Cambridge University Press & Assessment
978-1-009-29951-0 — Privacy-preserving Computing
Kai Chen , Qiang Yang
Excerpt
More Information

1

Introduction to Privacy-preserving Computing

Massive data and powerful computing resources have become the primary driving force in the development of big data and artificial intelligence. On one hand, mobile phones, social websites, and various sensors collect people's everyday activities continuously. On the other hand, high-performance computing facilities and efficient machine learning algorithms enable the training of complex models. However, improper usage of sensitive information in machine learning and data analysis may lead to catastrophes. For example, the leakage of personal information may expose individuals to fraud crimes. As a result, developing privacy-preserving theories and systems has become extremely necessary. In this chapter, we introduce the fundamental definitions and theories of privacy-preserving computing to help readers understand the basic concepts, technologies, and solutions of privacy-preserving computing.

1.1 Definition and Background

Nowadays, with the pervasive application of computers, a large amount of data is collected and processed by computers, which poses the following challenges to privacy protection:

- **Increased cost for privacy protection.** Massive, sensitive data such as names, ID numbers, and property information is stored in various forms of computer devices and accessed, updated, and transferred frequently. Its sheer scale and complex and volatile application scenarios greatly increase privacy protection costs compared with those of gathering only a small amount of information by statistical agencies in the early days.
- **Increased difficulty in privacy protection.** On one hand, private data can be stored in various locations such as personal mobile devices and data centers. Therefore, privacy protection schemes need to deal with privacy

protection issues under different devices or hosting modes. On the other hand, the risk of computer intrusion and sensitive data theft cannot be eliminated due to the prevalence of computing devices and the sophistication of attacking techniques. The diversity and complexity of modern devices increase the difficulty of privacy protection compared with processing with pen and paper manually in the early days.

- **Increased damage caused by privacy leakage.** The pervasion and abuse of sensitive data greatly increase the damage caused by privacy leakage. For example, disclosed ID numbers can be used to commit crimes.

Confronted with these challenges, we should not only rely on legal systems but also integrate mathematical theories and algorithms in privacy protection. Multiple techniques such as cryptography tools need to be used to prevent privacy leakage in the workload of big data analysis and machine learning and they are at the core of privacy-preserving computation.

1.1.1 Definition of Privacy-preserving Computing

Privacy-preserving computing refers to a series of techniques for computing without breaching raw data, which guarantee the available but invisible property of data during their usage. In this book, we focus on a series of techniques for protecting data privacy and enabling computing tasks at the same time, including secret sharing, homomorphic encryption, oblivious transfer, garbled circuit, differential privacy, and federated learning. Privacy-preserving computing incorporates multiple disciplines including cryptography, statistics, computer architecture, and artificial intelligence. The development of its theory and applications is inseparable from cloud computing, big data, and artificial intelligence. At present, privacy-preserving computing is mainly used in data query and analysis as well as machine learning, which have the following characteristics:

- **Data query and analysis**: This type of application usually consists of simple computing tasks such as searching, summing, averaging, and variance calculation, in which the definition and protection of individuals' data privacy is the most important topic.
- **Machine learning**: Machine learning involves the collection of training data and adopts optimization methods to learn models that can extract features and patterns from training data. The model is then used for tasks such as prediction, classification, or behavior guidance. The training and inference usually involve complex computations such as the sigmoid function. During model training, privacy-preserving computing needs to protect the privacy

of the training dataset. During model inference, i.e., predicting new data, privacy-preserving computing needs to protect the privacy of the incoming data.

1.1.2 Taxonomy of Privacy-preserving Computing

The definition of privacy protection varies with the requirement of computing tasks. For machine learning, privacy protection concentrates on both the training and the inference processes. In the training process, the training data and gradients need privacy protection, because training data usually contains sensitive information and gradients are generated from the private data with the training algorithm. During inference, in addition to the privacy of input data, the model parameters also need protection, because the model parameters are trained by private training data and may be exploited to get sensitive information. For databases, the result of queries may contain sensitive information of the data and need protection from random noise. Also, the column name in the queries may need protection to protect the database users' privacy.

Privacy-preserving computing can be classified into encrypted computation with cryptography-based security protocols at its core and privacy-preserving computation with a broader definition. Table 1.1 presents names and definitions of some concepts frequently used in privacy-preserving computing.

Encrypted computation uses cryptography tools to construct privacy-preserving computing applications so that multiple data owners can collaborate on computing tasks while protecting secret data. Secure Multi-Party Computation (MPC) (Goldreich, 1998) is one of the representatives of such tools. Cryptography tools encrypt data as ciphertext, which is indistinguishable from random numbers during communication. As a result, the plaintext cannot be accessed by participants except for the private key owners. Encrypted computation is formally proven to guarantee the cryptography-level security of data privacy. However, it is inefficient in practice due to the high computational or communication complexity of the cryptography tools. Recent works have focused on optimizing the performance of cryptographic tools in various applications such as machine learning (Hardy et al., 2017; Mohassel and Zhang, 2017) and data mining (Boneh et al., 2013; Chen et al., 2020; Evfimievski et al., 2003). With the development of cryptography tools such as secret sharing (De Santis et al., 1994), oblivious transfer (Rabin, 2005), garbled circuit (Yao, 1986), and homomorphic encryption (Gentry, 2009), encrypted computation still has a wide range of applications.

Cambridge University Press & Assessment
978-1-009-29951-0 — Privacy-preserving Computing
Kai Chen , Qiang Yang
Excerpt
More Information

Table 1.1 *Names and definitions in privacy-preserving computing.*

Name	Definition
Privacy Computing	Encrypted computation based on cryptography tools in a narrow sense. In a broad sense, it refers to all techniques used for protecting data privacy while achieving computational goals.
Privacy-Preserving Computation	Secures data acquisition and management before computing, data privacy protection in computing, and data privacy protection and interest allocation after computing. Some privacy-preserving computing techniques use encryption and can be regarded as part of privacy computing.
Secure Multi-Party Computing	Uses cryptography tools to construct privacy computing protocols at the security protocol layer so that multiple data owners can collaborate on computing a specific function without disclosing any other private information.
Secret Sharing	A tool dividing private data into multiple partitions for distribution and computation.
Homomorphic Encryption	A cryptographic scheme allowing cipher computing. After encrypting raw messages, only private key owners can decrypt the ciphertext to obtain results. It is a common tool in encrypted computation.
Oblivious Transfer	A commonly used data transmission model considering privacy protection. It ensures that, at the time of transferring private data from a sender to a receiver, the sender does not know the choice of the receiver and the receiver does not know other private data transferred by the sender either.
Garbled Circuit	Protects plaintext electronic signals by encrypting the input and output signals of logic gates. It is one of the most widely used privacy encryption techniques.
Differential Privacy	A flexible and effective privacy protection technique. It differs from cryptography-based schemes in that it does not encrypt data but protects privacy by adding random noise.
Trusted Execution Environment	A scheme providing privacy computing at a hardware level. It gives users a running environment that separates programs from data so that they cannot be stolen or tampered with by potential intruders.
Federated Learning	A privacy-preserving scheme proposed in the field of machine learning. The information required for model training is transferred between participants while the raw data is not. Federated learning can utilize privacy protection techniques such as homomorphic encryption to provide strict protection to the transferred information.

In encrypted computation, given input data from each participant $X_1, X_2, \ldots$ and computing task $Y = f(X_1, X_2, \ldots)$, all participants collaborate on private computing with various cryptography tools. Encrypted computation guarantees that no information other than the final output Y is disclosed in computing, ensuring that each participant's private data is kept secret. Note that cryptographic tools do not offer privacy protection in multiple independent tasks. Taking the millionaire problem as an example, suppose two millionaires are comparing their properties using encrypted computation so that they do not know how much the other's property is worth. Nonetheless, the intermediate results produced in multiple rounds of comparison can be combined to deduce each millionaire's property with high precision. Furthermore, in machine learning, when multiple data owners collaborate on the training of machine learning models, although the whole training process is encrypted by cryptography tools, a participant can still gather the gradients of the training data transmitted by other participants by analyzing the parameter updating process of its local model. Subsequently, some private information may be inferred from these gradients.

Privacy-preserving computation. Some works adopt noncryptographic tools or compromise on encryption for better performance. In this book, such new privacy-preserving computing techniques are called privacy-preserving computation to distinguish them from the traditional encrypted computation techniques based purely on cryptography tools.

Each participant's privacy leakage after a computing task is finished is closely related to the task's property. The subject of privacy-preserving computation is to study the possibility and degree of each participant's privacy leakage in the entire computing process and to measure and protect data privacy from the perspective of the whole task, which is also a focus of the research in this field.

Privacy-preserving computing techniques can be divided into multiple categories. These include Secure Multi-Party Computation (MPC) (Goldreich, 1998), Homomorphic Encryption (HE) (Gentry, 2009), Differential Privacy (DP) (Dwork, 2008), Trusted Execution Environment (TEE) (Pinto and Santos, 2019), and Federated Learning (FL) (Yang et al., 2019c), based on the development tracks, algorithm basis, and application characteristics of the techniques.

This book provides a comprehensive and in-depth overview and analysis of the aforementioned technologies for encrypted and privacy-preserving computation, allowing readers to gain a complete understanding of privacy-preserving computing. The book focuses on introducing the principles and implementations of technical approaches for privacy protection. Untechnical methods such as supervising companies or organizations by legislation and institutions are not covered in the book.

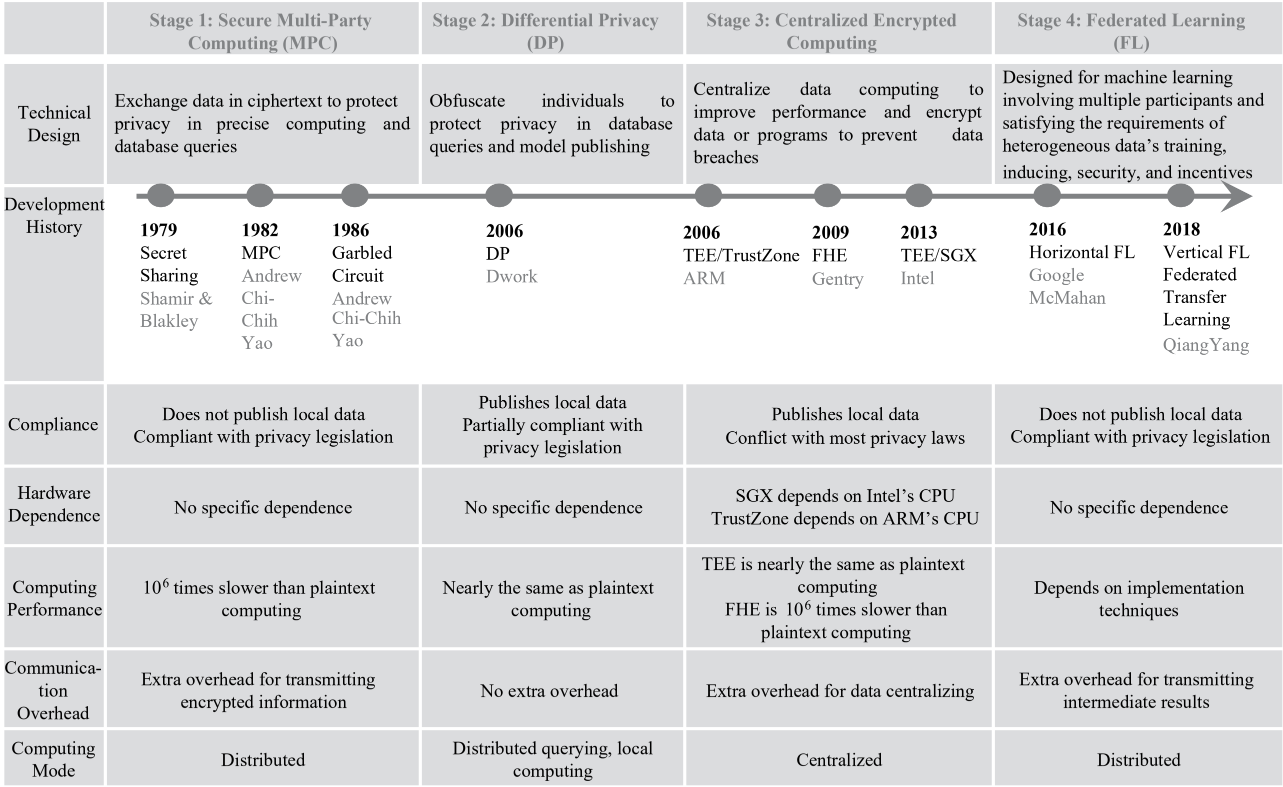
1.1.3 History of Privacy-preserving Computing

The development of privacy-preserving computing dates back to the time before the invention of computers. Some statistical agencies often used questionnaires to study social phenomena. To protect respondents' privacy on data such as names, ages, and replies, they usually promised that the collected information would only be used for research purposes and would be strictly supervised. At that time, the protection of the respondents' privacy relied mainly on institutions and public regulations. Additionally, in the early days, cryptography played an important role in certain key fields such as military intelligence (Singh, 1999). These small-scale cases of using simple cryptography tools can be seen as early examples of privacy-preserving computing.

The development of modern privacy-preserving computing can be divided into four stages. In each stage, new computing schemes are proposed from different perspectives to solve the omissions and defections in the previous stage. These schemes provide many different views and ideas to solve privacy-preserving computing problems so as to enrich the selections of techniques in different application scenarios. Figure 1.1 presents several critical moments and the corresponding functional features.

The first stage is the development of theories and applications of Secure Multi-Party Computation (MPC) (Goldreich, 1998). Multi-Party Computation is a type of encrypted computation that uses cryptography tools to construct a secure computational model under which multiple participants can collaborate on a computing task using their own data without the fear of leaking data to others. The proposal of Shamir Secret Sharing (Shamir, 1979) heralded the birth of MPC. Following that, a system with secret sharing (Yao, 1982) and garbled circuit (Yao, 1986) as its fundamental protocols was built to implement privacy protection through generating and exchanging random numbers and ensure the validity of computational results via predefined computing protocols. The idea of MPC is well-suited for precise computing and database queries, the security of which is provable. Some privacy-preserving legislation such as GDPR (Europe, 2019) requires that the data shall be kept locally, and MPC satisfies the requirement of not publishing local data. However, its main disadvantage is the enormous performance disparity with unencrypted computing. In the worst case, MPC runs 10^6 times slower than unencrypted

Cambridge University Press & Assessment
978-1-009-29951-0 — Privacy-preserving Computing
Kai Chen , Qiang Yang
Excerpt
More Information



	Stage 1: Secure Multi-Party Computing (MPC)	Stage 2: Differential Privacy (DP)	Stage 3: Centralized Encrypted Computing	Stage 4: Federated Learning (FL)
Technical Design	Exchange data in ciphertext to protect privacy in precise computing and database queries	Obfuscate individuals to protect privacy in database queries and model publishing	Centralize data computing to improve performance and encrypt data or programs to prevent data breaches	Designed for machine learning involving multiple participants and satisfying the requirements of heterogeneous data's training, inducing, security, and incentives
Development History	1979 Secret Sharing Shamir & Blakley 1982 MPC Andrew Chi-Chih Yao 1986 Garbled Circuit Andrew Chi-Chih Yao	2006 DP Dwork	2006 TEE/TrustZone ARM 2009 FHE Gentry 2013 TEE/SGX Intel	2016 Horizontal FL Google McMahan 2018 Vertical FL Federated Transfer Learning QiangYang
Compliance	Does not publish local data Compliant with privacy legislation	Publishes local data Partially compliant with privacy legislation	Publishes local data Conflict with most privacy laws	Does not publish local data Compliant with privacy legislation
Hardware Dependence	No specific dependence	No specific dependence	SGX depends on Intel's CPU TrustZone depends on ARM's CPU	No specific dependence
Computing Performance	10^6 times slower than plaintext computing	Nearly the same as plaintext computing	TEE is nearly the same as plaintext computing FHE is 10^6 times slower than plaintext computing	Depends on implementation techniques
Communication Overhead	Extra overhead for transmitting encrypted information	No extra overhead	Extra overhead for data centralizing	Extra overhead for transmitting intermediate results
Computing Mode	Distributed	Distributed querying, local computing	Centralized	Distributed

Figure 1.1 History of privacy-preserving computing (along the time dimension).

computing. The exact cost varies with different computing tasks and network environments. The bottleneck is the communication overhead.

The second stage is the theories and applications of differential privacy (DP). Differential privacy has long been applied to customer surveys. It differs from MPC in that it perturbs user data by adding random noise to the process or results of a computing task. Differential privacy is based on the obfuscation of data distribution (Dwork, 2008) and evaluates its privacy protection capability at a more flexible level than the cryptography-based techniques relying on the difficulty of solving NP-hard problems. From a legal compliance standpoint, DP satisfies certain privacy protection laws. Meanwhile, it runs at nearly the same speed as computing in plaintext, which is significantly faster than cryptography-based schemes because it does not encrypt data or require extra intensive communications. As a result, DP is initially applied to various artificial intelligence applications, such as Google Keyboard (Google, 2020) and Apple Siri (Apple, Differential Privacy Team, 2017; Ding et al., 2017), to protect end users' privacy.

The third stage is centralized encrypted computation such as trusted execution environment (TEE) and fully homomorphic encryption (FHE). Unlike the previous stages, this stage aims to find a manner to publish data securely. One technical route is TEE, which builds an isolated running environment where users can upload their data without worrying about it being stolen by other programs or computer devices. The implementation of TEE depends on specific hardware produced by different manufacturers, such as Intel SGX (Costan and Devadas, 2016) and ARM TruestZone (Pinto and Santos, 2019). Therefore, its security is not 100 percent guaranteed. However, it runs nearly as fast as plaintext computing,[1] making it more practical than homomorphic encryption. As for FHE, the earliest development of HE can be traced back to the privacy-preserving computing scheme proposed by Rivest. The first version of FHE was not published until 2009 (Gentry, 2009). Homomorphic encryption enables us to perform effective computations directly on ciphertext without decrypting it. However, the effectiveness needs to be paid for by increasing the communication time for ciphertext transmission. Additionally, homomorphic encryption consumes a significant amount of computational time, typically six orders of magnitude more than plaintext computing. Both TEE and FHE provide mechanisms for secure data publishing. However, they may conflict with privacy protection laws that prohibit publishing local data (Europe, 2019).

[1] Performing computation without encryption, in contrast to computing on ciphertext with homomorphic encryption.

The fourth stage is federated learning designed for machine learning model training and inference tasks. In contrast to conventional technical roadmaps, federated learning allows data owners to store their data locally and exchange only the protected parameters to accomplish a training process. Therefore, private local data is not exposed in the framework of federated learning. Moreover, the exchange of model parameters does not expose raw data and model contents. Additionally, federated learning protects the privacy of model inference. It focuses on the design of data heterogeneity and security mechanism for distributed machine learning and its performance optimization to avoid performance issues like those encountered in MPC. Federated learning can be implemented based on many privacy-preserving computing techniques such as MPC, DP, TEE, and FHE. Meanwhile, it needs to consider the characteristics of specific modeling and prediction tasks when developing methods for parameter protection. The security of federated learning is usually analyzed jointly with specific machine learning tasks. Compared to the third stage, federated learning guarantees that the local data is kept locally and will not be shared in any form. When regarded as a training paradigm, federated learning can be divided into horizontal and vertical federated learning (Yang et al., 2019c). Compared with the traditional centralized machine learning paradigm, the cost of federated learning comes primarily from cipher computation and the additional communications for transferring intermediate results.

Today, with the deepening of communication and cooperation between multiple research fields, a growing number of startups and large companies enter the privacy-preserving computing industry and release products such as WeBank's FATE and OpenMinded's Syft. Privacy-preserving computing is applied in a wide variety of fields including database query (Microsoft, 2016), vote counting (Xia et al., 2008), and machine learning (Google, 2020).

1.2 Main Technologies of Privacy-preserving Computing

This book introduces the mainstream privacy-preserving computing technologies in the following chapters. Chapters 2–5 introduce common cryptography tools in privacy-preserving computing, including secret sharing, homomorphic encryption, oblivious transfer, and garbled circuit. Differential privacy, trusted execution environment, and federated learning, which are the core techniques of privacy-preserving computation, are discussed in Chapters 6–8. Figure 1.2 illustrates the relationships between the techniques. Federated learning can realize more flexible privacy protection by integrating more privacy protection techniques, such as DP and TEE, than those based on traditional encrypted computation.

Cambridge University Press & Assessment
978-1-009-29951-0 — Privacy-preserving Computing
Kai Chen , Qiang Yang
Excerpt
More Information

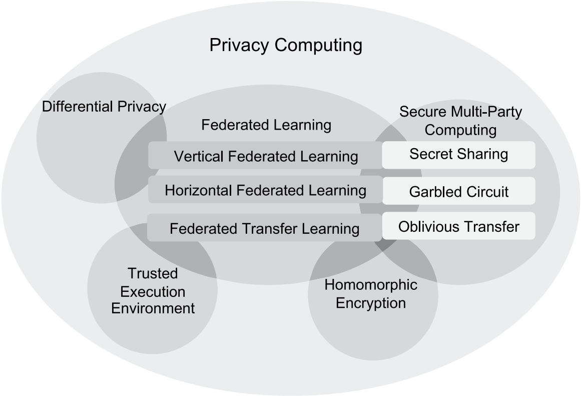

Privacy Computing
Differential Privacy
Federated Learning
Vertical Federated Learning
Horizontal Federated Learning
Federated Transfer Learning
Secure Multi-Party Computing
Secret Sharing
Garbled Circuit
Oblivious Transfer
Trusted Execution Environment
Homomorphic Encryption

Figure 1.2 Technical framework of privacy-preserving computing.

Chapter 2: Secret Sharing. Secret sharing is an important implementation of encrypted computation, which allows a participant to split private data randomly into multiple parts and send them to other participants. The raw data can be revealed only when a certain number of participants reach an agreement.

Chapter 3: Homomorphic Encryption. Homomorphic Encryption (HE) is an encryption method that supports computing on the ciphertext. Unlike non-homomorphic encryption methods, it allows users to encrypt private data and send ciphertext to an untrusted computer and perform computation on the ciphertext. In the view of an adversary, the input data of programs using HE contains no private information. Privacy will never be disclosed provided that the private keys of HE are not leaked.

Chapter 4: Oblivious Transfer. Oblivious Transfer (OT) defines a privacy-preserving data transmission model that divides participants into a sender and a receiver. The sender possesses private data and the receiver selects messages from the received data. They agree on sending a piece of the sender’s private data to the receiver on the assumption that the sender is unaware of the receiver’s choice and the receiver does not know the sender’s other private data either. Oblivious transfer has been widely applied to solving simple privacy-preserving computing problems and developing complex privacy-preserving computing protocols.

Chapter 5: Garbled Circuit. Garbled Circuit (GC) is one of the most adaptable privacy-preserving computing protocols, since it applies to all computing tasks that circuits can express. GC protects plaintext signals in circuits by encrypting the input and output signals of logic gates. A GC protocol classifies participants into a generator and an operator, who are responsible for garbling