute data for training. An interesting use case can be found in [59] where secret sharing is used for natural language processing tasks involving multiple input data sources from various parties. Utilizing a seq2seq model built on Long Short-Term Memory (LSTM), which is known to all parties, the system safeguards the raw input data from each individual source, highlighting the wide array of applications secret sharing can have in preserving data privacy.

Recent advancements have centered on server-side SMPC that are capable of online training, online inference, and multi-party training [25], [61], [62], [65], [66]. For instance, CrypTen strives to facilitate multi-party training using PyTorch's Application Programming Interface (API) [61], offering parallel computation support for communication protocol computations via graphic processing units. While

**TABLE 6.** Homomorphic Encryption-based SMPC.

| Year | Type of Service | Encryption Protocol | SMPC Type | Security Model | Machine Learning Model |
|---|---|---|---|---|---|
| 2018 [50] | Multi-party training | ElGamal-based | Server-aided | Malicious | Not mentioned |
| 2019 [54] | Online training, Multi-party training | Paillier scheme | Server-aided | Semi-honest | ANN, Autoencoders |
| 2019 [51] | Multi-party training | SEAL-based | Peer-to-peer | Semi-honest | Gradient boosting trees |
| 2019 [55] | Online training, Online inference, Multi-party training | Functional encryption | Server-aided | N/A | CNN, General neural networks |
| 2020 [52] | Multi-party training | Proxy re-encryption | Server-aided | Malicious | CNN |
| 2022 [53] | Multi-party training | Not mentioned | Peer-to-peer | Not mentioned | Not mentioned |

**TABLE 7.** Secret Sharing-based SMPC for Machine Learning.

| Year | Type of Service | Sharing Method | SMPC Type | Security Model | Machine Learning Model |
|---|---|---|---|---|---|
| 2019 [56] | Online inference | Sharemind (Additive sharing) | Server-side | Semi-honest | Polynomial regression |
| 2019 [57] | Online inference | SPDZ-based additive and binary sharing | Peer-to-peer | Malicious | Decision tree, SVM |
| 2020 [58] | Multi-party training | Shamir's secret sharing | Server-aided | Semi-honest | Private set intersection for federated learning |
| 2020 [59] | (Multi-party) Online inference | Additive and multiplicative sharing | Peer-to-peer | Semi-honest | LSTM-based seq2seq |
| 2020 [60] | Multi-party training | Additive and Shamir's sharing | Peer-to-peer | Semi-honest | Horizontal federated learning |
| 2021 [46] | Multi-party training | Additive sharing | Server-aided | Semi-honest | CNN |
| 2021 [61] | Online training, Online inference, Multi-party training | Additive and binary sharing | Server-side, Server-aided | Semi-honest | Linear model, Wav2Letter, Residual network, Vision transformer |
| 2021 [62] | Online training, Online inference, Multi-party training | Additive, replicated and binary sharing | Server-side, Peer-to-peer | Semi-honest | CNN |
| 2021 [63] | Multi-party training | Replicated sharing | Server-side | Malicious | Feature selection, Logistic regression |
| 2022 [64] | Online inference | Shamir's secret sharing | Server-side | Semi-honest | Not mentioned |
| 2022 [65] | Online training, Online inference, Multi-party training | Additive sharing | Server-side | Semi-honest | CNN |
| 2022 [66] | Online training, Online inference, Multi-party training | Additive sharing | Server-side | Semi-honest | CNN |
| 2022 [25] | Online training, Online inference, Multi-party training | Additive, replicated and binary sharing | Server-side | Malicious | CNN |

Beaver triples' generation depends on a trusted third party, this party can be substituted with solutions incorporating homomorphic encryption or oblivious transfer. CrypTen has implemented a variety of secret shared operations including addition, multiplication, dot product, outer product, matrix product, and convolution based on additive secret sharing. Binary secret sharing safeguards comparators. Moreover, CrypTen approximates exponential, logarithm, and reciprocal functions via limit approximation, Householder iterations, and Newton-Raphson iterations respectively. These approximations serve as the foundation for the private computation of machine learning activation functions.

Building on CrypTen, a three-party system, CryptGPU, has been proposed [62]. It employs a two-out-of-three replicated sharing scheme. With a focus on the private computation of Convolutional Neural Networks (CNNs), the authors of [65] approximate division and square root operations in the batch normalization layer through multiple rounds of addition and multiplication. They employ regression, Taylor series, Chebyshev polynomial approximation, Lagrange polynomial approximation, and parametric polynomials to model the activation functions.

In another approach to compute activation functions, an SMPC system consisting of three computation nodes is employed in [66]. One node generates Beaver triples for secure multiplication and computes non-linear element-wise functions, such as sigmoid and hyperbolic tangent activation functions. A common sequence between the two computing

nodes is used to randomly permute elements within shared inputs and the results of these activation functions are computed on the third node following reconstruction.

The last work reviewed that satisfies all service models on server-side systems is [25]. The study underscores quantizing neural networks to boost computational speed and minimize communication. The authors propose an exponential algorithm aimed at adding support to negative values while separately computing the integer and fraction parts. This work is grounded in MP-SPDZ, which implements various protocols for operations under a malicious adversary model.

In addition to machine learning's training and inference operations, an SMPC system dedicated to multi-party feature selection has been proposed [63]. This proposition outlines a scenario where multiple data providers share their data with either three or four computing nodes for the purpose of feature selection. To provide a refined dataset for model training at the computing providers, the Gini scores of the features are computed. To safeguard against malicious adversaries, information-theoretic message authentication codes are used to validate the correctness of secret values.

## VII. SECURITY VULNERABILITIES OF SMPC SYSTEMS

Since one major application of SMPC in machine learning is to enhance the security of federated learning, some security vulnerabilities of federated learning are common in SMPC systems. On the other hand, as SMPC systems are distributed systems operating on heterogeneous devices [7], it shares some of the vulnerabilities of IoT systems [6]. Table 8 presents security vulnerabilities or attacks on SMPC systems.

The common attacks from the field of federated learning are data poisoning, malicious server, man-in-the-middle attacks, collusion attacks, and dropout of clients [67]. Data poisoning in federated learning refers to the corruption of training data by the clients or data providers. In the context of SMPC, this corruption can be extended to any shared data or computation results from the data providers by the computing providers. In [25], [57], [63], this vulnerability is mitigated using information-theoretic message authentication codes [68]. A special case of data poisoning is a malicious server. This attack aims at server-aided SMPCs. The compromised centralized aiding server could corrupt the computation results. The authors of [50] and [52] adopted the bilinear aggregate signature to ensure the server does not modify computation results. However, their clients operate with a shared set of public and private keys, and the server could obtain the keys through man-in-the-middle attacks. The server could intercept the shared keys and modify the data in later stages. For other types of SMPCs, any party could launch a man-in-the-middle attack to intercept communications between participants and modify the shared data or computation results. Collusion attacks are another method for the server to obtain the clients' shared keys. A compromised client, in the interest of the server, can directly share the keys with the server. For other types of SMPCs, collusion between multiple computing providers up to a threshold allows the reconstruction of shared data in secret sharing-based SMPC systems. Therefore, the choice of nodes is important, especially in dishonest majority settings. The final attack inspired by federated learning vulnerabilities is the dropout of clients. Participating parties could drop out for justifiable reasons, such as power outages or network issues. However, this unstable behavior should be recorded and reflected during the node selection process of future SMPC operations. The incorporation of blockchain could be a solution [53].

The attack identified by the IEEE standards is side-channel attack [12]. Side-channel attacks involve gathering or modifying information from a participating device through physical access. One mitigation method is to adopt other secure computation methods such as trusted execution environments [69]. Other countermeasures [70] were discussed at code, operation system, and architectural levels. Common network-related attacks are denial of service attacks, eavesdropping, and endpoint security [6]. A denial of service attack could be initiated by any internal or external parties to disrupt the service of an SMPC participating node. Eavesdropping is similar to man-in-the-middle attacks, but eavesdropping only listens to the communication without the attempt of modification. Finally, endpoint security concerns malicious behaviors from physically compromised devices. The authors of [71] have studied the detection of such compromised devices.

The IEEE standards did not directly identify many vulnerabilities of SMPC systems. However, the IEEE standards provided a list of recommended security requirements targeting network-related attacks [12]. For example, adopting secure transmission protocols such as Transport Layer Security (TLS) protects the system against man-in-the-middle attacks and eavesdropping. Network attacks and counter-attacks are a widely discussed topic. It has already been studied in other literature [6].

## VIII. CHALLENGES AND BARRIERS IN SMPC ADOPTION

While SMPC holds significant potential for privacy-preserving computations, several challenges remain in its widespread adoption for real-world applications.

### A. COMPUTATIONAL OVERHEAD

SMPC introduces significant computational overhead compared to traditional machine learning due to the complex cryptographic operations, like homomorphic encryption and secret sharing, needed for data privacy. This results in extended training and inference duration, especially for real-time applications such as autonomous driving or medical diagnostics [72], [73]. Additionally, the increased computational demands can lead to higher costs, particularly in cloud-based settings, due to prolonged resource usage [74].

### B. LEGAL AND REGULATORY CONCERNS

One of the primary challenges pertains to legal and regulatory issues. Different jurisdictions have different laws and regu-

**TABLE 8.** Common Attacks on SMPC Systems [6], [12], [67].

| Attacks | Description | Sources of Attack | Types of Adversery |
|---|---|---|---|
| Data Poisoning | Corrupting the values of shared data or computation results. | Data Provider, Computing Provider | Malicious |
| Malicious Server | The aggregating server in federated learning applications sends malicious model parameters. | The Aiding Server of Server-aided SMPC | Malicious |
| Man-in-the-Middle Attacks | Intercept the shared data or computation results and replace them with corrupted value. | Any Party | Malicious |
| Collusion Attacks | Collusion of parties beyond a threshold could reconstruct shared data. | Computing Provider | Malicious |
| Dropout of Clients | Participating parties drop out due to various possible reasons. | Any Participant | Malicious, Non-Malicious |
| Side-channel Attacks | Gathering or modifying information from a participating device through physical access. | Any Participating Device | Malicious |
| Denial of Service Attacks | An external adversary could exhaust communication and computation resources by flooding the participating nodes with requests. | Any Party | Malicious |
| Eavesdropping | An adversary secretly listens to a data provider's communication to obtain all data shares and reconstruct the original data. | Any Party | Semi-honest |
| Endpoint Security | SMPC does not ensure the security of the computation environment. A compromised device potentially allows the attacker to directly access the data and computation results of participants | Any Party | Malicious |

lations regarding data privacy, and navigating these can be complex. For instance, while SMPC allows for computation on encrypted data, it's possible that some jurisdictions might still consider this to be data processing under their laws, potentially requiring consent or other legal bases. Additionally, international data transfer regulations could also come into play in multi-party computations involving parties from different countries [75]. Therefore, organizations wishing to utilize SMPC must ensure that they are compliant with all relevant laws and regulations.

### C. USABILITY CHALLENGES

Another challenge is the usability of SMPC solutions. Implementing SMPC requires significant technical expertise and understanding of cryptographic principles, which can be a barrier for many organizations. Moreover, it can be challenging to integrate SMPC solutions with existing IT infrastructure and workflows, further complicating its adoption. There is a need for more user-friendly and easily integrable SMPC solutions to lower the entry barrier for non-expert users [75].

### D. MODEL AND DATA CONSTRAINTS

SMPC offers a promising avenue for privacy-preserving computations, but its application in machine learning is not universal. Certain machine learning models and datasets pose challenges when integrated with SMPC. For instance, deep learning models with multiple layers and a large number of parameters can be computationally intensive for SMPC due to the cryptographic operations involved [72]. Similarly, algorithms that rely heavily on floating-point arithmetic or iterative processes might not be optimal for SMPC, as these operations can be inefficient and slow when encrypted. Additionally, datasets with high dimensions or those requiring frequent updates can further complicate the integration with SMPC, leading to increased computational overhead and potential inaccuracies [73].

These challenges represent significant hurdles to the widespread adoption of SMPC. However, with ongoing research and development efforts, as well as increasing awareness and understanding of privacy issues, it is anticipated that these challenges will be progressively addressed, paving the way for broader usage of SMPC in real-world applications.

## IX. IMPLEMENTATION GUIDELINES FOR SMPC IN MACHINE LEARNING SYSTEMS

When implementing SMPC in machine learning systems, there are a few guidelines and best practices that can be helpful in achieving a successful deployment.

### A. CHOOSE THE RIGHT SMPC METHOD

One of the first considerations when implementing SMPC is to choose the right method. Homomorphic encryption and secret sharing-based SMPC each have their own strengths and weaknesses, as discussed previously. Therefore, the choice of method should depend on the specific requirements of the use case, such as the level of privacy required, the amount of computational resources available, and the nature of the machine learning tasks to be performed.

### B. CONSIDER EFFICIENCY

Efficiency is a crucial consideration when implementing SMPC. As mentioned earlier, homomorphic encryption can be computationally intensive, while secret sharing-based methods often require significant communication bandwidth. Therefore, steps should be taken to optimize the efficiency of the SMPC process, such as using efficient cryptographic protocols, minimizing the amount of data to be processed,

and leveraging hardware accelerators or parallel computing techniques when possible.

### C. INCORPORATE SECURITY MEASURES

It is essential to incorporate appropriate security measures when implementing SMPC. This includes measures such as user authentication, authorization, and auditing, as recommended by the IEEE. Ensuring secure storage and transmission of data, using multiple protocol encryption schemes, and verifying the SMPC version are also important for maintaining system security.

### D. IMPLEMENT VERIFICATION MECHANISMS

To ensure the integrity of data and the correctness of computation results, it's important to have a verification plan in place. This can involve techniques like bilinear signatures for federated learning, morphism-based methods for computation verification, and even blockchain technology for data integrity verification.

### E. ADDRESS USABILITY ISSUES

Usability is another crucial consideration when implementing SMPC. Efforts should be made to make SMPC solutions as user-friendly and easily integrable as possible to reduce the barriers to adoption. Providing clear documentation, offering user support, and building user-friendly interfaces can all contribute to improving the usability of SMPC solutions.

### F. STAY INFORMED ABOUT LEGAL AND REGULATORY ISSUES

It is crucial to stay informed about relevant legal and regulatory issues. Given the potential complexity of navigating data privacy laws and regulations, it may be beneficial to seek legal advice when implementing SMPC, particularly in scenarios involving international data transfers or sensitive data types.

## X. LIMITATIONS AND FUTURE WORK

This section presents a forward-looking view of SMPC applications in machine learning, highlighting current limitations and potential future work related to IEEE recommended security requirements, large-scale machine learning, information exchange, incentive schemes, data verification, and operational efficiency of SMPC.

### A. ADDITIONAL IEEE RECOMMENDED SECURITY REQUIREMENTS

Existing literature on SMPC applications often overlooks certain security requirements recommended by the IEEE [12]. These include authentication, which guarantees user identity management; authorization, controlling user access privileges; and auditing, a systematic process that logs significant system activities. Future work could look into integrating these security measures within the SMPC system to further protect against malicious access.

### B. LIMITATIONS OF SMPC FOR LARGE-SCALE MACHINE LEARNING

The adoption of SMPC for large-scale machine learning is limited by the general overheads of SMPC, and the nature of the encryption methods. Large-scale machine learning operations are bounded by large data volumes and complex model structures [76]. In the SMPC environment, large number of participants should also be a feature of large-scale machine learning operations. The increases in data volume, model complexity, and participants significantly increase the computational and communication overheads of SMPC systems [77] and thus limit system scalability.

This article explored the application of homomorphic encryption and secret sharing to share data between different parties. Floating-point computation under these encryption methods could affect the precision of the computation results. Most secret sharing methods operate in a finite field, which limits the machine learning data and model representation to integers. One of the method is to perform model quantization to convert the values to integers [25]. Other methods [78] explore floating-point representation by multiple variables. Inevitably, both types of methods involve rounding up the values, which sacrifices precision. Homomorphic encryption deal with floating-points by fixed-point arithmetic, approximate arithmetic, and custom encodings [79]. Fixed-point arithmetic has rounding errors, approximate arithmetic suffers from approximation errors, and custom encodings from encoding and further operations.

### C. PREPROCESSING AND INFORMATION EXCHANGE FOR SMPC

While a significant amount of work focuses on the computational aspects of machine learning operations within SMPC, less attention has been paid to necessary preprocessing steps such as data cleaning, normalization, and feature selection [80]. The development of secure implementations of these preprocessing operations is a promising area for future research [72]. Additionally, the exchange of dataset and computation requirements prior to SMPC computation is crucial. Blockchain technology offers a potential solution for facilitating this information exchange [81], yet comprehensive blockchain systems that coordinate a wide range of SMPC operations are currently underexplored.

### D. INCENTIVE MECHANISMS IN SMPC SYSTEMS

The establishment of a cohesive group of computation contributors for SMPC, especially in frameworks leveraging secret sharing, involve a meticulous evaluation of the incentives driving participant engagement [72]. While existing literature provides some insights, it often lacks comprehensive strategies tailored for this context [73]. A well-structured incentive mechanism should not only recognize and reward the computational contributions but also acknowledge the value brought in by data providers. Such a holistic approach can potentially amplify participation rates in SMPC endeavors. Furthermore, ensuring the authenticity and accuracy of

the data and computational outputs is paramount, warranting robust verification protocols within these incentive structures [80].

### E. VERIFICATION METHODS FOR DATA INTEGRITY AND COMPUTATION RESULTS

As the input and output data in SMPC are encrypted, verifying data integrity and computation results requires further operations. There are some methods to ensure the integrity of the shared data. In [50], [52], the authors used bilinear signatures to ensure the aggregated weights in federated learning are unchanged. There is also a low-overhead morphism-based method to verify computation results [82]. There is also the use of information-theoretic message authentication codes [68]. However, these methods do not ensure the data is computed through the desired process. On the other hand, in the application of machine learning, it is more important to verify the data quality of the original data from the data providers. Methods such as differential privacy only verify the possession of a dataset with certain quality [83]. However, it does not ensure this data quality during operation as all data are encrypted. Data quality verification during operation on private datasets should be developed for SMPC applications in machine learning.

### F. OPTIMIZING OPERATIONAL EFFICIENCY IN SMPC SYSTEMS

The current landscape of SMPC implementations often grapples with substantial computational and communicative overheads, rendering them suboptimal for environments with limited resources or applications demanding real-time responses [72]. A pivotal direction for upcoming research would be the refinement of SMPC's operational efficiency. This could be achieved by minimizing the communication rounds or curtailing the data volume exchanged during SMPC processes [73].

Incorporating differential privacy methodologies can strike a balance between ensuring rigorous privacy safeguards and facilitating more streamlined computations. Moreover, the advent of quantum computing, with its potential to revolutionize computational paradigms, presents a captivating prospect for augmenting SMPC's capabilities. However, it's imperative to note that quantum-enhanced SMPC is still in its formative phase and warrants extensive exploration.

### G. EVALUATING COMPUTATION AND COMMUNICATION EFFICIENCY OF SMPC

Homomorphic encryption and secret sharing, the two primary SMPC techniques used in machine learning, each have inherent computational and communication challenges. Future work should aim to benchmark the computation and communication costs of these and other SMPC techniques, with a focus on specific application scenarios. Such benchmarking could illuminate the trade-offs between security, privacy, and efficiency, and help in identifying the most suitable SMPC method for a given machine learning application [9].

In a broader perspective, the continued evolution of SMPC applications in machine learning will require focused research efforts in several areas, including security measures, preprocessing and information exchange protocols, incentive schemes, operational efficiency, and the evaluation of computational and communication costs. These efforts will be crucial for bridging the gap between the theoretical potential of SMPC and its practical utility in real-world machine learning applications.

## XI. FUTURE TRENDS IN SMPC FOR MACHINE LEARNING

Secure Multi-party Computation (SMPC) holds significant potential for driving the future of privacy-preserving machine learning. While current efforts focus on overcoming technical challenges and expanding applications, the field is expected to evolve towards more complex, efficient, and holistic solutions in the long term. Here, we speculate on some of the potential trends that might shape the future of SMPC in machine learning.

### A. INTEGRATION WITH OTHER PRIVACY-PRESERVING TECHNIQUES

As the field matures, we might see more integration of SMPC with other privacy-preserving techniques like Differential Privacy (DP) and Federated Learning (FL). Combining these techniques could lead to more robust solutions that offer a better trade-off between privacy, utility, and performance. For instance, SMPC can be used to securely aggregate model updates in FL while DP can be applied to provide statistical guarantees of privacy.

### B. STANDARDIZATION AND INTEROPERABILITY

There will likely be a push towards standardization and interoperability in SMPC protocols and implementations. Standardization can streamline the design and deployment of SMPC solutions, foster collaboration, and promote the adoption of best practices. Meanwhile, interoperability can facilitate the integration of different SMPC systems and allow for more flexible and scalable privacy-preserving computations.

### C. HARDWARE-SPECIFIC OPTIMIZATIONS

Future research might explore hardware-specific optimizations for SMPC. By leveraging specialized hardware such as Graphical Processing Units (GPUs), Tensor Processing Units (TPUs), or even quantum processors, we might achieve significant improvements in the computation and communication efficiency of SMPC operations.

### D. AUTOMATED AND ADAPTIVE SMPC SYSTEMS

The development of automated and adaptive SMPC systems could be another major trend. These systems could automatically select the most appropriate SMPC method based on the specific requirements and constraints of a given task.

They might also adapt to changes in the computational environment, data characteristics, or privacy requirements to maintain optimal performance and security.

### E. LEGAL AND ETHICAL CONSIDERATIONS

As SMPC becomes more widely adopted, there will likely be increased scrutiny of the legal and ethical implications of its use. This could lead to the development of new regulations, guidelines, and ethical frameworks for SMPC in machine learning, with a particular emphasis on ensuring fairness, transparency, and accountability.

The future of SMPC in machine learning is likely to be exciting and dynamic, marked by continual innovation, collaboration, and evolution. As we continue to push the boundaries of what is possible, SMPC has the potential to become an indispensable tool for enabling privacy-preserving machine learning in a wide range of applications.

## XII. CONCLUSION

Secure Multi-party Computation (SMPC) continues to gain traction in machine learning, offering novel ways to safeguard data privacy during computations. Through a review of various studies, this paper affirms the efficacy and versatility of both homomorphic encryption-based and secret sharing-based SMPC methods across different machine learning tasks—ranging from model training and inference to feature selection and private set intersection.

Nonetheless, present SMPC implementations confront several constraints, primarily around the significant computational resources and communication bandwidth requirements. Additionally, there is a distinct lack of standardized security measures and practices, alongside an inadequacy of robust methods for verifying data integrity and computation results. Current incentive structures within SMPC systems are far from comprehensive, often neglecting key elements such as data quality and sustained participation.

Despite these obstacles, the promise and potential of SMPC in machine learning remain intact. Future research is poised to address these limitations by refining cryptographic techniques and computation strategies for enhanced efficiency, standardizing security measures, and introducing more robust incentive and verification mechanisms. With such advancements on the horizon, SMPC is slated to play an even more critical role in propelling secure, privacy-preserving machine learning applications.

.

## APPENDIX A MATHEMATICAL ILLUSTRATION OF SECRET SHARING METHODS

### A. ADDITIVE SECRET SHARING

In Additive Secret Sharing, a secret is divided into shares, and each share is generated by adding a random value to a specific portion of the secret. The shares are distributed among the parties, and the secret can only be reconstructed by combining a sufficient number of shares.

Here is a simplified example of Additive Secret Sharing for three parties. Suppose we want to share a secret $s$:

1) Random values $x$, $y$, and $z$ are generated by the dealer (the one who shares the secret).
2) The dealer computes three shares as follows:
   - The share for Party 1 is $s_1 = s + x$.
   - The share for Party 2 is $s_2 = s + y$.
   - The share for Party 3 is $s_3 = s + z$.
3) Each party now holds one share, which contains a portion of the secret $s$ plus a random value.
4) To reconstruct the secret, any subset of parties with a sufficient number of shares can add their shares together:
   - If Party 1 and Party 2 collaborate, they can compute $s$ as $s_1 + s_2 = (s+x)+(s+y) = 2s+x+y$.
   - Similarly, Party 2 and Party 3 can compute $s$ as $s_2 + s_3 = (s+y)+(s+z) = 2s+y+z$. Party 1 and Party 3 can compute $s$ as $s_1 + s_3 = (s+x)+(s+z) = 2s+x+z$.
5) The result of any of these computations is $2s$ plus the sum of the random values ($x+y$, $y+z$, or $x+z$). By subtracting the sum of the random values, the parties can obtain the original secret $s$.

In practice, these computations are performed modulo a prime number to ensure that the secret remains secure and the operations are reversible. Additionally, techniques such as Shamir's Secret Sharing can be used to extend the scheme to work with more than three parties and achieve the desired threshold for secret reconstruction.

### B. SHAMIR'S SECRET SHARING

Shamir's Secret Sharing scheme is a type of polynomial secret sharing. The basic concept of Shamir's Secret Sharing is that $k$ points are sufficient to define a polynomial of degree $(k-1)$. In a $(k,n)$ threshold scheme, a secret $s$ is shared among $n$ participants such that any $k$ of these shares can reconstruct the secret, but $k-1$ or fewer shares reveal no information about the secret.

Let's illustrate this with a concrete example: Suppose you want to share a secret $s$ among $n$ participants such that at least $k$ participants are needed to reconstruct the secret.

1) Choose a $(k-1)$ degree polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$, where $a_0 = s$ (the secret), and $a_1, ..., a_{k-1}$ are randomly chosen coefficients.
2) Generate the shares of the secret by evaluating the polynomial at $n$ distinct points $x_1, ..., x_n$. Each share $s_i$ is a pair $(x_i, f(x_i))$.
3) Any group of $k$ or more participants can reconstruct the polynomial (and thus the secret) using their shares and the method of polynomial interpolation (such as Lagrange interpolation).
4) If there are fewer than $k$ shares, the secret $s$ can't be reconstructed because there are infinitely many degree $(k-1)$ polynomials that pass through any given set of $(k-1)$ or fewer points.

Remember to state clearly that the operations are usually performed in a finite field to ensure the security of the scheme and avoid real-number computation issues.

### C. REPLICATED SECRET SHARING

In Replicated Secret Sharing, a secret is shared among n parties in such a way that any subset of size less than half of the parties learns nothing about the secret, but any subset of size greater than or equal to half can recover the secret.

Here is a simplified example of Replicated Secret Sharing for three parties. Suppose we want to share a secret $s$:

1) First, two random values $x$ and $y$ are generated by the dealer (the one who shares the secret).
2) The dealer then computes three shares as follows:
   - The share for Party 1 is $(x, y)$.
   - The share for Party 2 is $(x, s - y)$.
   - The share for Party 3 is $(s - x, y)$.
3) Each party now has two values, one from the set $x, s - x$ and one from $y, s - y$. Note that any single share reveals nothing about the secret, since it could be created by any pair of random values $x$ and $y$.
4) Any two parties can reconstruct the secret by adding their shares together:
   - Party 1 and Party 2 can compute $s$ by adding their shares component-wise: $(x, y) + (x, s - y) = (2x, s) = (s, s)$.
   - Similarly, Party 2 and Party 3 can compute $s$ as $(x, s - y) + (s - x, y) = (s, s)$.
   - Party 1 and Party 3 can compute $s$ as $(x, y) + (s - x, y) = (s, s)$.
5) The result is $(s, s)$. The first component is discarded, and the second component $s$ is the reconstructed secret.

Again, in practice, these operations are performed in a finite field to ensure the security of the scheme and avoid issues with real number computations.

## APPENDIX B OPEN-SOURCE SOFTWARE AND TOOLS FOR SMPC

There's a growing assortment of open-source software, libraries, and tools available for the implementation of Secure Multi-party Computation (SMPC). Developers now have a variety of options tailored to their specific needs. Below, we highlight several noteworthy options currently accessible.

#### 1) Paillier Cryptosystem [17]

This encryption scheme is widely recognized for its partial homomorphic properties. It enables the encryption of integers and performs addition operations on the encrypted integers, gaining popularity due to its simplicity and efficiency.

#### 2) CrypTen [84]

Originating from Facebook's AI Research lab (FAIR), CrypTen aspires to provide a secure platform for machine learning using PyTorch. Specifically engineered for multi-party computation, CrypTen has implemented a number of useful machine learning functions.

#### 3) TF Encrypted [85]

TF Encrypted, an extension of TensorFlow, offers a set of tools specifically designed for encrypted machine learning. It accommodates secure computation primitives such as secret sharing and multi-party computation, along with privacy-preserving training and inference.

#### 4) MP-SPDZ [86]

MP-SPDZ is a comprehensive framework for multi-party computation. It supports a variety of secret sharing types and fully homomorphic encryption. Equipped with implementations of numerous cryptographic protocols, it can be utilized for a vast array of applications.

#### 5) HElib [87]

HElib is a software library dedicated to implementing homomorphic encryption, allowing for arithmetic operations to be performed on encrypted data. This feature makes it an advantageous tool for privacy-preserving computations.

#### 6) Microsoft SEAL [88]

The Simple Encrypted Arithmetic Library (SEAL) by Microsoft is a user-friendly library designed for performing homomorphic encryption. It offers a high-level API for executing arithmetic operations on encrypted data.

#### 7) Obliv-C [89], [90]

Obliv-C enhances the C programming language, providing benefits for developers well-versed in C. While it's designed for distributed systems, it may not be as feature-rich as some other SMPC libraries.

#### 8) Sharemind [91], [92]

Sharemind supports data processing between multiple parties while ensuring privacy. Although it might lack flexibility for non-standard computations, it makes up for it with a strong focus on secure computation principles.

#### 9) SCALE-MAMBA [93], [94]

SCALE-MAMBA stands out with its flexibility and user-friendliness, supporting a wide array of computations. Additionally, it offers integration capabilities with other languages such as C++ and Python, thus enhancing its versatility.

#### 10) JIFF [95]

Uniquely focused on web-based SMPC, JIFF serves as an excellent choice for developers creating secure web applications. It may, however, not be as suitable for non-web use cases.

#### 11) ABY [96], [97]

ABY concentrates on efficient two-party computation, leveraging a mix of cryptographic protocols. While it's an ex-

cellent fit for two-party computations, it might not be the optimal choice for multi-party scenarios.

#### 12) Fresco [98]

Fresco is a Java-based SMPC framework, an appealing choice for developers with a background in Java. It provides a high-level language for creating complex protocols from simple building blocks.

## ACKNOWLEDGMENT

The authors acknowledge the support of Food Agility CRC Ltd, funded under the Commonwealth Government CRC Program. The CRC Program supports industry-led collaborations between industry, researchers and the community. The financial and in-kind support of Robert Bosch (Australia) Pty Ltd & Robert Bosch GmbH in completing this work is also acknowledged.

## REFERENCES

[1] I. Zhou, I. Makhdoom, N. Shariati, M. A. Raza, R. Keshavarz, J. Lipman, M. Abolhasan, and A. Jamalipour, "Internet of things 2.0: Concepts, applications, and future directions," IEEE Access, vol. 9, pp. 70 961–71 012, 2021.

[2] P. V. Desai, "A survey on big data applications and challenges," in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2018, pp. 737–740.

[3] S. A. Lokhande, "Effective use of big data in precision agriculture," in 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2021, pp. 312–316.

[4] D. Winkler, A. Korobeinykov, P. Novák, A. Lüder, and S. Biffl, "Big data needs and challenges in smart manufacturing: An industry-academia survey," in 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA ), Vasteras, Sweden, 2021, pp. 1–8.

[5] IBM, Cost of a Data Breach Report 2022, IBM, 2022, (Accessed: September 11, 2022). [Online]. Available: https://www.ibm.com/security/data-breach

[6] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Anatomy of threats to the internet of things," IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1636–1675, 2019.

[7] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y. an Tan, "Secure multi-party computation: Theory, practice and applications," Information Sciences, vol. 476, pp. 357–372, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518308338

[8] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in Proceedings of the 2001 workshop on New security paradigms, 2001, pp. 13–22.

[9] Y. Wang, T. Li, H. Qin, J. Li, W. Gao, Z. Liu, and Q. Xu, "A brief survey on secure multi-party computing in the presence of rational parties," Journal of Ambient Intelligence and Humanized Computing, vol. 6, no. 6, pp. 807–824, 2015.

[10] D. Feng and K. Yang, "Concretely efficient secure multi-party computation protocols: survey and more," Security and Safety, vol. 1, p. 2021001, 2022.

[11] D. Evans, V. Kolesnikov, M. Rosulek et al., "A pragmatic introduction to secure multi-party computation," Foundations and Trends® in Privacy and Security, vol. 2, no. 2-3, pp. 70–246, 2018.

[12] "IEEE Recommended Practice for Secure Multi-Party Computation," IEEE Std 2842-2021, pp. 1–30, 2021.

[13] Q. Do, B. Martini, and K.-K. R. Choo, "The role of the adversary model in applied security research," Computers & Security, vol. 81, pp. 156–181, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818306369

[14] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," ACM Computing Surveys (Csur), vol. 51, no. 4, pp. 1–35, 2018.

[15] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469–472, 1985.

[16] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in Advances in Cryptology — EUROCRYPT'98, K. Nyberg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 127–144.

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in Cryptology — EUROCRYPT '99, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.

[18] C.-I. Fan, Y.-W. Hsu, C.-H. Shie, and Y.-F. Tseng, "Id-based multireceiver homomorphic proxy re-encryption in federated learning," ACM Trans. Sen. Netw., vol. 18, no. 4, nov 2022. [Online]. Available: https://doi.org/10.1145/3540199

[19] J. Li, Z. Qiao, K. Zhang, and C. Cui, "A lattice-based homomorphic proxy re-encryption scheme with strong anti-collusion for cloud computing," Sensors, vol. 21, no. 1, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/1/288

[20] A. Beimel, "Secret-sharing schemes: A survey," in Coding and Cryptology, Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 11–46.

[21] Q. Li, I. Cascudo, and M. G. Christensen, "Privacy-preserving distributed average consensus based on additive secret sharing," in 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1–5.

[22] R. Cramer, I. Damgård, and Y. Ishai, "Share conversion, pseudorandom secret-sharing and applications to secure computation," in Theory of Cryptography, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 342–362.

[23] J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein, "High-throughput secure three-party computation for malicious adversaries and an honest majority," in Advances in Cryptology – EUROCRYPT 2017, J.-S. Coron and J. B. Nielsen, Eds. Cham: Springer International Publishing, 2017, pp. 225–255.

[24] V. Goyal, H. Li, R. Ostrovsky, A. Polychroniadou, and Y. Song, "Atlas: Efficient and scalable mpc in the honest majority setting," in Advances in Cryptology – CRYPTO 2021, T. Malkin and C. Peikert, Eds. Cham: Springer International Publishing, 2021, pp. 244–274.

[25] M. Keller and K. Sun, "Secure quantized training for deep learning," in International Conference on Machine Learning. PMLR, 2022, pp. 10 912–10 938.

[26] W. Huang, M. Langberg, J. Kliewer, and J. Bruck, "Communication efficient secret sharing," IEEE Transactions on Information Theory, vol. 62, no. 12, pp. 7195–7206, 2016.

[27] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh, "Federated learning and differential privacy for medical image analysis," Scientific reports, vol. 12, no. 1, p. 1953, 2022.

[28] J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "The limits of differential privacy (and its misuse in data release and machine learning)," Communications of the ACM, vol. 64, no. 7, pp. 33–35, 2021.

[29] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, "An adaptive federated learning scheme with differential privacy preserving," Future Generation Computer Systems, vol. 127, pp. 362–372, 2022.

[30] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," arXiv:1803.01164 [cs], Sep. 2018.

[31] M. M. Eihab Mohammed Bashier, Muhammad Badruddin Khan, "Introduction to machine learning," in Machine Learning. Boca Raton, FL: CRC Press, 2016, ch. 1, pp. 37–92.

[32] T. Hastie, R. Tibshirani, and J. Friedman, "Introduction," in The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York, NY: Springer New York, 2009, ch. 1, pp. 1–8.

[33] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[34] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2015, pp. 896–902.

[35] S. Hijazi, A. Page, B. Kantarci, and T. Soyata, "Machine learning in cardiac health monitoring and decision support," Computer, vol. 49, no. 11, pp. 38–48, 2016.

[36] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, Denver Colorado, USA, 2015, pp. 1322–1333.

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[38] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, Jan. 2019.

[39] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: a survey," Knowledge and Information Systems, vol. 64, no. 4, p. 885–917, 2022.

[40] K. K. Coelho, M. Nogueira, A. B. Vieira, E. F. Silva, and J. A. M. Nacif, "A survey on federated learning for security and privacy in healthcare applications," Computer Communications, vol. 207, pp. 113–127, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036642300172X

[41] S. Shen, T. Zhu, D. Wu, W. Wang, and W. Zhou, "From distributed machine learning to federated learning: In the view of data privacy and security," Concurrency and Computation: Practice and Experience, vol. 34, no. 16, p. e6002, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6002

[42] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: a survey," Complex & Intelligent Systems, vol. 7, pp. 639–657, 2021.

[43] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2597–2604.

[44] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 181–192.

[45] H. Shi, Y. Jiang, H. Yu, Y. Xu, and L. Cui, "Mvfls: multi-participant vertical federated learning based on secret sharing," The Federate Learning, pp. 1–9, 2022.

[46] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Melbourne, Australia, 2021, pp. 828–835.

[47] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1333–1345, 2018.

[48] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," Applied Sciences, vol. 12, no. 19, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/19/9901

[49] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178.

[50] X. Ma, F. Zhang, X. Chen, and J. Shen, "Privacy preserving multi-party computation delegation for deep learning in cloud computing," Information Sciences, vol. 459, pp. 103–116, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518303608

[51] Z. Feng, H. Xiong, C. Song, S. Yang, B. Zhao, L. Wang, Z. Chen, S. Yang, L. Liu, and J. Huan, "Securegbm: Secure multi-party gradient boosting," in 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 1312–1321.

[52] X. Ma, C. Ji, X. Zhang, J. Wang, J. Li, K.-C. Li, and X. Chen, "Secure multiparty learning from the aggregation of locally trained models," Journal of Network and Computer Applications, vol. 167, p. 102754, 2020.

[53] S. Singh, S. Rathore, O. Alfarraj, A. Tolba, and B. Yoon, "A framework for privacy-preservation of iot healthcare data using federated learning and blockchain technology," Future Generation Computer Systems, vol. 129, pp. 380–388, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21004726

[54] A. N. Khan, M. Yu Fan, A. Malik, and R. A. Memon, "Learning from privacy preserved encrypted data on cloud through supervised and unsupervised machine learning," in 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2019, pp. 1–5.

[55] R. Xu, J. B. Joshi, and C. Li, "Cryptonn: Training neural networks over encrypted data," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 2019, pp. 1199–1209.

[56] Z. Li, C. Kitcharoenpaisan, P. Phunchongharn, Y. Yang, R. S. M. Goh, and Y. Li, "Efficient multi-party computation algorithm design for real-world applications," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 2019, pp. 1006–1009.

[57] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev, "New primitives for actively-secure mpc over rings with applications to private machine learning," in 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1102–1120.

[58] L. Lu and N. Ding, "Multi-party private set intersection in vertical federated learning," in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 2020, pp. 707–714.

[59] Q. Feng, D. He, Z. Liu, H. Wang, and K.-K. R. Choo, "Securenlp: A system for multi-party privacy-preserving natural language processing," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3709–3721, 2020.

[60] R. Kanagavelu, Z. Li, J. Samsudin, Y. Yang, F. Yang, R. S. Mong Goh, M. Cheah, P. Wiwatphonthana, K. Akkarajitsakul, and S. Wang, "Two-phase multi-party computation enabled privacy-preserving federated learning," in 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, VIC, Australia, 2020, pp. 410–419.

[61] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," in Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 4961–4973.

[62] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "Cryptgpu: Fast privacy-preserving machine learning on the gpu," in 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2021, pp. 1021–1038.

[63] X. Li, R. Dowsley, and M. De Cock, "Privacy-preserving feature selection with secure multiparty computation," in Proceedings of the 38th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 6326–6336. [Online]. Available: https://proceedings.mlr.press/v139/li21e.html

[64] E. Vedadi, Y. Keshtkarjahromi, and H. Seferoglu, "Adaptive gap entangled polynomial coding for multi-party computation at the edge," in 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 2022, pp. 1217–1222.

[65] C. Berry and N. Komninos, "Efficient optimisation framework for convolutional neural networks with secure multiparty computation," Computers & Security, vol. 117, p. 102679, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822000773

[66] F. Zheng, C. Chen, X. Zheng, and M. Zhu, "Towards secure and practical machine learning via secret sharing and random permutation," Knowledge-Based Systems, vol. 245, p. 108609, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122002738

[67] N. Bouacida and P. Mohapatra, "Vulnerabilities in federated learning," IEEE Access, vol. 9, pp. 63 229–63 249, 2021.

[68] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "Spdz 2k: Efficient mpc mod 2k for dishonest majority." CRYPTO, 2018.

[69] A. Dubey, R. Cammarota, V. Suresh, and A. Aysu, "Guarding machine learning hardware against physical side-channel attacks," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 18, no. 3, pp. 1–31, 2022.

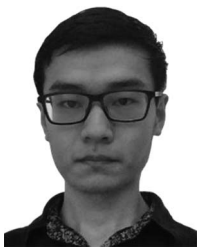
[70] Y. Lyu and P. Mishra, "A survey of side-channel attacks on caches and countermeasures," Journal of Hardware and Systems Security, vol. 2, pp. 33–50, 2018.

[71] I. Makhdoom, M. Abolhasan, D. Franklin, J. Lipman, C. Zimmermann, M. Piccardi, and N. Shariati, "Detecting compromised iot devices: Existing techniques, challenges, and a way forward," Computers & Security, vol. 132, p. 103384, 2023.

[72] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in 2017 IEEE symposium on security and privacy (SP). IEEE, 2017, pp. 19–38.

[73] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan et al., "Towards federated learning at scale: System design," Proceedings of machine learning and systems, vol. 1, pp. 374–388, 2019.

[74] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," Cryptology ePrint Archive, 2016.
[75] C. Bennett and S. Oduro Marfo, "Privacy, voter surveillance and democratic engagement: challenges for data protection authorities," in International Conference of Data Protection and Privacy Commissioners (ICDPPC), 2019.
[76] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, "A survey on large-scale machine learning," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 6, pp. 2574–2594, 2022.
[77] X. Wang, S. Ranellucci, and J. Katz, "Global-scale secure multiparty computation," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 39–56.
[78] A. Chandramouli, A. Choudhury, and A. Patra, "A survey on perfectly secure verifiable secret-sharing," ACM Computing Surveys (CSUR), vol. 54, no. 11s, pp. 1–36, 2022.
[79] T. V. T. Doan, M.-L. Messai, G. Gavin, and J. Darmont, "A survey on implementations of homomorphic encryption schemes," The Journal of Supercomputing, vol. 79, no. 13, pp. 15 098–15 139, 2023.
[80] P. Rindal and M. Rosulek, "Faster malicious 2-party secure computation with {Online/Offline} dual execution," in 25th USENIX Security Symposium (USENIX Security 16), 2016, pp. 297–314.
[81] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4177–4186, 2019.
[82] O. O. Olakanmi and K. O. Odeyemi, "Trust-aware and incentive-based offloading scheme for secure multi-party computation in internet of things," Internet of Things, vol. 19, p. 100527, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660522000294
[83] X. Yao, X. Zhou, and J. Ma, "Differential privacy of big data: an overview," in 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS). IEEE, 2016, pp. 7–12.
[84] "Crypten," https://crypten.ai/.
[85] "Tf encrypted," https://tf-encrypted.io/.
[86] M. Keller, "Mp-spdz: A versatile framework for multi-party computation," Cryptology ePrint Archive, Paper 2020/521, 2020, https://github.com/data61/MP-SPDZ. [Online]. Available: https://eprint.iacr.org/2020/521
[87] S. Halevi and V. Shoup, "Design and implementation of helib: a homomorphic encryption library," Cryptology ePrint Archive, Paper 2020/1481, 2020, https://github.com/homenc/HElib. [Online]. Available: https://eprint.iacr.org/2020/1481
[88] "Microsoft seal," https://www.microsoft.com/en-us/research/project/microsoft-seal/.
[89] S. Zahur and D. Evans, "Obliv-c: A language for extensible data-oblivious computation," Cryptology ePrint Archive, 2015.
[90] "Obliv-c: A language for extensible data-oblivious computation," https://oblivc.org/.
[91] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in Computer Security - ESORICS 2008, S. Jajodia and J. Lopez, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 192–206.
[92] "Sharemind: Privacy technology enabling new data-driven services," https://sharemind.cyber.ee/.
[93] L. Benmouffok, K. Singh, N. Heulot, and D. Augot, "Privacy-preserving initial public offering using scale-mamba and hyperledger fabric," in 2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2021, pp. 85–88.
[94] "Scale-mamba mpc software system," https://github.com/KULeuven-COSIC/SCALE-MAMBA.
[95] "Jiff: Javascript library for building web-based applications," https://github.com/multiparty/jiff.
[96] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in NDSS, 2015.
[97] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 35–52. [Online]. Available: https://doi.org/10.1145/3243734.3243760
[98] "Fresco: A framework for efficient secure computation," https://fresco.readthedocs.io/en/latest/intro.html.



IAN ZHOU Ian Zhou received the B.S. degree in computer science from the University of Sydney, Australia, in 2016. He also received the MBA and Ph.D. degree from the University of Technology Sydney in 2019 and 2023. His Ph.D. research focused on machine learning-based frost monitoring systems. His other research interest includes AI, IoT, cyber-physical system, and blockchain.

FARZAD TOFIGH completed his PhD at University of Technology, Sydney (UTS) in 2019 focusing on non-intrusive techniques for estimating crowd density through the use of distributed sensor networks (IoT) and an innovative approach that utilizes electromagnetic energy monitoring with metamaterial absorbers. At UTS, Farzad was involved in various research initiatives, developing and deploying IoT solutions for the university's industry partners, including a low-power IoT hub equipped with multiple radio technologies and satellite communication capabilities for the monitoring of remote fuel tanks. His research interests are centered around wireless sensing and communication methods, consensus algorithms, and distributed estimation techniques.

MASSIMO PICCARDI (SM'05) received the MEng and PhD degrees from the University of Bologna, Bologna, Italy, in 1991 and 1995, respectively. He is currently a Full Professor of computer systems with University of Technology Sydney, Australia. His research interests include natural language processing, computer vision and pattern recognition and he has co-authored over 200 papers in these areas. Prof. Piccardi is a Senior Member of the IEEE, a member of its Computer and Systems, Man, and Cybernetics Societies, a member of the International Association for Pattern Recognition, and a member of the Association for Computational Linguistics. He presently serves as an Associate Editor for the IEEE Transactions on Big Data.

MEHRAN ABOLHASAN (Senior Member, IEEE) completed his B.E in Computer Engineering and PhD in Telecommunications on 1999 and 2003 respectively at the University of Wollongong. Prof. Abolhasan has over 20 years of experience in R&D and serving in various research leadership roles. Some of these previous roles include serving as the Director of Research programs for the Faculty of Engineering and IT, Deputy Head of School for Research at the School of Electrical and Data Engineering and Lab Director for Telecommunication and IT Research Institute at University of Wollongong. He is currently the leader of intelligent Networks and Applications Lab within the Global Big Data Technology Centre at the Faculty of Engineering and IT at University of Technology Sydney.

Professor Abolhasan has authored over 180 international publications and has won over seven million dollars in research funding. His current research Interests include: 5G/6G Wireless Networks, Software Defined Networking, Tactile Internet, Intelligent Transportation Systems (ITS), Internet of Things (IoT), Wireless Mesh, Wireless Body Area Networks and Sensor networks.

DANIEL FRANKLIN completed his PhD in Telecommunications Engineering, "Enhancements to Channel Models, DMT Modulation and Coding for Channels Subject to Impulsive Noise", at the University of Wollongong in 2007 and also holds a Bachelor of Engineering (Electrical - Honours I, University of Wollongong, 1999). He is currently a Senior Lecturer in the School of Electrical and Data Engineering at the University of Technology Sydney. His current research and commercial interests are split between radiation engineering - including positron emission tomography, computed tomography, radiotherapy, radiation dosimetry and pharmacokinetic modelling - and telecommunications engineering - including security applications and network protocols, multimedia, image and signal processing, wired and wireless communication systems, and communications protocols.

JUSTIN LIPMAN Justin Lipman (S'94, M'04, SM'12) received a PhD in Telecommunications Engineering from University of Wollongong, Australia in 2004. He is an Industry Associate Professor at the University of Technology Sydney (UTS) and a visiting Associate Professor at Hokkaido University's Graduate School of Engineering. He is the Director of Research Translation in the Faculty of Engineering and IT and is Director of the RF Communications Technologies (RFCT) Lab - where he leads industry engagement in RF technologies, Cybersecurity, Privacy Preserving Technologies, Internet of Things and Tactile Internet. He serves as committee member in Standards Australia contributing to International IoT standards. He was previously the Deputy Chief Scientist of the Food Agility Cooperative Research Center. Prior to joining UTS, over a 12 year period, he held a number of senior management and technical leadership roles at Intel and Alcatel driving research and innovation, product development, architecture and IP generation. He is an IEEE Senior Member. His research interests are in enabling "things" to be adaptive, connected, distributed, ubiquitous and secure.