Information Sciences 476 (2019) 357–372

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Secure Multi-Party Computation: Theory, practice and applications

Chuan Zhao[a,b,*], Shengnan Zhao[c], Minghao Zhao[d], Zhenxiang Chen[a,b], Chong-Zhi Gao[e], Hongwei Li[f], Yu-an Tan[g]

[a] Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China
[b] School of Information Science and Engineering, University of Jinan, Jinan 250022, China
[c] School of Computer Science and Technology, Shandong University, Jinan, China
[d] School of Software, Tsinghua University, Beijing, China
[e] School of Computer Science, Guangzhou University, China
[f] Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[g] School of Computer Science, Beijing Institute of Technology, China

ARTICLE INFO

Article history:
Received 15 May 2018
Revised 14 October 2018
Accepted 15 October 2018
Available online 16 October 2018

Keywords:
Secure Multi-Party Computation
Generic protocol
Cloud security
Secure outsourcing
Privacy-preserving technology

ABSTRACT

Secure Multi-Party Computation (SMPC) is a generic cryptographic primitive that enables distributed parties to jointly compute an arbitrary functionality without revealing their own private inputs and outputs. Since Yao's seminal work in 1982, 30 years of research on SMPC has been conducted, proceeding from pure theoretical research into real-world applications. Recently, the increasing prevalence of the newly emerging technologies such as cloud computing, mobile computing and the Internet of Thing has resulted in a re-birth of SMPC's popularity. This has occurred mainly because, as a generic tool for computing on private data, SMPC has a natural advantage in solving security and privacy issues in these areas. Accordingly, many application-oriented SMPC protocols have been constructed. This paper presents a comprehensive survey on the theoretical and practical aspects of SMPC protocols. Specifically, we start by demonstrating the underlying concepts of SMPC, including its security requirements and basic construction techniques. Then, we present the research advances regarding construction techniques for generic SMPC protocols, and also the cutting-edge approaches to cloud-assisted SMPC protocols. Then, we summarize the concrete application-oriented protocols that are currently available, and finally, we present a discussion of the current literature and conclude this survey.

1. Introduction

With the rapid development of the Internet of Things, mobile computing, big data and cloud computing, people's lifestyles are undergoing a dramatic change. These technologies provide a new model for the collection, storage, dissemination, and processing of information, which offers great convenience for the whole society as a whole. To share resource provided by these different techniques, the cooperative computation between different organizations or individuals has become increasingly frequent. Owners of various types of data, through cooperative computation, would like to integrate their

* Corresponding author at: School of Information Science and Engineering, University of Jinan, Jinan 250022, China.
E-mail address: ise_zhaoc@ujn.edu.cn (C. Zhao).

https://doi.org/10.1016/j.ins.2018.10.024
0020-0255/

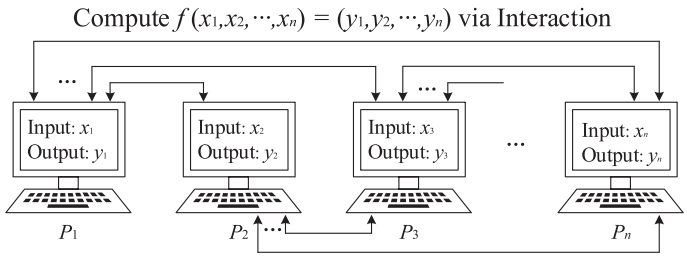

Fig. 1. Diagram of Secure Multi-Party Computation.

resources without leaking their own confidential data, thereby obtaining more valuable information. For example, the data collected from different sensors in the IoT system may be aggregated to generate the targeted result; the cloud and the clients may cooperate to provide appropriate services. At the same time, however, the leaks of private information and secret data frequently occur. Such security threats severely limit the application and popularization of these data processing technologies. Consequently, the development of privacy-preserving data processing methods in multi-party setting has become an urgent task.

As a core technology in the field of information security, cryptography plays a central role in every aspect of ensuring security and provides theoretical foundations and technical support for data privacy, integrity and authentication. Among the cryptography research, Secure Multi-Party Computation (SMPC) is a generic cryptographic primitive that enables jointly computing in a privacy-preserving manner. As an important fundamental research topic in the field of cryptography, SMPC addresses the problem of cooperative computation performed on private data from several participants in a secure fashion within a distributed computing scenario. Informally, in the SMPC scenario, two or more parties holding private inputs wish to compute some joint functionality using these inputs. In this task, maintaining security requires each participant to obtain its own objective output and nothing else. Here, functionality is a general concept that may refer to almost any cryptographic task, such as encryption, authentication, a zero-knowledge proof, a commitment schemes, oblivious transfer, and other none-cryptographic protocol (i.e., application-oriented task encompasses contract signing, electronic voting, machine learning, genome data processing and so on). It can be said that SMPC is the most general and basic theoretical research topic in the field of cryptography. Any cryptographic task that involves multiple parties can be viewed as an SMPC task. Intensive research on SMPC has promoted the development of fundamental primitives such as zero-knowledge proofs, oblivious transfer, and secret sharing. SMPC establishes a theoretical basis for provable security in interactive protocols, and has greatly facilitated the development of modern cryptography.

This survey aims to provide a comprehensive summary of the state-of-the-art security solutions for SMPC. Before proceeding with the descriptions of the individual schemes and their properties, we present the general framework in Section 2, including basic concepts, research areas, security requirements, and various building blocks. Generic constructions for secure computation, including secure protocols for both semi-honest and malicious models, and state-of-the-art construction techniques, are discussed in Section 3. Then, in Section 4, we discuss cloud-assisted SMPC solutions. Section 5 covers application-oriented computing for various computational domains. Finally, we conclude this survey in Section 6.

2. Secure Multi-Party Computation: threats, security requirements, and building blocks

In an SMPC setting, two or more parties P_i $(i = 1, \ldots, n)$ with private inputs x_i in a distributed computing environment wish to jointly and interactively compute an objective functionality $f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$ based on their private inputs. Once the computation is complete, each party P_i should obtain its own corresponding output y_i without acquiring any other information (as shown in Fig. 1). SMPC is aimed at constructing secure protocols to allow multiple mutually distrust ed participants to jointly compute an objective function over their inputs while ensuring correctness of the outputs and maintaining the privacy of the inputs even in the face of dishonest behaviour. The concept of SMPC was first introduced by Yao in the 1980s. For the past three decades, SMPC has been intensively researched via a series of theoretical and practical research works.

2.1. Research areas

SMPC covers a large research scope from theoretical to practical concern, including SMPC building blocks, generic SMPC protocols, cloud-assisted SMPC, and application-oriented protocols. A diagram presenting a high-level overview of SMPC research areas is shown in Fig. 2.

The theoretical research includes studies of security models, feasibility and complexity, among other topics. The purpose of practical research is to transform the elegant theoretical ideals emerging from such studies into concrete SMPC protocols that address real-world security problems. Practical SMPC research focuses on efficiency. It involves the study of methods and techniques for improving the efficiency of both generic and specific protocols in terms of computational cost, communication cost, and the number of rounds of interaction.

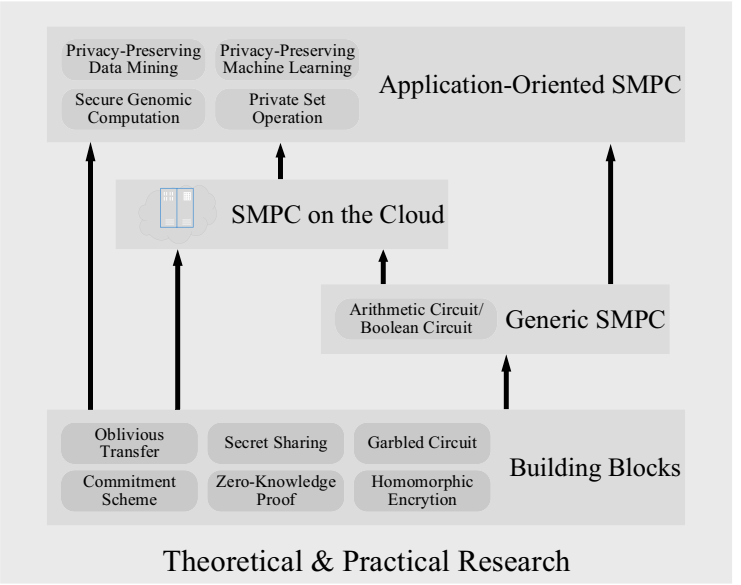

Fig. 2. Research areas related to Secure Multi-Party Computation.

Generic SMPC consists of the conversion of an objective computation task into an arithmetic or a Boolean circuit C, the decomposition of C into a series of basic combinations of arithmetic gates or logic gates and the subsequent design of an SMPC protocol for the circuit under a specific security model. The underlying cryptograph ic tools used for this purpose include secret sharing, homomorphic encryption, oblivious transfer, and garbled circuits. The developed circuit performs the computation layer by layer and eventually completes the computation task in a secure manner. Researchers in recent years have concentrated on increasing the efficiency of such protocols under various security models. At the same time, to balance the trade-off between efficiency and security, some new models with a lower security level have been proposed.

Research on cloud-assisted SMPC is aimed at improving protocol efficiency by making use of resources available on cloud severs to significantly reduce of computation and communication overheads.

Application-oriented SMPC includes studies in the fields of private set operations, privacy-preserving machine learning, data mining, and secure genomic computation.

All the protocols mentioned above involve fundamental building blocks, such as oblivious transfer, secret sharing, coin tossing, homomorphic encryption, commitment schemes, zero-knowledge proofs. These building blocks lay the foundation for both theoretical and practical SMPC.

2.2. Threats and security requirements

SMPC considers the potential for dishonest behaviour by dishonest participants. The purpose of an attack from such a party may be to discover the private information of others or to cause an error during the computation task. During an SMPC task an adversary may control some subset of the participants. An adversary-controlled party referred to as a corrupted participant, completely follows the adversary's instructions in executing the protocol. To formally prove that a protocol is secure, researchers have proposed numerous definitions of security, which mainly attempt to guarantee the following important security requirements:

- *Privacy*: No party should able to obtain any information other than its own output and information that can be derived from its own input and output.
- *Correctness*: The output received by each participant should be correct.
- *Independence of input*: The input selected by a corrupted participant must be independent of the inputs of the honest participants.
- *Guarantee of output*: Corrupted participants should not be able to prevent honest participants from receiving their own outputs. In other words, the adversary should not be able to interrupt the computation by running a "denial of service" attack.
- *Fairness*: Corrupted participants should receive their own outputs if and only if the honest participants receive their own outputs.

Ideal/Real simulation paradigm. The above list does not constitute a definition of security; it merely enumerates certain requirements that a security protocol must satisfy. By contrast, a definition of security should include all important security

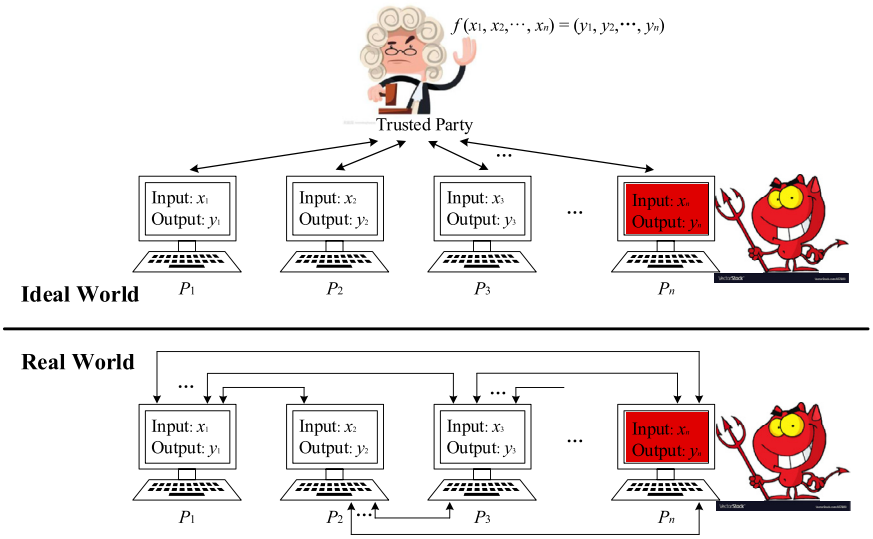

Fig. 3. Ideal/Real simulation paradigm.

requirements and cover all possible adversarial attacks. In addition, it should be simple and sufficiently succinct for practical use.

The current standard definition of security is as follows. Consider an "ideal world" in which there is an external trusted party to help participants perform their computations. In such a world, participants can simply send their inputs to the trusted party through a completely secure channel, and then the trusted party computes the selected functionality and sends the corresponding output to each participant independently and secretly. Because the only action performed by the participants is to send their inputs to the trusted party, the only freedom left to the adversary is to select the inputs of the corrupted parties. Thus, in an ideal world, all the security requirements listed above are satisfied.

However, in the "real world", no external party can be trusted by all participants. Instead, the participants jointly execute a protocol without any external assistance. If this protocol can simulate the results obtain in the ideal world described above, i.e., if the real protocol executed by the participants can guarantee that no adversary can do more harm than is possible during execution in the ideal world, then this protocol is said to be secure. This condition can be expressed as follows: for any adversary who performs a successful attack in the real world, an adversary in the ideal world must always be able to successfully execute the same attack. However, the attack cannot be successfully executed in the ideal world; therefore, all real-world attacks on protocol execution must fail.

A more intuitive definition of security is given below. The security of a protocol can be evaluated based on the comparison of two joint output distributions: that obtained by the adversary and the honest participants in the real world and that obtained through execution in the ideal world. That is, for an adversary executing any attack against a real protocol, an adversary executing the same attack in the ideal world is considered, and the security criterion is satisfied if the joint input/output distribution of the adversary and participants in the ideal world is the same as the corresponding distribution in the real world. Under this condition, the execution of the real protocol "simulates" the effect of the ideal world. This definition of security, which is illustrated in Fig. 3, is called the ideal/real simulation paradigm.

2.3. Security model

The above informal definition of security does not take the security model into account. In the study of SMPC, the security model defines the capabilities of the adversary. The security of a protocol is meaningful only when it is discussed under a specific security model. A protocol is considered secure only if it is able to resist any adversarial attacks under the corresponding security model. Based on the behaviour of the adversary, security models can be divided into the following different types.

- *Semi-honest adversary model*: In the semi-honest adversary model, corrupted participants must execute the protocol correctly. However, the adversary can obtain comprehensive information on the internal status of each corrupted part y (including the transcripts of all received messages) and will attempt to use this information to obtain additional information that should remain confidential. Although this is a very weak adversary model, it captures many practical scenarios. For instance, suppose that several organizations or companies want to collaborate on some task. They will no behave dishonestly due to the reputation loss or negative press they would suffer if caught cheating; however, they would like to acquire as much of other participants' private informationas possible to gain an advantage.

Table 1
Garbled gate — an example.

(a) An OR gate g

Input wire w_1 (u)	Input wire w_2 (v)	Output wire w_3 (u OR v)
0	0	0
0	1	1
1	0	1
1	1	1

(b) An OR gate with garbled keys

Input wire w_1 (u)	Input wire w_2 (v)	Output wire w_3 (u OR v)
k_1^0	k_2^0	k_3^0
k_1^0	k_2^1	k_3^1
k_1^1	k_2^0	k_3^1
k_1^1	k_2^1	k_3^1

(c) A garbled OR gate $G(g)$

Ciphertexts in a garbled OR gate (with random permutation)
$\mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^0}(k_3^0))$
$\mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^1}(k_3^1))$
$\mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^0}(k_3^1))$
$\mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^1}(k_3^1))$

- *Malicious adversary model*: In the malicious model, corrupted participants may arbitrarily deviate from the protocol's specifications based on the adversary's instructions. A protocol that is secure against a malicious adversary can guarantee that any adversarial attacks will fail. However, to achieve this level of security, a heavy price must be paid in terms of the protocol's efficiency. This is the preferred model for joint computation tasks performed among competitors. The participants may behave dishonestly in order to cause an error, or the generation of a faulty computation result, even though they may be caught cheating, so as to maximize their benefits.
- *Covert adversary model*: The abovementioned semi-honest adversary model is too weak, but requiring security under the malicious adversary model results in protocols that are too inefficient. To overcome these difficulties, covert adversary model is proposed. A covert adversary may exhibit malicious behaviour, but it has a given probability of being caught cheating by honest participants. This model is representative of many financial or political settings, in which honest behaviour cannot be assumed, but the companies and institutions involved cannot afford the loss of reputation associated with being caught cheating. In this model, the adversary must weigh the risk of being caught against the benefits of cheating.

2.4. Building blocks

In this section, we briefly describe some of the building blocks used in SMPC, including garbled circuits, oblivious transfer and its extensions, the cut-and-choose paradigm, commitment schemes, and homomorphic encryption.

2.4.1. Garbled circuit

Yao's protocol [48] is a garbled circuit-based approach for secure two-party computation (S2PC). As a fundamental cryptographic primitive, the garbled circuit plays an important role in realizing generic protocols. A garbled circuit for a given objective function has the following properties:

- Each circuit wire corresponds to two garbled keys: one for bit 0 and one for bit 1.
- If each input wire of the circuit is given a garbled key, the entire garbled circuit can be evaluated obliviously to obtain the output value of the circuit without leaking any other information.

To better explain the construction of a garbled circuit, we first take an OR gate (see Table 1a) as an example to describe in detail how to construct a single garbled gate. Let the input wires of the OR gate g be denoted byas w_1 and w_2, and let the output wire be denoted by w_3. The input bits for w_1 and w_2 are u and v ($u, v \in \{0, 1\}$), respectively. The garbled version of gate g can be constructed as follows:

- Each circuit wire is assigned two garbled keys. Specifically, for circuit wire w_1 (resp. w_2 and w_3), two keys $(k_1^0, k_1^1 \in \{0,1\}^n)$ (resp. $(k_2^0, k_2^1 \in \{0,1\}^n)$) and $(k_3^0, k_3^1 \in \{0,1\}^n)$, where n is the security parameter) are randomly selected, where the first corresponds to bit 0 and the second corresponds to bit 1 (see Table 1b and Fig. 4).
- For each row in Table 1b, the keys in the third column are double-encrypted using the keys in the first two columns. Specifically, given the key values k_1^u and k_2^v, $\mathsf{Enc}_{k_1^u}(\mathsf{Enc}_{k_2^v}(k_3^{u\mathsf{OR}v}))$ is calculated (in this paper, we use $\mathsf{Enc}_k(m)$ to denote the encryption of a plaintext m under a key k; if the random number r used in the encryption algorithm needs to be

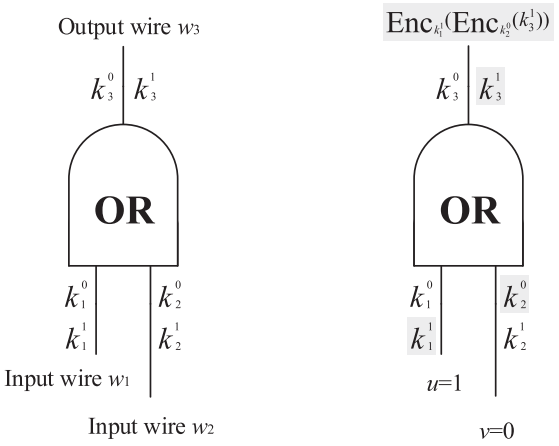

Fig. 4. A garbled OR gate.

explicitly specified, the encryption is denoted by $\mathrm{Enc}_k(m;r)$). The double-encryption scheme used here is a symmetric encryption scheme with a "hidden domain" and a "valid verifiable domain". After double-encryption operation is performed on each record in Table 1b, four ciphertexts are obtained. Random permutation of these four ciphertexts yields a garbled version of gate g, denoted by $G(g)$. (see Table 1c).

Given the keys k_1^u and k_2^v, one can try to decrypt each ciphertext in the garbled gate $G(g)$. Because the encryption scheme itself has the property of a valid verifiable domain, it is easy to confirm which plaintext is the corresponding $k_3^{u\mathrm{OR}v}$. No additional information is revealed during this procedure. Please see Fig. 4 for an intuitive illustration of the decryption process.

A garbled circuit can be directly constructed by combining garbled gates layer by layer. Specifically, because the output wire of the previous gate in the circuit serves as an input wire to the next circuit gate, the garbled key that is output by the previous garbled gate is also a garbled key for the input to the next garbled gate. Therefore, once two garbled keys have been selected for each circuit wire, a garbled version of each circuit gate can be constructed layer by layer as described above with regard to the construction of the garbled OR gate $G(g)$, and thus, an entire garbled circuit can eventually be constructed.

After the construction of a garbled circuit is complete, if each *input wire* of the circuit is given a garbled key, the first layer of the circuit can be successfully decrypted. After each garbled gate in the first layer is successfully decrypted, the result is the garbled keys on the output wires. Because these output keys are also the input keys to the garbled gates in the next layer, the decryption of the garbled circuit can continue in a layer by layer manner. Finally, the output value of the circuit is obtained.

2.4.2. Oblivious transfer and its extensions

The standard 1-out-of-2 oblivious transfer operation is an S2PC task in which one participant, the sender (denoted by S), holds two n bits long strings, x_0 and x_1, and the other participant, the receiver (denoted by R), selects between the two data strings held by the sender via a selection bit, $\sigma \in \{0, 1\}$. After the transfer is complete d, S has no output, and cannot know which string R received, whereas R has obtained the string x_σ corresponding to σ but knows nothing about the other string, $x_{1-\sigma}$. Oblivious transfer, which is denoted as the function $\mathcal{F}_{ot}$, can be formally defined as shown in the protocol diagram presented in Fig. 5.

Oblivious transfer extension. Oblivious transfer is used extensively in SMPC protocols. For instance, in garbled circuit based SMPC protocols, each input wire must run an oblivious transfer protocol. In secret sharing based SMPC protocols, each AND gate must run at least one oblivious transfer protocol. Consequently, millions of oblivious transfer instances must be run in an SMPC protocol, which leads to an excessively high operation cost. To solve this problem, a method of extending oblivious transfer has been proposed. An oblivious transfer extension protocol works by running a few "base" oblivious transfer instances. The number of "base" instances depends on the security parameters used. Based on these "base" instances, we can further obtain many more oblivious transfer instances can be further obtained through the use of only computationally cheap symmetric cryptographic operations. In this way, oblivious transfer extensions can be achieved with high efficiency.

2.4.3. Homomorphic encryption

Homomorphic encryption is a form of encryption that allows computations on ciphertexts to generate an encrypted result that, when decrypted, matched the result that would have been obtained if the computations had been performed on the plaintext. With homomorphic encryption, computations yielding correct results can be performed on encrypted data (e.g., arithmetic operations, searches, comparisons, and edits). Moreover, the data are processed in ciphertext form throughout

Functionality $\mathcal{F}_{ot}$

- **Input:**
  - S inputs two strings $x_0, x_1 \in \{0,1\}^n (n \in \mathbb{N})$.
  - R inputs a selection bit $\sigma \in \{0,1\}$.
- **Output:**
  - S has no output.
  - R outputs x_σ, and learns nothing about $x_{1-\sigma}$.

Fig. 5. 1-out-of-2 oblivious transfer functionality $\mathcal{F}_{ot}$.

the entire process and do not require decryption. This special property of homomorphic encryption plays a key role in the construction of cryptographic schemes and security protocols.

According to the various restrictions on the functions supported by different encryption schemes, homomorphic encryption schemes can be divided into somewhat homomorphic and fully homomorphic schemes. Somewhat homomorphic encryption algorithms support only certain specific functions (such as limited addition and multiplication operations). Somewhat homomorphic algorithms are easy to implement, and their computational overhead is small; thus, such algorithms are already being used in practice. By contrast, fully homomorphic algorithms can support arbitrary functionality, but their computational overhead is enormous; thus, they are still some distance from pracitcal applications. We now present detailed descriptions of *C*-homomorphic encryption and fully homomorphic encryption (FHE).

C-homomorphic encryption and Fully Homomorphic Encryption. Informally, a *C*-homomorphic encryption scheme for a set of functions *C* can be described as follows. Given the plaintext m and the corresponding ciphertext $e = \mathsf{Enc}(m)$, where Enc is the encryption algorithm, for an operation $f \in C$ on the plaintext, there will be a corresponding operation F on the ciphertext that satisfies $F(e) = \mathsf{Enc}(f(m))$, namely: $F(\mathsf{Enc}(m)) = \mathsf{Enc}(f(m))$. Based on the concept of *C*-homomorphic encryption, an FHE algorithm is an encryption algorithm that is *C*-homomorphic with compactness for any set of functions *C*. In other words, the output ciphertext length is polynomial in the input length regardless of what kinds of complex operations are performed on the ciphertexts. The formal definition of an FHE scheme is given as follows.

Definition (Fully Homomorphic Encryption): An FHE scheme consists of four algorithms: KeyGen, Enc, Dec, and Eval. The first three algorithms must be computationally efficient, and the runtime must be polynomial in the security parameter, usually characterized as the bit length of the key. These four algorithms are described as follows:

- KeyGen: This algorithm takes the security parameter λ as input and generates a key pair consisting of one public key and one private key; this process is denoted by $\mathsf{KeyGen}(\lambda) \to (pk, sk)$.
- Enc: Given the public key pk, this algorithm maps a message m to the corresponding ciphertext c; this process is denoted by $\mathsf{Enc}(pk, m) \to c$.
- Dec: Given the private key sk, this algorithm maps a ciphertext c to the original message m; this process is denoted by $\mathsf{Dec}(sk, c) = m$.
- Eval: Given a set of functions C, for each Boolean function $f \in C$ and deletedfor any ciphertexts $c_1, c_2, \ldots, c_t$, where $c_i = \mathsf{Enc}(pk, m_i), (i = 1, 2, \ldots, t)$, Eval outputs the corresponding ciphertext of $f(m_1, m_2, \ldots, m_t)$ under the public key pk; this process is denoted by $\mathsf{Eval}(pk, f, c_1, c_2, \ldots, c_t) = \mathsf{Enc}(pk, f(m_1, m_2, \ldots, m_t))$.
  An encryption scheme $\epsilon = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is called an FHE scheme, if for all Boolean functions, the ciphertexts output by Eval can be correctly decrypted and there exists a polynomial such that the length of $\mathsf{Eval}(pk, f, c_1, c_2, \ldots, c_t)$ is less than $p(\lambda)$.

With an FHE algorithm, a data user can outsource encrypted data to a server to directly perform various operations on those data without exposing any confidential information those data contain. The supported operations include querying and modifying the encrypted data. Once the operations on the encrypted data operation have been completed, the results are returned to the data user, who decrypts the received encrypted data using the corresponding decryption key. Throughout the entire process, the server helps the data user to perform complex operations without obtaining any information from the user's data.

Threshold Fully Homomorphic Encryption and multi-key Fully Homomorphic Encryption. When applied in SMPC, FHE usually takes the form of *multi-user* FHE. In this scenario, multiple users encrypt their own private data. All desired functions are computed on these encrypted data. Finally, the users decrypt the results to obtain the corresponding plaintexts.

There are two main types of multi-user FHE: threshold FHE and multi-key FHE. In the former, the key generation process is an interactive SMPC protocol in which multiple users jointly negotiate a public key and acquire a *secret share* of the corresponding private key. Then, all users use the common public key to encrypt their private data and send them to a server with powerful computing capabilities. This server performs arbitrary function computations on the received ciphertexts. Finally, the users interactively apply a decryption protocol to obtain the plaintext of the computation result.

Multi-key FHE schemes differ from threshold FHE schemes as follows. In the former case, each user has a *public-private* key pair. Users use their own public keys, instead of using a common public key, to encrypt their private data. In this way, each user can encrypt the personal data that he wishes to outsource using his public key and then store it on an external server. When the users request the server to compute a specific function, the server performs the computations locally using the stored encrypted data, without requiring any user interaction. After the computations have been completed, the users interactively apply the decryption protocol to recover the plaintext of the computation result.

3. Generic Secure Multi-Party Computation

A generic SMPC protocol allows two or more parties to jointly compute an arbitrary functionality in a secure manner. Such a protocol is not constructed to solve a *specific* problem; it is for a *general-purpose* protocol. In this section, we survey several important approaches in the literature.

3.1. Secure computation in the semi-honest model

Yao's Protocol. In FOCS 1986 [48], Andrew Yao proposed a constant-round, semi-honest protocol for generic S2PC based on garbled circuits and oblivious transfer. In this methodology, the objective functionality is represented by a Boolean circuit. One of the participants (called the Generator, denoted by Gen) is required to construct a garbled version of the circuit, while the other participant (called the Evaluator, denoted by Eval) is responsible for evaluat ing the garbled circuit obliviously. As a result, both parties obtain the correct output while preserving the privacy of their individual inputs. Specifically, suppose that Gen holds a secret input x and that Eval holds a secret input y. They do not trust each other and want to jointly compute the objective functionality $f(x, y)$ without revealing their individual secret information. The protocol diagram presented in Fig. 6 offers a more detailed description of this seminal work.

In Yao's protocol, each circuit gate involves only a constant number of symmetric cryptographic operations (except for the input gates, which also involve asymmetric operations due to the invocation of oblivious transfer (OT)); consequently, the circuit is very efficient. However, because the required circuit may be very large, it had previously been considered impractical to use a circuit-based protocol to accomplish an SMPC task. The emergence of Yao's protocol thus attracted widespread attention in the cryptograph ic academic community. One of the main research directions related to this approachinvolves optimizing the method of garbled circuit construction to effectively improve the computational efficiency [46].

Yao's seminal work was previously considered solely as a theoretical result; however, in 2004, Malkhi et al. [34] implemented a circuit-based generic secure computation platform, known as Fairplay. Since then, additional research works have emerged addressing the implementation of SMPC programming tools, which have greatly facilitated efforts to improve the efficiency of generic SMPC protocols and apply them in practice.

3.2. Achieving security against malicious adversaries

Yao's protocol achieves security only against semi-honest adversaries, which is insufficient in practice. To resist attacks by malicious adversaries, early solutions typically applied generic zero-knowledge proofs to constrain participants' behaviours. Although these techniques have contributed to the design of constant-round secure protocols, such protocols are less than practical. Literatures concerning ways to avoid the use of generic zero-knowledge proofs also exists , but the round complexity of these protocols is linear in the circuit depth. We summarize several popular technologies for constructing efficient SMPC protocols in the malicious model.

Cut-and-choose paradigm. The cut-and-choose paradigm was first proposed by Lindell and Pinkas [30], with the goal of constraining a potentially malicious generator Gen to act in a semi-honest manner. The cut-and-choose paradigm forces Gen to construct a garbled circuit that computes the objective functionality correctly and reveals nothing more than the expected output. Specifically, Gen is required to construct s instances of mutually independent garbled circuits (where s is a statistical security parameter) and send them to the circuit evaluator, Eval. Eval then opens some number of these instances (called the check circuits) to detect whether these garbled circuits are all correctly constructed. If so, Eval assumes that there is a high probability that the remaining garbled circuits (called the evaluation circuits), are also constructed correctly. Otherwise, Eval aborts the protocol. Then, both parties compute each computation circuit as in Yao's protocol, and take the majority of the computation results as the protocol output.

Yao's protocol

Input: The input of GEN is $x \in \{0,1\}^n$, and the input of EVAL is $y \in \{0,1\}^n$, where $n \in \mathbb{N}$.

Auxiliary Input: The Boolean circuit C of the objective functionality $f(\cdot,\cdot)$.

Protocol:

Step 1: GEN constructs a garbled version of Boolean circuit C.

Step 2: For each circuit input line to which EVAL's input corresponds, GEN and EVAL run a 1-out-of-2 oblivious transfer protocol. The input of GEN consists of the two garbled keys associated with the input line; the input of EVAL is the corresponding input bit on that input line. After all executions of OT protocols have been completed, EVAL gets the garbled keys corresponding to its input y.

Step 3: GEN sends the garbled circuit, and the garbled keys corresponding to its own input x, to EVAL.

Step 4: Given a garbled key on each circuit input line, EVAL computes the garbled circuit, and obtains the computation result $f(x,y)$.

Step 5: (Optional) EVAL sends the computation result $f(x,y)$ to GEN.

Fig. 6. Yao's protocol (secure computation of $f(\cdot,\cdot)$).

Applying the cut-and-choose approach to Yao's protocol is an efficient way to achieve security in the malicious model. The cut-and-choose technique replaces complex and inefficient zero-knowledge proofswith a new security parameter, the statistical security parameter s. This parameter is used to characterize cheating probabilities that depend not on any cryptographic computational assumptions but only on the number of garbled circuits.

While preventing a malicious GEN from constructing incorrect circuits, the cut-and-choose technique also introduces some new issues related to concerns such as participant input consistency, selective failure attacks, and the probability of cheating. The cut-and-choose paradigm does not guarantee that an adversary can never succeed at cheating. Roughly speaking, if $s/4$ circuits are faulty and are not selected as the check circuits, the adversary will succeed in cheating with a probability of $2^{-s/4}$ (under the assumption that each circuit has an equal probability of being randomly chosen as a check circuit or evaluation circuit). In this case, if a cheating probability of 2^{-40} is acceptable, 160 garbled circuit instances must be constructed. However, if the cheating probability were optimal, i.e., 2^{-s}, 40 copies would be sufficient to achieve a cheating probability of 2^{-40}. Therefore, the cheating probability is highly important in determining at protocol's efficiency. Many research works have been developed to reducing the cheating probability. Lindell and Pinkas [31], relying on the decisional Diffie–Hellman (DDH) assumption, proposed a new method that ensures the consistency of GEN's input with a cheating probability of $2^{-0.311s}$. Shen et al. [45] succeeded in reducing the cheating probability to $2^{-0.32s}$ by checking 60% of the circuits and pointed out that this was optimal for the cut-and-choose method. However, this result relies on the assumption that EVAL checks some percentage of the circuits and takes the majority result of the remaining circuits as the output. In CRYPTO 2013, Lindell [28] proposed a new protocol ensures that GEN will succeed in cheating if and only if all of the circuits to be checked are correctly constructed, and all of the circuits to be evaluated are faulty and evaluate to the same fake value.

This method achieves an optimal cheating probability of 2^{-s}. The core idea is that if Eval finds that Gen is cheating, Eval can obtain Gen's input via a punishment mechanism, allowing it to compute $f(x, y)$ on its own.

In addition to the above methods, researchers have also exploited other techniques to improve the efficiency of cut-and-choose based protocols. For example, to achieve good amortized efficiency, researchers have proposed batched cut-and-choose approaches, in which a batch of instances of the same functionality are efficiently executed between the same parties using possibly different inputs.

3.3. Other state-of-the-art methods for generic SMPC

LEGO. Unlike the method of Lindell and Pinkas [30], in which the cut-and-choose technique is applied to the entire circuit, the LEGO method proposed by Nielsen and Orlandi [38] relies on a cut-and-choose test performed at the gate level for better asymptotic efficiency. This method requires the circuit generator Gen to send many NAND gates to the receiver. The receiver Eval opens a random subset of these gates to be checked. If the check succeeds, Eval randomly permutes the unopened gates into buckets, representing redundant NAND gates. With the help of Gen, Eval solders together the gates within each bucket and then solders the buckets together to form a circuit that will correctly compute the function even if a minority of the gates in each bucket are faulty.

However, LEGO invokes public-key primitives for each gate, which is a drawback for the efficiency of the protocol. In addition, LEGO is not compatible with known optimization techniques for garbled circuits. To overcome these disadvantages, Frederiksen et al. [13] proposed MiniLEGO, in which the Pedersen commitment scheme used in LEGO is replaced with a much more efficient XOR-homomorphic commitment scheme while achieving the same level of statistical security. Unlike in LEGO, because MiniLEGO uses standard garbled circuits, all the known optimization methods for garbled circuits are applicable. A more recent independent work [39] has presented an implementation called TinyLEGO [14]that demonstrates the efficiency of the LEGO approach in practice. However, the security of the protocol still depends on the majority of the circuits in every bucket being correct. Recently, Zhu and Huang [50] proposed a faster LEGO protocol, which they call JIMU. The newly proposed protocol does not rely on homomorphic commitments but is able to guarantee security as long as at least one of the garbled gates in each bucket is correct. The faulty gate detection rate is double that of previous works.

Preprocessing model. The preprocessing model is a novel method for improving the online execution efficiency of secure computation protocols. It works by dividing such a protocol into two phases: a preprocessing phase and an online phase. In the preprocessing phase, the participants' inputs need not be known; consequently, preprocessing can be performed at any time prior to the objective computation task. Bendlin et al. [2] showed that given a preprocessing functionality, an arithmetic circuit can be efficiently computed with unconditional security even over a large field. The computational complexity of each party is a constant multiplied by the complexity of the work that needs to be done in the clear. Damgård et al. [9] maintained the original conclusion of [2], and reduced computational and communication complexity of the online phase to be linear in the number of participants, with the computational cost per participant being only slightly larger than that of directly computing the circuit. Nielsen et al. [37] used an OT-based approach and achieved better performance via OT extension. They related the outputs and inputs of the OTs, and implemented a practical two-party computation protocol under the malicious model. Damgård and Zakarias [10] presented a secure protocol for securely computing a Boolean circuit in the presence of a dishonest majority. The computational burden for each participant is the same as that of computing the circuit directly (up to a constant factor). Recently, Damgård et al. [8] focused on Boolean circuits and introduced a new protocol called tiny table, in which the idea is to implement each gate using a scrambled version of its truth table. They achieved a high-efficiency S2PC protocol under the preprocessing model, with both the communication complexity and the preprocessed data size being linear in the circuit size $|C|$. The computational complexity of the proposed protocol is $O(s^{\epsilon}|C|)$, where s is the statistical security parameter and takes a constant value such that $\epsilon < 1$.

IPS compiler. Based on one-way functions, Goldreich et al. [17] showed how to compile a computation protocol that is secure against semi-honest adversaries into a protocol that is secure against malicious adversaries. This famous technique is known as the GMW compiler. In 2008, Ishai et al. [21] proposed a completely different structure called the IPS compiler, which uses multi-party agreement for the basic operations of the semi-honest adversaries (which are mostly honest in this case) and convert multi-party agreements with an honest majority into multi-party agreements with a dishonest majority under the malicious model. For computing the multiparty protocol, the underlying structure of the IPS compiler is a black box that works similarly to the GMW compiler for two- and multi-party scenarios. In addition, it offers better progressive complexity and results in a single round protocol in a non-interactive context. However, its constant is larger than that of the preprocessing model, and it is not as efficient as the preprocessing model in the online phase. Ishai et al. [22] used the IPS compiler to study the secure computation of arithmetic circuits on a limited ring in the presence of any malicious adversary. This approach can be used to execute only black-box calls on ring operations and standard cryptographic primitives and achieves unconditionality in the OT hybrid model. To ensure protocol security while minimizing black box and communication costs, Lindell et al. [29] optimized the watch list creation phase in the IPS compiler and proposed the use of a hidden security protocol under a given adversary model as the underlying component to achieve more efficient progressive security. In addition, Lindell et al. [29] analysed the specific efficiency of the IPS compiler and demonstrated that it could be the most efficient protocol in some cases.

Table 2
Comparison of different methods for generic SMPC.

Construction approach	Scheme	Accuracy	OTs	Rounds of communication	Complexity
Cut-and-choose	[30]	$2^{-s/17}$	$O(max(s, n))$	$O(1)$	$O(sn)$
	[31]	$2^{-0.311s}$	$O(n)$	12	$7sn + 22n + 7s + 5$
	[45]	$2^{-0.32s}$	$O(n)$	$O(1)$	$O(n + s)$
	[28]	2^{-s}	$O(n)$	$O(1)$	$O(sn)$
LEGO	[38]	2^{-s}	$O(n + s)$	$O(1)$	$O(s\|C\|/log\|C\|)$
Preprocessing model	[9]	$2^{-\Theta(s)}$	-	-	$O(n^2/s \cdot \|C\|)$
	[10]	2^{-s}	-	-	$O(log(\|C\|) \cdot ploylog(s))$
	[8]	2^{-s}	-	-	$O(s^e\|C\|)$
IPS Compiler	[21]	-	$O(n)$	$O(d)$	$O(\|C\|) + ploy(s, d, log\|C\|)$

3.4. Comparison of different methods

In Table 2, a comparison is given among four different approaches for generic SMPC.

4. Cloud-assisted Secure Multi-Party Computation

While multiple techniques exist for constructing generic SMPC protocols, they are currently computationally intensive, causing SMPC to still be impractical. To achieve more computationally efficient SMPC, recent work has focused on developing secure techniques for outsourcing the most expensive parts of computational tasks to the cloud. Rather than simply treating the cloud as a trusted party, these outsourced protocols seek to use the cloud for computation without revealing any input or output values. By allowing parties to securely outsource their computations to a cloud provider, cloud-assisted SMPC offers a different but effective approach to making generic SMPC protocols practical and scalable.

4.1. New challenges

Kamara et al. [24] first proposed server-aided SMPC (as another term for cloud-assisted SMPC). The incorporation of a cloud server helps improve the run-time efficiency of SMPC protocols. However, it also introduces new challenges. In particular, it changes both the computation model and the security model of SMPC. In classic SMPC, it is implicitly assumed that the computations are executed in a *homogeneous* computing environment, in which all participants play similar roles. In cloud-assisted SMPC, the protocols run in a *heterogeneous* environment, which contains in addition to the parties evaluating the functionality an untrusted server that (1) provides no input to the function being computed and (2) does not receive any output from the computation task but (3) has access to a large amount of computational resources. This setting is referred to as server-aided SMPC or cloud-assisted SMPC, as shown in Fig. 7. In such a model, the parties evaluating the functionality are allowed to outsource heavy computation tasks for SMPC to the cloud server.

Although the new computation model significantly improves protocol efficiency, cloud-assisted SMPC faces new security challenges. For instance, the cloud server may collude with a subset of the parties to obtain additional information about other parties' inputs. Moreover, a malicious or "lazy" cloud server may return forged computations result to the outsourcing parties. Therefore, new techniques are required to guarantee the security of cloud-assisted SMPC.

4.2. Construction approaches

Cloud-assisted SMPC based on garbled circuits. In S2PC based on garbled circuits, one party is the circuit generator and the other is the circuit evaluator. In the cloud-assisted model, both the generation and evaluation of the garbled circuits can be outsourced to the cloud server. A party who outsources his computation task is called an outsourcing party, and other parties are called non-outsourcing parties.

In such an outsourcing setting, it is desirable to design SMPC protocols with linear complexity, which means that the computation and communication costs of the outsourcing parties increase linearly with the sizes of their inputs and outputs. To achieve this level of complexity, a non-collusion assumption is necessary. Although the security in the non-collusion model is weaker than that in the standard malicious model for SMPC, the non-collusion model is sometimes preferable because it supports a wide range of applications.

Kamara et al.'s seminal work [24] proposed two secure single-server-aided S2PC protocols in which the circuit evaluation task is outsourced to the cloud. Later, Carter et al. [5] considered outsourcing secure function evaluation for power-limited devices such as mobile phones. In contrast to the protocol presented in [24], in which the parties were assumed to have low computation capabilities but high bandwidth, the work presented in [5] considered a scenario in which two parties, a mobile device with low bandwidth and an application server, want to run S2PC protocols with the help of a cloud server. In the proposed protocol, the mobile device outsources the complex computations necessary for circuit evaluation to the cloud server. In [5], a primitive called outsourced oblivious transfer was introduced, which allows mobile devices to delegate the task of garbled key transmission to a cloud server. Mood et al. [35] mainly considered cloud-assisted protocols that

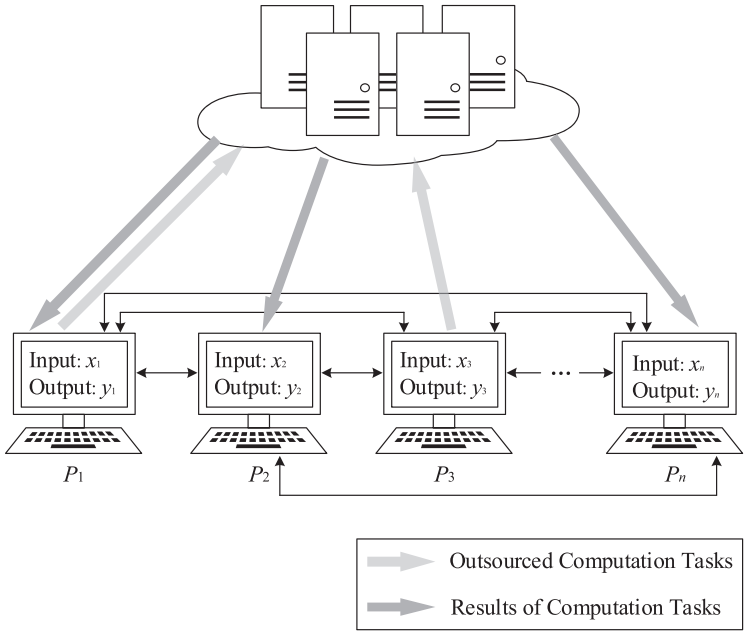

Fig. 7. Diagram of cloud-assisted SMPC.

reuse encrypted values within the cloud. These authors presented the notion of PartialGC, which allows encrypted values generated during garbled circuit computations to be reused.

In addition to the circuit evaluation task, the circuit generation task can also be outsourced to a cloud server. In contrast to the scenario proposed in [5], Carter et al. [4] considered a mobile device acting as a garbled circuit generator and succeeded in securely outsourcing the generation of garbled circuits to an untrusted cloud server.

The above works considered outsourcing protocols in a single-server-aided model. By contrast, Kerschbaum [25] presented a scheme in which garbled circuit generation is outsourced to multiple servers categorized as encryption servers and evaluation servers. These two types of servers are responsible for encrypting and evaluating the garbled circuits, respectively. The proposed protocol achieves three types of obliviousness: input and output obliviousness, function obliviousness and outsourcing obliviousness.

Cloud-assisted SMPC based on homomorphic encryption. Homomorphic encryption can be naturally combined with cloud computing to construct cloud-assisted SMPC protocols [32,36,40]. The basic idea is as follows. To perform a computation task, all participants encrypt their data with an FHE scheme and upload the resulting ciphertexts to the cloud. Then, the cloud performs computations on these ciphertexts and returns the resulting ciphertexts. the date privacy achieved depends on the security of the underlying FHE scheme. However, none of the participants should directly possess the secret key for the FHE scheme. Thus, the problem becomes one of how to decrypt the computation result.

In [1], Asharov et al. solved this problem by sharing the secret key among all the participants. They constructed a cloud-assisted SMPC protocol based on a threshold FHE scheme. In this scenario, during each computation, all of the participants involved are required to generate the parameters of the system, including the secret keys, public keys, and evaluation keys. In practice, however, when participating in such protocols, participants prefer to generate their own long-term parameters ahead of time. Therefore, López-Alt et al. [32] presented a model of on-the-fly multiparty computation. In their model, all participants have their own long-term public/secret key pairs and the SMPC protocols used can be constructed via a multi-key FHE scheme.

The efficiency of the works mentioned above [1,32] depends on the efficiency of the underlying FHE schemes. To improve the protocol efficiency, Peter et al. [40] proposed a new practical protocol based only on additively homomorphic encryption schemes. They used a special type of additively homomorphic encryption referred to as the BCP encryption scheme, in which there is a master secret key that can be used to decrypt ciphertexts encrypted with any public key. In addition, these authors assumed the existence of two non-colluding cloud servers: one storing the master secret key for the underlying encryption scheme and the other storing all user-generated ciphertexts. These two servers perform the necessary computations interactively after all the users involved have uploaded their data.

Comparison. The framework of cloud-assisted SMPC based on FHE is simpler than that based on garbled circuits. However, the practicality of implementations following this approach relies on the efficiency of the underlying FHE schemes, and unfortunately, the question of how to construct practical FHE schemes remains an open problem. Nevertheless, if the goal is

Table 3
Comparison of approaches for cloud-assisted SMPC.

Construction approach	Classification	Scheme	Sever-aided	Security model
Garbled circuit	Outsourcing EVAL	[5,24]	Single	Malicious
	Outsourcing GEN	[4]	Single	Malicious
	Outsourcing Both	[25]	Dual	Semi-honest
Homomorphic encryption	Threshold FHE	[1]	Single	Malicious
	Multi- Key FHE	[32]	Single	Malicious
		[40]	Dual	Semi-honest

not to construct general-purpose protocols, the implementation requires only somewhat homomorphic encryption schemes for certain specific functionalities, and it is still possible to formulate practical protocols.

The construction of cloud-assisted SMPC protocols based on garbled circuits does not require the use of inefficient public key cryptographic tools. However, such protocols still have two disadvantages. First, cloud-assisted SMPC protocols based on garbled circuits can achieve security only under the assumption of non-collusion, while the security achieved based on homomorphic encryption schemes holds even when corrupted parties collude arbitrarily. Second, when executing a protocol based on garbled circuits, at least one user (not the cloud server) must perform a computation whose overhead is linear in the circuit size of the circuit, while for a protocol based on homomorphic encryption, the overhead of any participant other than a cloud server depends only on the length of that participant's input/output, thereby minimizing the participants' local computation and communication costs. A brief comparison of the different approaches is presented in Table 3.

4.3. Cloud computing in specific SMPC situations

Cloud computing has introduced a new paradigm of resource organization and utilization. It is inevitable that secure protocols will need to be designed for and implemented in the cloud settings. As a powerful external resource, cloud not only is used as a supplementary service for SMPC but also serves as an important participant for securely performing specific computational tasks in cryptography, including, encryption schemes, and mathematical calculations [6]. In addition, some new security problems have been introduced or have reemerged in the era of cloud computing, such as searchable encryption, oblivious RAM, verifiable computation [7],and secure deduplication [26]. These problems can also be abstracted as SMPC functionalities and can be addressed via SMPC protocols. It is thus crucial to provide comprehensive practical schemes for SMPC in the cloud computing setting.

5. Application-oriented Secure Multi-Party Computation

In addition to studying advanced techniques for generic SMPC protocols, researchers have also paid attention to more practical uses of SMPC, i.e., application-oriented SMPC. In this section, we focus on some of these practical SMPC applications, including privacy-preserving machine learning, private set intersection and secure genomic sequence comparison.

5.1. Securing machine learning with SMPC

Machine learning is generally regarded as the most important tool for big data systems, because of its ability to efficiently discover valuable knowledge buried in large volumes of data. However, it takes hours or even many days to train image processing algorithms with deep learning techniques, even when this training is accelerated with multiple GPUs. Thus, many cloud service providers offer machine learning services such as Azure Machine Learning and the Google Cloud Machine Learning Engine. Using these platforms, users can outsource their machine learning tasks to the cloud. However, because of the security and privacy concerns, they prefer not to disclose their data to the cloud. Thus, privacy and security have become a hot issue in relation to outsourced learning tasks. In this section, we summarize the recent works on securing machine learning tasks using SMPC with different architectures and different security requirements.

Privacy-preserving machine learning with multiple data sources. In this scenario, data are collected from different users. For example, smart wearable devices may collect data from the users wearing them, and then upload these data to a data centre. As another example of an advanced application, users might upload their positions to the cloud to obtain location-based services from their service provider. Due to privacy concerns, these users wish to prevent their sensitive data from leaking to the cloud. Thus, the challenging question emerges of how to protect each user's sensitive data while also ensuring that machine learning based services can be provided for all users.

Thus far, schemes for private learning from distributed data sources have been proposed that capitalize on two techniques. One is based on randomization, such as the use of differential privacy to add noise to the original data [27]. The other involves construction via SMPC. In the latter case, a general approach is to first design a data aggregation scheme for aggregating data encrypted with different keys into an integrated dataset. Besides, there are also research works for specific privacy-preserving machine learning tasks with multiple data sources. For example, in [16], a privacy-preserving naïve bayes classification method was constructed.

Protecting the data and the trained model in the two-party case. In this scenario, a cloud server (called Bob) holds a dataset and trains a model with a specific machine learning algorithm. For profit reasons, Bob wishes not to leak this model to other entities. The user (called Alice) hopes to receive personalized service based on her own data but does not want her private data to leak. This situation is suitable for the application of the classic SMPC protocol: Alice inputs her private data, and Bob inputs the model. At the end of the protocol execution, Bob has learned nothing about Alice's data, and Alice has learned as little as possible about Bob's model. Approaches of this type include those presented in [3,11].

5.2. Private set operations

Set operations are basic operations commonly used in SMPC. They are basic tools for tasks such as privacy-preserving data queries and data mining. Freedman et al. [15] first studied two-party set intersection and proposed a protocol based on Paillier's homomorphic cryptosystem. The proposed protocol achieved security against semi-honest adversaries in the standard model, and resisted malicious parties in the random oracle (RO) model. Freedman et al.'s work introduced the research direction of private set operations, and the tools they used have become the basis for subsequent research works.

One research direction involves expanding the scope of set operations, while another research direction is to design a private set intersection (PSI) protocol under a stronger security model or a simpler cryptographic assumption. To design a PSI protocol resistant to malicious adversaries without introducing inefficient zero-knowledge proofs, Hazay and Lindell [19] proposed to use oblivious pseudorandom function evaluations (OPRF) to solve the set intersection and pattern matching problems. For the full malicious security model, there has been different approaches for PSI protocols based on other paradigms [18,44].

There are mainly five different techniques to construct efficient PSI protocols, including A Naïve Solution, Third Party-Based PSI, Public-Key-Based PSI, Circuit-Based PSI and OT-Based PSI. Recently, an increasing number of research works on OT-Based PSI have appeared. In this paragraph, we mainly focus on PSI protocols based on OT extension. Dong et al. [12] proposed a PSI protocol based on OT extension and Bloom filter. Pinkas et al. [41] optimized Dong et al.'s work using *Random Garbled Bloom Filter* protocol, which is based on the random OT extension and achieves security in the semi-honest model. A weak notion of security is formalized in [42], in which Rindal and Rosulek described an inexpensive protocol paradigm for PSI and optimized [41] to achieve weakly malicious security. Later, Rindal and Rosulek [43] tried to extend PSI protocols to be secure in the presence of malicious adversaries, and showed that Bloom filter-based protocol is not secure against malicious adversaries.

5.3. Secure genomic sequence comparison

Genome sequence comparison is one of the most basic and widely used operations in genomic computation. For secure genomic computation, the main technique adopted is S2PC. Privacy-preserving genome sequence comparison is a specific scenario of S2PC that focuses on achieving the alignment of two genome sequences in a secure manner. Generic construction methods for S2PC protocols can be classified into two main categories based on the underlying techniques used: homomorphic-encryption-based construction and garbled-circuit-based construction. Generic construction can enable the secure computation of any functionality and has many advantages: namely, it is more intuitive, more secure, and more efficient in most cases. Due to these advantages, studies on secure genomic sequence comparison methods in the academic community are typically based on generic construction techniques.

Construction based on garbled circuits. Based on garbled circuit technology, Jha et al. [23] proposed three secure sequence comparison protocols under the semi-honest model. In the first protocol, the circuit is directly constructed based on Yao's garbled circuit method. In the second protocol, the circuits are divided, and the results of each circuit are shared among the participants to achieve better performance. The last protocol combines the first two protocols and is optimized in terms of protocol efficiency and scalability. The main drawback of this solution lies in its inefficiency when addressing large-scale computation tasks. Huang et al. [20] further optimized the scheme presented in [23], to reduce the computational cost to the participants, although the cost of interaction between the participants remained high. Wang et al. [47] officially defined the edit distance problem as a similar patient query problem, and introduced a common reference sequence to be used in preprocessing a genomic data set. By calculating the approximate edit distance, the efficiency of sequence comparison on large data sets is greatly improved. However, this approach also introduces some problems. First, the introduction of a common reference sequence leads to the leakage of some information about the data distribution. Second, performing calculations based on a reference sequence results in a decrease in accuracy. Recently, Zhu and Huang [49] studied more generalized edit distance algorithms, including weighted edit distance, longest common subsequence, and heaviest common subsequence algorithms.

Construction based on homomorphic encryption. In addition to the above methods, researchers also combined homomorphic encryption with edit distance to construct secure genomic sequence comparison protocols. Edit distance refers to the minimum number of operations needed to edit one string into another. This metric has a wide range of applications in bioinformatics, search engines, intrusion detection, and speech recognition. It quantitatively describes the similarity between two

strings. It is a common way to securely compute the edit distance between two genomic sequences using dynamic programming. However, due to the high computational complexity of the dynamic programming algorithm, protocols based on this technique are not efficient at all. One effective way to improve the efficiency is to use outsourced computation to reduce participants' local computational cost. Ma et al. [33] studied ways of performing secure outsourcing of sequence comparison algorithms, introduced multiple servers to share the participants' computational tasks, and constructed a secure outsourcing protocol that can resist malicious attacks.

6. Conclusions

After more than three decades of SMPC research and development, the fundamental theories of SMPC have become relatively mature. Researchers have designed a number of theoretical models. However, in recent years, as Internet technology and computing capabilities have advanced, researchers have become more concerned on the practical applications of SMPC. Not only has the efficiency of general-purpose SMPC protocols been greatly improved, but SMPC models for various application scenarios have also been proposed and studied. This survey has provided a systematic overview of the existing solutions for SMPC, including generic construction methods and cloud-assisted secure computation, as well as protocols for specific applications such as privacy-preserving machine learning, private set operations, and secure genomic computation. Although extensive research has already solved many of the challenges in SMPC, a number of interesting problems remain to be fully explored. As one of the core directions in the field of cryptography, SMPC has been demonstrated to have a powerful role in all aspects. The rich theory it contains is worth untiring efforts and in-depth study.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 61702218, 61672262), Shandong Province Higher Educational Science and Technology Program (No. J18KA349), Natural Science Foundation of Shandong Province (No. ZR2014JL042, ZR2014FL011, ZR2015FL023), Shandong Provincial Key R&D Program (No. 2016GGX101001), CERNET Next Generation Internet Technology Innovation Project (No. NGII20160404), Science and Technology Program of University of Jinan (No. XKY1709), and Doctoral Program of University of Jinan (No.160100224).

References

[1] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, D. Wichs, Multiparty computation with low communication, computation and interaction via threshold fhe, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 483–501.
[2] R. Bendlin, I. Damgård, C. Orlandi, S. Zakarias, Semi-homomorphic encryption and multiparty computation, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2011, pp. 169–188.
[3] R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, in: Proceedings of the NDSS, 4324, 2015, p. 4325.
[4] H. Carter, C. Lever, P. Traynor, Whitewash: outsourcing garbled circuit generation for mobile devices, in: Proceedings of the 30th Annual Computer Security Applications Conference, ACM, 2014, pp. 266–275.
[5] H. Carter, B. Mood, P. Traynor, K. Butler, Secure outsourced garbled circuit evaluation for mobile devices, in: Proceedings of the 22nd USENIX conference on Security, USENIX Association, 2013, pp. 289–304.
[6] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, D.S. Wong, New algorithms for secure outsourcing of large-scale systems of linear equations, IEEE Trans. Inf. Forensics Secur. 10 (1) (2015) 69–78.
[7] X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable computation over large database with incremental updates, IEEE Trans. Comput. 65 (10) (2016) 3184–3195.
[8] I. Damgård, J.B. Nielsen, M. Nielsen, S. Ranellucci, The tinytable protocol for 2-party secure computation, or: gate-scrambling revisited, in: Proceedings of the Annual International Cryptology Conference, Springer, 2017, pp. 167–187.
[9] I. Damgård, V. Pastro, N. Smart, S. Zakarias, Multiparty computation from somewhat homomorphic encryption, in: Proceedings of the Advances in Cryptology–CRYPTO, Springer, 2012, pp. 643–662.
[10] I. Damgård, S. Zakarias, Constant-overhead secure computation of boolean circuits using preprocessing, in: Proceedings of the Theory of Cryptography, Springer, 2013, pp. 621–641.
[11] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. Nascimento, W.-S. Poon, S. Truex, Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation, IEEE Trans. Dependable Secure Comput. (2017), doi:10.1109/TDSC.2017.2679189.
[12] C. Dong, L. Chen, Z. Wen, When private set intersection meets big data: an efficient and scalable protocol, in: Proceedings of the ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 789–800.
[13] T.K. Frederiksen, T.P. Jakobsen, J.B. Nielsen, P.S. Nordholt, C. Orlandi, Minilego: efficient secure two-party computation from general assumptions, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2013, pp. 537–556.
[14] T.K. Frederiksen, T.P. Jakobsen, J.B. Nielsen, R. Trifiletti, Tinylego: an interactive garbling scheme for maliciously secure two-party computation, IACR Cryptol. ePrint Arch. 2015 (2015) 309.
[15] M.J. Freedman, K. Nissim, B. Pinkas, Efficient private matching and set intersection, in: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, pp. 1–19.
[16] C.-Z. Gao, Q. Cheng, P. He, W. Susilo, J. Li, Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack, Inf. Sci. 444 (2018) 72–88.
[17] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game, in: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, ACM, 1987, pp. 218–229.
[18] C. Hazay, Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs, J. Cryptol. 31 (2) (2018) 537–586.
[19] C. Hazay, Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, in: Proceedings of the Theory of Cryptography Conference, Springer, 2008, pp. 155–175.
[20] Y. Huang, D. Evans, J. Katz, L. Malka, Faster secure two-party computation using garbled circuits., in: Proceedings of the USENIX Security Symposium, 201, 2011, pp. 331–335.
[21] Y. Ishai, M. Prabhakaran, A. Sahai, Founding cryptography on oblivious transfer–efficiently, in: Proceedings of the Annual International Cryptology Conference, Springer, 2008, pp. 572–591.

[22] Y. Ishai, M. Prabhakaran, A. Sahai, Secure arithmetic computation with no honest majority, in: Proceedings of the Theory of Cryptography Conference, Springer, 2009, pp. 294–314.
[23] S. Jha, L. Kruger, V. Shmatikov, Towards practical privacy for genomic computation, in: Proceedings of the IEEE Symposium on Security and Privacy (sp 2008), IEEE, 2008, pp. 216–230.
[24] S. Kamara, P. Mohassel, B. Riva, Salus: a system for server-aided secure function evaluation, in: Proceedings of the ACM Conference on Computer and Communications Security, ACM, 2012, pp. 797–808.
[25] F. Kerschbaum, Oblivious outsourcing of garbled circuit generation, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, 2015, pp. 2134–2140.
[26] J. Li, X. Chen, X. Huang, S. Tang, Y. Xiang, M.M. Hassan, A. Alelaiwi, Secure distributed deduplication systems with improved reliability, IEEE Trans. Comput. 64 (12) (2015) 3569–3579.
[27] T. Li, J. Li, Z. Liu, P. Li, C. Jia, Differentially private naive bayes learning over multiple data sources, Inf. Sci. 444 (2018) 89–104.
[28] Y. Lindell, Fast cut-and-choose based protocols for malicious and covert adversaries, in: Proceedings of the Advances in Cryptology-CRYPTO, Springer, 2013, pp. 1–17.
[29] Y. Lindell, E. Oxman, B. Pinkas, The ips compiler: Optimizations, variants and concrete efficiency, in: Proceedings of the Annual Cryptology Conference, Springer, 2011, pp. 259–276.
[30] Y. Lindell, B. Pinkas, An efficient protocol for secure two-party computation in the presence of malicious adversaries, in: Proceedings of the Advances in Cryptology-EUROCRYPT, Springer, 2007, pp. 52–78.
[31] Y. Lindell, B. Pinkas, Secure two-party computation via cut-and-choose oblivious transfer, in: Proceedings of the Theory of Cryptography, Springer, 2011, pp. 329–346.
[32] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: Proceedings of the 44th Annual ACM Symposium on Theory of Computing, ACM, 2012, pp. 1219–1234.
[33] X. Ma, J. Li, F. Zhang, Refereed computation delegation of private sequence comparison in cloud computing., IJ Netw. Secur. 17 (6) (2015) 743–753.
[34] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella, et al., Fairplay-secure two-party computation system., San Diego, CA, USA, in: Proceedings of the USENIX Security Symposium, 4, 2004, p. 9.
[35] B. Mood, D. Gupta, K. Butler, J. Feigenbaum, Reuse it or lose it: more efficient secure computation through reuse of encrypted values, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 582–596.
[36] P. Mukherjee, D. Wichs, Two round multiparty computation via multi-key FHE, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2016, pp. 735–763.
[37] J.B. Nielsen, P.S. Nordholt, C. Orlandi, S.S. Burra, A new approach to practical active-secure two-party computation, in: Proceedings of the Advances in Cryptology-CRYPTO, Springer, 2012, pp. 681–700.
[38] J.B. Nielsen, C. Orlandi, Lego for two-party secure computation, in: Proceedings of the Theory of Cryptography Conference, Springer, 2009, pp. 368–386.
[39] J.B. Nielsen, T. Schneider, R. Trifiletti, Constant round maliciously secure 2PC with function-independent preprocessing using LEGO, IACR Cryptol. ePrint Arch. 2016 (2016) 1069.
[40] A. Peter, E. Tews, S. Katzenbeisser, Efficiently outsourcing multiparty computation under multiple keys, IEEE Trans. Inf. Forensics Secur. 8 (12) (2013) 2046–2058.
[41] B. Pinkas, T. Schneider, M. Zohner, Faster private set intersection based on OT extension, in: Proceedings of the USENIX Security Symposium, 14, 2014, pp. 797–812.
[42] P. Rindal, M. Rosulek, Faster malicious 2-party secure computation with online/offline dual execution., in: Proceedings of the USENIX Security Symposium, 2016, pp. 297–314.
[43] P. Rindal, M. Rosulek, Improved private set intersection against malicious adversaries, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2017, pp. 235–259.
[44] P. Rindal, M. Rosulek, Malicious-secure private set intersection via dual execution, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 1229–1242.
[45] C.-h. Shen, et al., Two-output secure computation with malicious adversaries, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2011, pp. 386–405.
[46] X. Wang, S. Ranellucci, J. Katz, Authenticated garbling and efficient maliciously secure two-party computation, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 21–37.
[47] X.S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, D. Bu, Efficient genome-wide, privacy-preserving similar patient query based on private edit distance, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 492–503.
[48] A.C. Yao, How to generate and exchange secrets, in: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, IEEE, 1986, pp. 162–167.
[49] R. Zhu, Y. Huang, Efficient privacy-preserving general edit distance and beyond, Technical Report, 2017. Cryptology ePrint Archive
[50] R. Zhu, Y. Huang, Jimu: faster lego-based secure computation using additive homomorphic hashes, in: Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2017, pp. 529–572.