Practical Secure Computation Outsourcing: A Survey

ZIHAO SHAN, KUI REN, and MARINA BLANTON, University at Buffalo, USA
CONG WANG, City University of Hong Kong, China

The rapid development of cloud computing promotes a wide deployment of data and computation outsourcing to Cloud Service Providers (CSPs) by resource-limited entities. Based on a pay-per-use model, a client without enough computational power can easily outsource large-scale computational tasks to a cloud. Nonetheless, the issue of security and privacy becomes a major concern when the customer's sensitive or confidential data is not processed in a fully trusted cloud environment. Recently, a number of publications have been proposed to investigate and design specific secure outsourcing schemes for different computational tasks. The aim of this survey is to systemize and present the cutting-edge technologies in this area. It starts by presenting security threats and requirements, followed with other factors that should be considered when constructing secure computation outsourcing schemes. In an organized way, we then dwell on the existing secure outsourcing solutions to different computational tasks such as matrix computations, mathematical optimization, etc., treating data confidentiality as well as computation integrity. Finally, we provide a discussion of the literature and a list of open challenges in the area.

CCS Concepts: • **Information systems → Data encryption**; • **Security and privacy → Privacy protections**; *Privacy-preserving protocols*; *Management and querying of encrypted data*; • **Computer systems organization → Cloud computing**;

Additional Key Words and Phrases: Cloud Computation; Large-scale Dataset; Data security; Privacy Preservation; Computation Integrity

ACM Reference Format:
Zihao Shan, Kui Ren, Marina Blanton, and Cong Wang. 2017. Practical Secure Computation Outsourcing: A Survey. *ACM Comput. Surv.* 0, 0, Article 0 (2017), 40 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The exponential growth in the quantity of data generated nowadays has been drawing increasing attention. It is estimated that the amount of usable data created will be over 15 zettabytes by 2020, compared to 0.9 zettabytes in 2013 [1]. This has led to an unavoidable challenge, as data owners and analysts have to figure out a way to properly store and effectively analyze the large-scale data.

The storage capacity and computational capability of processing large-scale data is still limited by the hardware and available memory of computer systems, such as cell-phones, modern portable laptops, security access cards, and sensors for the majority of clients or companies. Either they are unable to run a large-scale data-related task or the amount of time needed for computation is far from reasonable. A potential solution to this is to seek help from a sophisticated computing infrastructure. The evolution of the supercomputers steadily provides high computational capacity and facilities for a wide range of computation-intensive tasks. For example, the U.S. government has

Authors' addresses: Zihao Shan, zihaosha@buffalo.edu; Kui Ren, kuiren@buffalo.edu; Marina Blanton, mblanton@buffalo.edu, University at Buffalo, Davis Hall, White Road, Amherst, NY, 14260, USA; Cong Wang, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong, China, congwang@cityu.edu.hk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2017 Association for Computing Machinery.
0360-0300/2017/0-ART0 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

been investing in supercomputers, pursuing the goals of the National Strategic Computing Initiative and building the overall capacity and capability of an enduring national HPC ecosystem [124]. Many universities host supercomputers on campus to meet the demand of research, such as Blue Waters at the University of Illinois at Urbana-Champaign, which is claimed to help predict the behavior of complex biological systems for simulating the evolution of the cosmos [22]. Nonetheless, normal clients are not able to equip themselves with expensive supercomputers due to the high cost of purchasing and maintaining them. Moreover, the possibility for a typical company to get access to a supercomputer with sufficient computing power is small based on the government or university policies.

The above phenomenon contributes to the motivation of developing solutions for outsourcing computation, also known as cloud computation in a large-scale setting. Clients like individuals or companies can easily upload their data to a powerful cloud server which can then expedite the time of executing tasks on large-scale data and return the results to the clients or companies. Hence, the clients can take advantage of the available computational power on a pay-per-use basis, even if they cannot get direct access to supercomputers. A server with large computational abilities can share its spare computation resources out of a financial incentive. For instance, Amazon Web Services Computing (AWS) provides on-demand delivery of computing power as described above. The process is cost-effective to the clients who can leverage the received results for effective decision making such as market investment choices based on the observed patterns. Also, due to the AWS infrastructure setting, clients can avoid decisions of capacity prior to deploying the application because they can easily scale it up or down when the server conducts the computation [6].

Despite the significant benefits offered by the computation outsourcing paradigm, the primary obstacle to its wide adoption is the security issue. Essentially, the data involved in large-scale computation may contain valuable or sensitive information. Because of the physical isolation between the clients and CSPs, the clients are unable to judge whether a given cloud server is trusted or not before outsourcing their computation tasks, which could lead to critical concerns. For example, in the case of a smart grid, the cloud would be responsible for the computation on a power company's data, which is collected from its residents. A curious cloud server may inspect the data and deduce behavioral information of the residents. If the information is captured by a malicious party, it can give rise to many illegitimate consequences, including theft and possibly even terrorism. Therefore, in many cases, clients are unwilling to share the data with others including CSPs, even though the latter claim that their servers are completely secure and can be fully trusted. Besides, a trusted cloud can also suffer from external threats. An external attacker can exploit technical vulnerabilities of a cloud service to gain further access to the data residing in the cloud. The outsourced data is intrinsically not secure from the point of view of cloud clients, and thus it is highly desired that the servers should not obtain any information about the data used in the computation. Practical privacy and security protection measures should be in place to ensure that cloud computing is appealing to a large range of clients.

Other than the challenges of data confidentiality, another crucial challenge is to ensure computation integrity. In the context of computation outsourcing, this is also called result verifiability (or checkability). In fact, clients should have the ability to check correctness of the results of outsourced computation. The cloud server assigned to a task may not honestly conduct the computation and may simply return an invalid result due to financial or timing reasons. For example, a cloud server can randomly generate the output based on the size and characteristics of the computational task. The computational cost of doing so is often very minor compared to the cost of running the computation itself. Moreover, an honest cloud server may undergo a software or hardware failure during its computation or data transmission. Many articles in the literature have addressed this challenge and carefully designed the verification procedure. It is required that the procedure should be substantially more efficient than the time of executing the original task. Otherwise, it will conflict with the motivation for computation outsourcing. To achieve efficient verification, certain properties of the function used in the computation are often exploited to verify the returned solution.

Recent research has made steady advances in addressing these concerns, focusing on large-scale engineering and scientific computing problems and computationally intensive applications. One line of research originated from [63] provides a general mechanism for secure computation outsourcing. The solution combined fully homomorphic encryption (FHE) [66] with the evaluation of Yao's garbled circuits [145], achieving both data confidentiality and result verifiability. However, applying this general mechanism to large-scale computations is currently far from practical. Due to this reason, this line of theoretical research is not the primary focus of this survey. In contrast, another line of research aims to design solutions of immediate applicability to computation outsourcing. Such publications can be broadly classified into two groups: secure outsourcing of fundamental functions and secure outsourcing of computational tasks for specific applications. Constructions from the first group focus on commonly used mathematical operations or functions such as matrix operations, linear equations, and mathematical optimization problems. They typically exploit properties of the underlying mathematical functions to design specific protocols that can be utilized as building blocks in more complex computations. On the other hand, publications from the second group usually start from an individual application scenario and design an outsourcing solution for the entire task at hand.

Thus, this survey aims to provide an organized overview of the state-of-the-art solutions addressing security issues in computation outsourcing. Before proceeding with the description of individual constructions and their properties, we lay down the general framework in Section 2, including common system models, general security requirements, and various tradeoffs. Although general theoretical techniques for secure computation outsourcing, such as FHE, are not the main focus of this survey due to their current impracticality for large-scale tasks, we briefly discuss them in Section 3. Then in Section 4, we discuss solutions for securely outsourcing fundamental functions, from the simpler to more complex operations. Section 5 covers secure outsourcing of application-based computations for different computational domains. An in-depth security analysis and comparison of many constructions is presented in Section 6. Consequently, Section 7 presents performance evaluation of the constructions discussed in earlier sections. Open issues and suggestions for future research directions are identified in Section 8. Finally, we conclude the paper in Section 9.

2 SECURE COMPUTATION OUTSOURCING: THREATS, REQUIREMENTS AND EFFICIENCY

In this section, we first describe the system architecture of outsourcing computation applicable to most of the related work. Then, we demonstrate typical security threats and some corresponding requirements. Following that, we identify schemes similar to, but not identical to those using the secure computation outsourcing model. Lastly, we briefly discuss the balance between security and efficiency at the end of the section.

2.1 System Architectures for Secure Computation Outsourcing

A common secure computation outsourcing architecture is illustrated in Figure 1. A typical asymmetric system involves two main different entities: a client C (or customer, data user, etc.) and a cloud server CS (or server). Due to the inability to carrying out the desired computation, a client C would like to outsource an expensive computational task Φ to a cloud server CS, who possesses massive computational power and significant storage capacity. Because the cloud server CS may

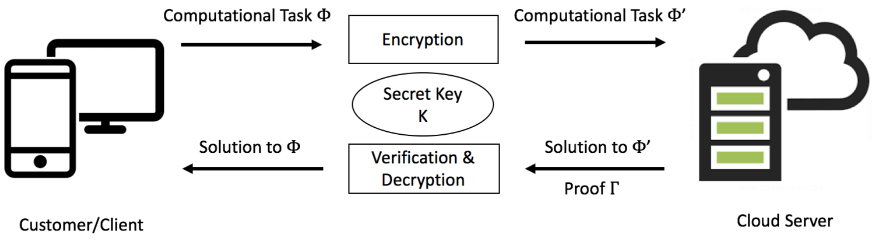

Fig. 1. A system architecture with sequences of data/message flows.

not be fully trusted, the client C can locally apply a secret key K to transform Φ into an encrypted problem Φ' to protect data privacy. Then the client C outsources the problem Φ', in place of Φ, to the cloud server CS and wants to receive the solution of Φ'. The cloud server CS operates on a pay-per-use basis and applies its resources to solve Φ'; it consequently returns the solution of Φ' to the client C together with a proof Γ. Note that the proof Γ is optional and is generated to allow the client to verify that the solution supplied by the CS is correct. After obtaining the returned solution from the cloud server CS, the client C recovers the answer of Φ using the secret key K. Moreover, the client C validates the correctness of solution by checking the solution itself or verifying the proof Γ. Based on the verification result, the client can choose to accept or reject the solution. Also, the cloud server CS may store an encrypted database related to the task uploaded by the client beforehand. The client C and the cloud server CS may engage in several rounds of interaction to solve the computation completely.

Note that some proposed system models may contain another independent party for a specific design objective: a party which may be responsible for results verification, data aggregation, or another function (e.g., see [79, 109]). Two or more servers can work independently, e.g., for result verification. The servers can also work collaboratively. However, in either case they are generally assumed to be non-colluding. In some application scenarios, the client's role in the system can be decoupled into two distinct parties—the data owner and data user—to fit the actual situation. This case is mostly present in collaborative outsourced data mining, such as [146]. Additional details and the variants of the system architecture will be discussed in Section 6.

2.2 Security Threats in Computation Outsourcing

On a positive side, the use of computation outsourcing lowers some of the currently existing security risks for clients. For instance, clients who upload their data to a cloud and no longer store everything locally face a reduced risk of having sensitive information leaked in case their laptop is lost or stolen [115]. However, new challenges and threats to information assets residing in the cloud are introduced because the data stored remotely is out of users' control. In the context of cloud computation, these security threats can be categorized as concerning *data confidentiality* (which refers both to the data used in the computation, i.e., computation input, and to the solution or output of the computation) and *computation integrity.*

From the client's perspective, threats to data confidentiality potentially come from the cloud service provider itself. Thus, it is often required that the cloud server should not gain any knowledge of the possibly sensitive client's data. To model the server's behavior, two types of adversarial models are usually considered: The first one is called the "honest-but-curious," or semi-honest, model [72]. In this model, the server is assumed to faithfully follow the protocol's steps and thus correctly execute the computation and return the correct result to the client. Meanwhile, the server still tries to learn sensitive information about the client's data or the computed results in order to profit from it based on the nature of the computational tasks. The other, stronger security model treats the server as a fully malicious entity that can arbitrarily deviate from the prescribed computation. Then a malicious server can deviate from the computation in the attempt to learn more information about the client's data than what an honest-but-curious server could or it may want to save its own computational resources (such as energy and time) and intentionally return an incorrect result (such a randomly chosen output) to the client. By returning a seemingly valid but wrong result, the cloud server hopes that the client will not be able to detect the cheating. A server who skips a portion of its computation is also sometimes referred to as a "lazy" server in the literature. Some publications (such as [149]) also define a lazy server as one who attempts to lower its work and assumed to not intentionally disrupt the computation by investing more time that what is necessary to complete its computational task.

In addition to internal problems, a cloud server might suffer from external attacks. External threats include remote software or hardware attacks against the cloud infrastructure or application and social engineering. Successful break-ins into the cloud infrastructure both expose the data its servers handle to external parties and can compromise correctness of the returned result if the computation becomes corrupt. From the client's view, any detected problems (such as, e.g., incorrect output returned by the server) will be attributed to the server's misbehavior, regardless of whether they were triggered internally or externally.

Threats in the other direction—originated at the client and intended to harm the cloud server—are not typically discussed in the secure outsourcing literature. Publications that address threats of malicious clients corrupting workloads or attempting to steal information from tasks of other clients sharing the cloud's infrastructure are present in the literature, but their overview is beyond the scope of this survey.

2.3 Requirements for Secure Computation Outsourcing

Recall from Subsection 2.1 that the original computational problem Φ is to be locally transformed into Φ' by the client. As a result, only Φ' is accessible to the cloud server as the input. Then one of the major security requirements is that the cloud server cannot derive any sensitive or meaningful knowledge about client's data from Φ', which is known as *input privacy*. Furthermore, after evaluating problem Φ' and determining the corresponding output of the function, the cloud server should be unable to learn any information about the result of executing Φ itself, including any intermediate and final results. This is called *output privacy*. In the majority of the constructions that comply with the previously described architecture, input and output privacy is achieved by either data transformation or data encryption. Then to guarantee that the server learns not information about the client's data, the server's view of the outsourced data (in a transformed or encrypted form) should be computationally or statistically indistinguishable from randomly sampled values. Because of slight differences in the system architecture employed by some schemes, the privacy requirement might be formulated differently, as we detail later in Section 6.

Verifying correctness of the result returned by the cloud server and ensuring that a true answer is obtained has been widely acknowledged as another crucial security requirement of computation outsourcing. This property refers to computation integrity and is called *checkability* [79] or *verifiability* [63] in the literature. It requires that an incorrect output to problem Φ' returned by a malicious (or lazy) cloud server should pass the verification process on the client side with a very small or negligible probability. Note that the checkability is occasionally not a necessary requirement in some publications due to the particularity of the functions or a weaker threat model.

As another major requirement for secure computation outsourcing, *efficiency* commonly refers to the client's ability to reduce its local computation, which serves as the main motivation for utilizing computation outsourcing. The savings are based on the difference between the effort required for executing the computational task Φ locally and the effort involved in using computation outsourcing, typically measured theoretically or in some cases empirically. Computation associated with computation outsourcing involves preparation of Φ' including data encryption, result recovery including decryption, and output verification. The cost of input and output transformation or encryption/decryption depends on the data size and the employed encryption techniques, which are often symmetric.

Lastly, if the server conducts the computation faithfully and sends the correct results to the client, the client should be able to successfully verify and recover the result of the computation. This property is known as *correctness* [116].

2.4 Overview of Related Topics

According to the system architecture and requirements described in Subsections 2.1 and 2.3, we identify two classes of related work, on which we comment below.

Secure multi-party computation (MPC) allows for cooperative evaluation of an arbitrary function by multiple parties, taking each party's private data as the input and preserving data confidentiality throughout the computation. The result of the computation is returned to an agreed-upon set of participants according to the specified functionality. The earliest general solutions for secure function evaluation were given by Yao for the two-party setting secure in the presence of semi-honest participants [144] and by [72] for the multi-party setting secure against malicious participants. The general architecture of secure MPC typically assigns symmetric workloads to the computational parties, and the data contributed by every party resides in the system in a protected form. This setup is typically not suitable for achieving the goals of outsourcing large-scale computations, where a client wishes to improve the speed of running computational tasks[1].

Delegating computation with cheating detection assumes a network composed of several computational devices of different computational capabilities that interact with each other. It allows weak devices to delegate their computational tasks to more powerful devices, which is similar to the architecture of securely outsourcing large-scale computations. Several early results include provisions for detecting server's misbehavior. However, they permit the server to have access to the data used in the delegated tasks in the clear, which violates one of the main security requirements of secure computation outsourcing. These publications include [53, 73, 74], and others. In addition to these early studies, several more recent results on delegating specific computations such as those in [38, 142] treat both computation verification and data privacy so that they can be classified as secure outsourcing schemes.

2.5 Comparison to Other Surveys

We also identify several other related survey articles on secure computation outsourcing, which have a different scope and focus from those of this survey. In particular, the work of [50] provides a a survey of privately and publicly verifiable computing techniques. Its emphasis is on verifiable computing achieved via cryptographic means such as proof and argument systems, homomorphic and functional encryption, and others. Meanwhile, a review of the existing approaches for secure outsourcing computation based on homomorphic encryption is given in [62] and [123]. Note that the scope of these surveys is different from our main focus, as we cover schemes for a wide

[1]Some secure multi-party computation techniques are suitable as solutions for secure computation outsourcing with multiple servers, which we will discuss in Subsection 6.1.

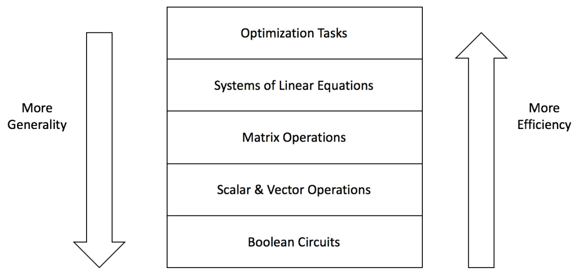

Fig. 2. A hierarchy of computational levels.

range of specific computational tasks of practical performance. Also, these surveys do not consider schemes that involve more than one server. In [36] and [112], secure outsourcing of several scientific computational tasks is discussed. The set of functions covered in these articles, however, is relatively limited. In this survey, on the other hand, not only do we provide a comprehensive overview of the constructions for securely outsourcing fundamental functions, but also cover constructions for application-specific tasks of practical relevance. Lastly, [2] treats both privacy and integrity of outsourced computation, including function-independent and function-specific computation, but these topics are given relatively small coverage because the scope of the article is broader and covers additional security properties in cloud storage and cloud computing.

2.6 Tradeoffs between Efficiency and Generality

When outsourcing computation to a public cloud, it is necessary to ensure confidentiality of the data used in the computation while maintaining relative efficiency of the computation. Data protection can be implemented via different mechanisms, and transformation or encryption techniques adopted in different schemes in the literature reflect the tradeoffs between efficiency and generality. On the one side of spectrum, some schemes focus on solving general computational tasks. These general solutions, such as those employing FHE, aim to support any computable function, which can be represented as an arithmetic or Boolean circuit. In such schemes the server will be able to evaluate the circuit in encrypted form and achieve proper data protection. However, these mechanisms have been criticized as impractical because of their high overhead and large circuit sizes, especially for large-scale computational tasks. This motivates efficiently solving specific computational problems instead of only treating general approaches. Thus, schemes on the other side of spectrum aim to provide an efficient outsourcing strategy for computing a specific task over large-scale data, but the developed solution may not generalize to any other type of computation.

In this survey, we propose to categorize the available solutions in the literature based on the generality of the technique. We organize operations, from elementary to more complex, in a hierarchy of computational levels (first introduced in [129]) as shown in Figure 2 to systematically classify and organize these solutions. This is motivated by the fact that many constructions achieve similar levels of security and cannot be easily categorized based on the security properties they achieve, while using a hierarchy of computational levels allows us to arrive at a well-organized structure for presenting these solutions.

The general classification methodology is based on the following observation: any computational problem can be represented as computations at different computational levels, which can be organized in a hierarchy. A complex task at a top level of the hierarchy can be decomposed into simpler operations at lower levels of the hierarchy. Thus, secure outsourcing solutions for computations at lower levels can be used as building blocks for securely outsourcing a computation at a higher level, potentially resulting in multiple solutions to the task at hand. We demonstrate this on the example of linear programming.

If there exist proper secure outsourcing protocols, the client can decompose its task Φ into a number of sub-tasks at lower levels of the hierarchy, transform each of them into the corresponding protected form, and re-assemble the result from the outputs of the sub-tasks that it receives. For example, instead of directly outsourcing its linear programming task, the client can separately outsource matrix operations or vector operations to a cloud server. This can often increase the client's work due to function decomposition and result assembly or it may also increase the number of interaction rounds.

As an observation related to function decomposition at the client side, we note that in some cases this will allow the client to hide the specifics of the function being outsourced. While throughout this article the task being outsourced is assumed to be known to the cloud and outsourcing of private function evaluation is beyond the scope of this survey, there is potential for this type of function decomposition to provide a limited form of function privacy. For example, in the case of decomposing a linear programming task into small components, the cloud server might not be aware what exactly the client's task is. We do not treat this topic further in this survey.

In the rest of this survey, we present techniques from the most general at the bottom of the hierarchy to more specific moving up in the hierarchy. Section 3 treats outsourcing of Boolean circuit evaluation using encryption. In Section 4, we survey outsourcing of fundamental functions related to different computational levels of the hierarchy, from scalar operations to optimization problems. Moreover, there are a number of application-specific solutions which are often too complex and not always compatible with the hierarchy in Figure 2. We discuss them in Section 5, categorized by the application domain.

3 ENCRYPTION TECHNIQUES FOR OUTSOURCING GENERAL FUNCTIONS

As mentioned in Section 2, data encryption is a basic way to provide data confidentiality. Traditional symmetric encryption schemes such as AES with strong security guarantees have been long supported as a security measure by leading cloud service providers. However, these conventional algorithms are not able to support computation over encrypted data. Homomorphic encryption (HE) is a type of cryptosystem that allows certain operations to be carried out on encrypted data without decrypting it. Early homomorphic cryptosystems such as RSA [111], El Gamal [56] and Paillier [104] can only support a single operation on ciphertexts such as addition, multiplication, or XOR and are called partially homomorphic. New cryptographic solutions for computation outsourcing became possible after Gentry's discovery of the first viable fully HE (FHE) which solved a long-standing major problem in cryptography and theoretical computer science [65, 66]. FHE allows for an unlimited number of additions and multiplications to be performed directly on encrypted data, and hence it allows for evaluation of an arbitrary functionality on encrypted data, represented as an arithmetic circuit. Notwithstanding, the overhead associated with the currently available FHE techniques makes their use for non-trivial computations infeasible. Aiming to address this performance issue, the area has experienced new research advances.

In this section, we first describe how FHE can be employed for securely outsourcing any computable function represented as a Boolean circuit and give an overview of landmark results that provide optimizations and implementation of FHE and related techniques. Consequently, we discuss efficiency issues.

3.1 Outsourcing Evaluation of Boolean Circuits

Yao's garbled circuit (GC) evaluation [144, 145], proposed in the 1980s, provides a general approach for secure two-party computation. It is a constant-round protocol for securely evaluating an arbitrary function represented as a Boolean circuit. To preserve data privacy, a circuit has to be garbled anew for each evaluation and the approach is secure in the presence of a semi-honest garbler and malicious evaluator. Capitalizing on the fact that the evaluator is unable to predict (garbled) output without evaluating the function, Gennaro et al. [63] combined the technique with FHE to enable secure and verifiable outsourcing of arbitrary functions. In this solution, the client supplies a garbled circuit once and it can be evaluated by a cloud server in encrypted form on multiple inputs in such a way that cloud's misbehavior is detected by the client. While providing a conceptually elegant and general secure outsourcing solution, the construction suffers from the significant computational burden brought by FHE and the need to decrypt ciphertexts inside FHE. Thus, performance of secure outsourcing of Boolean circuit evaluation would greatly benefit from optimized garbling schemes or improved FHE constructions, which are popular lines of research.

In particular, early research on developing efficient GC protocols focused on making implementions of MPC practical MPC, starting from the work of [94]. In 2012, Bellare et al. [13] for the first time treated GC as an actual primitive, called garbling scheme, and provided a scheme based on a dual-key cipher (DKC). In this scheme circuit evaluation required one or two calls to a fixed-key blockcipher per gate, which was subsequently improved in [12] to use a single permutation call per gate and be compatible with other efficiency improvements such as "free" XOR gates and garbled row reduction. Recently, a sequential construction of GC has been proposed in [119] to achieve compactness and scalability.

The design of different FHE constructions can be divided into three different families based on the underlying mathematics [39]: (i) schemes based on ideal lattices [66], (ii) schemes over the integers [39] that rely on the hardness of finding an approximate greatest common divisor (GCD) of large integers, and (iii) schemes based on the Learning with Errors (LWE) and Ring Learning with Errors (RLWE) assumptions [29]. We review recent advances in each category.

The FHE scheme based on ideal lattices proposed by Gentry in [66] requires that the user first sets up a scheme that supports only a finite number of multiplications. This scheme is referred to as Somewhat Homomorphic Encryption (SwHE). To reduce the noise accumulated through successive homomorphic multiplications, the encrypted message is to be re-encrypted. This is done by "squashing" the decryption circuit so that it can be handled within the capacity of the SwHE scheme. This operation allows us to obtain a new ciphertext with a reduced amount of noise that encrypts the same plaintext, and this process is called bootstrapping. As a result, the construction can support an unbounded number of homomorphic operations with bootstrapping applied as needed. To improve performance of this scheme, Gentry and Halevi [68] significantly reduce the asymptotic complexity of key generation for the underlying SwHE and introduce a batching technique for encryption. They follow the work of Smart and Vercauteren [117] who also built an FHE from a SwHE scheme with small ciphertext and key sizes using a specific type of lattices that can be represented by integers. Meanwhile, a new method for building FHE is given in [67]. The authors propose a hybrid scheme of SwHE and a multiplicatively HE scheme to eliminate the need for the squashing step. Early FHE constructions following Gentry's work were lattice-based. The challenges of schemes in this category are large key and ciphertext sizes. Lattice-based cryptography also contributed to advances in other cryptographic areas, including a new framework for constructing an attribute-based encryption scheme [23] and a homomorphic signature scheme [75].

The schemes over the integers also follow the construction framework of the initial proposal by Gentry to achieve conceptual simplicity. The first scheme of this kind was proposed by van Dijk et al. [126], to which we refer as DGHV. The security of the construction is based on the difficulty of finding an approximate integer GCD and their SwHE scheme uses only integer arithmetics for homomorphic addition and multiplication. A follow-up optimization work by Coron et al. [44] stores key elements in a new form which allows the size of the public key to be reduced, while ensuring semantic security under a stronger approximate GCD assumption. The key size is further reduced in [45] by using a compression technique and a technique called modulus switching is introduced which allows bootstrapping to be eliminated. These two schemes exhibit performance similar to those of existing lattice-based schemes. Lastly, the work of Cheon et al. [39] extends the DGHV scheme with the capability of using several plaintext bits in a single ciphertext while providing semantic security relying on the hardness of the approximate GCD.

The logic for constructing LWE-based FHE schemes was proposed in [29]. After building a SwHE scheme from LWE, one can apply a key-switching technique to control the dimension expansion of a ciphertext that results from homomorphic multiplication of other ciphertexts as well as a modulus-switching technique to manage the noise. The iterations of these two techniques result in an FHE scheme. Construction of RLWE-based FHE schemes follows a similar process. Batched RLWE-based schemes were introduced in [70] and [28] to reduce the cost of homomorphic evaluation to be polylogarithmic (in the security parameter). The latter is known as the BGV cryptosystem and is more efficient than the Gentry's initial proposal because of the absence of bootstrapping technique, which is replaced by a technique called modulus switching. However, we note that all of the constructions in this category can only provide evaluations of circuits of a polynomial (in the security parameter) depth, also known as leveled FHE.

As of time of this publication, LWE- and RLWE-based FHE has still been primarily of theoretical interest. Because modulus switching operations in BGV are still not within the reach practicality, Brakerski [27] presented a scale-invariant scheme without switching the modulus. Consequently, Gentry et al. [64] proposed a relatively simple FHE scheme based on LWE—known as the GSW cryptosystem—where matrix addition and multiplication are used to represent homomorphism. It was later shown in [30] and [4] that GSW achieves polynomial-factor growth in the error rate, which is significantly lower than the quasi-polynomial growth rate in concurrent work [3]. In addition, a variant of multi-key FHE based on a variant of the NTRU computational problem was proposed in [93], called the LTV scheme. This scheme enables computation on values encrypted under multiple and unrelated keys. Lastly, Bos et al. [25] showed how Brakerki's noise-management technique [27] could be applied to the multi-key LTV FHE scheme [93]. Note that most of the advances in FHE cryptosystems based on LWE and RLWE can be ported to FHE constructions over the integers. For example, instead of using modulus switching for controlling noise in schemes over the integers, Coron et al.[43] apply the technique of [27] to reduce the growth of noise to linear in the multiplicative depth of the evaluation circuit.

Another line of work on improving efficiency of FHE schemes focuses on optimizing SwHE constructions. Naehrig et al. [98] implemented a SwHE scheme based on RLWE which enjoys relative efficiency. By converting between different message encodings in a ciphertext, they were able to further optimize application-specific realizations. In addition, a parallel computing technique called Single Instruction Multiple Data (SIMD) was used in [69] to improve the speed of FHE operations. It was targeted at the main bottleneck in bootstrapping which requires homomorphically reducing one integer modulo another. Another work [118] designed a SwHE scheme which supports SIMD as well as computation in finite fields. It used SIMD operations for parallel re-encryption, which led to a considerable performance improvement. Furthermore, the work [105] focused on minimizing the number of bootstrapping operations so as to improve computation time using FHE. A number

Table 1. Performance of AES encryption using FHE on the same platform (Source: [48]).

Technique	Time per Block	Technique	Time per Block
GHS (SIMD) [71]	2400 s	Accelerating NTRU [48]	7.3 s
NTRU [51]	55 s	HElib [76]	2 s

of software [71, 77, 78] and hardware [84, 108, 136] implementations of SwHE/FHE schemes have emerged, aiming to achieve a significant performance improvement of the available constructions.

3.2 Performance Issues

As mentioned in Subsection 2.6, using FHE for evaluation of client's outsourced tasks results in representing computation as a circuit, and the approach supports any desired functionality. As seen from Figure 2, this type of outsourcing resides at the bottom of the computational hierarchy. Even though a number of breakthroughs have been recently made to improve performance of FHE, secure computation outsourcing based on FHE and Boolean circuits remains to be impractical for most applications (such as those specified at higher levels of the computational hierarchy), especially when input datasets are large.

Note that comprehensive benchmarks for comparing performance of software implementations of FHE schemes are currently not available [24]. A number of publications, however, report on performance of evaluating AES-128 using FHE. For example, Gentry et al. [71] was the first to provide a software implementation of FHE based on a modified HE scheme of [28] using the techniques in [118], [70], and new optimizations. A customized implementation of the LTV scheme is available in [93]. Another work [48] builds a library that uses discrete Fourier transform to support FHE evaluation of AES. This implementation brings the evaluation time down to slightly over 7 seconds per block. Shortly after, HElib [77]—a software library that implements homomorphic evaluation of AES encryption—brings the amortized per-block time down to about 2 seconds. Table 1 gives a brief comparison of performance of different FHE implementations evaluating AES encryption. Note that in the secure outsourcing scheme of [63], homomorphic decryption of an AES circuit (or a similar operation) will need to be performed for each gate of the Boolean circuit representing the outsourced task, effectively multiplying the times in the table by the number of circuit gates.

The main conclusion that we can make is that this general approach of secure computation outsourcing is currently far from practical for engineering applications, even for moderate-scale computations. This observation motivates scholars to seek efficient solutions for securely outsourcing specific types of computation, operating at higher levels of the computational hierarchy. We consequently discuss solutions of this kind in Sections 4 and 5.

4 SECURE OUTSOURCING OF FUNDAMENTAL FUNCTIONS

In the context of computation outsourcing, solutions at the level of the entire task that provide proper input/output privacy are often preferred because they save client's computational cost. There are many publications that focused on constructing schemes for secure outsourcing of fundamental mathematical functions. Some of these outsourcing techniques can be directly used for stand-alone tasks, while others may work as building blocks of more complex computational tasks. In this section, we give an overview of existing techniques for outsourcing these fundamental functions which frequently appear in large-scale data analytics. The order in which we present them follows the hierarchy in Figure 2.

4.1 Scalar Operations

As an elementary component of general operations, operations on scalars (i.e., numeric values) refer to binary operations—such as addition, subtraction, multiplication, division, square root, exponentiation, etc.—and scalar products. Among scalar binary operations, exponentiation modulo a large integer is known as a costly operation in cryptographic schemes [38]. Hence, securely outsourcing this operation has been considered in many recent studies. We will discuss this specific computation in Subsection 5.5. Scalar addition and multiplication, on the other hand, can be performed efficiently even in computationally constrained environments and therefore are often not amenable to cost-effective outsourcing solutions.

The *scalar product* operation is defined as the summation of the element-wise products of two vectors of the same size. The objective is to keep the elements of the two private input vectors from being revealed during the protocol. Solutions from secure multi-party literature such as [52, 54, 125] assume that each client holds a vector and is actively engaged in the computation to determine the result, which does not fit the outsourcing model. Liu et al. [58] proposed a solution called the Protocol for Outsourced Scalar Product (POSP) with the assumption that clients separately maintain their own private vectors. These vectors are encrypted and outsourced to the server. On receiving a query from one client, the server performs some computation and interacts with the second client prior to returning the result to the client who originated the computation. The server and either client are unable to learn information about the other client's data other than the result of the computation. The client's work is linear in the input size, but the design suffers from a number of disadvantages such as expensive computation and the need to clients to remain online and respond to the server during the computation.

Computing the distance between two vectors is widely used in biometric computations. Its secure outsourcing approaches also assume multiple clients, as proposed in [19] and [21]. We discuss biometric-related computation in Subsection 5.3. Outsourcing of scalar operations is also useful in many different areas such as cryptography or data mining. These operations typically act as building blocks for more complex computational tasks at higher levels of the hierarchy in Figure 2. It is worth mentioning that existing solutions usually apply HE to provide strong privacy protection during outsourcing of scalar operations. The tradeoff is that most of these schemes suffer from inefficiency issues.

4.2 Matrix Operations

As we move up higher in the computational hierarchy, we arrive at matrix operations which can be realized as a combination of several vector operations. However, individually outsourcing computation for each row and column of large matrices is not advisable due to massive computing cost. There have been many studies focusing on securely outsourcing a matrix operation as a single task. In this subsection, we survey the state-the-of-art solutions for large-scale outsourcing of different matrix operations.

4.2.1 Matrix Multiplication. Let $\Phi = (\mathbf{X}, \mathbf{Y})$ denote a matrix multiplication task. The client first encrypts the task to obtain $\Phi' = (\mathbf{X}', \mathbf{Y}')$ and outsources it to a cloud server. The cloud server is expected to compute $\mathbf{M}' = \mathbf{X}' \times \mathbf{Y}'$ and return it to the client. The client decrypts $\mathbf{M}'$ to obtain solution $\mathbf{M}$ and verifies correctness of the result.

Matrix multiplication is one of the fundamental computational task rooted in scientific and engineering fields such as statistics, image processing, and several others. For $n \times n$ matrices, the conventional algorithm runs in $O(n^3)$ time, but several asymptotically faster solutions are available. For example, Strassen's algorithm performs matrix multiplication in $O(n^{2.807})$ time, while Coppersmith-Winograd's algorithm improves it to $O(n^{2.376})$. The latter, however, does not work well for large-scale matrices because of large constants hidden behind the big-O notation. When the dimension of matrices reaches 10^{20}, implementations adopt the former algorithm. In what follows, we will generally represent the complexity of matrix multiplication by $O(n^{\rho})$.

Outsourcing matrix multiplications with checkability has been studied in the literature. Benjamin and Atallah [14] give protocols for secure outsourcing of matrix multiplication using HE applied to the original and random matrices. Atallah and Frikken [8] were the first to develop a protocol that uses only one server and eliminates the possibility of server collusion (and thus compromised security) present in multi-server solutions. The principal thought is to apply secret sharing to decompose the original matrix into multiple matrices, where an element of each transformed matrix is just a secret share of the corresponding element in the original matrix. Then it is possible to delegate multiple matrix multiplication operations on secret-shared matrices to the server while protecting input privacy. The client can later use secret sharing to reassemble the result of the original matrix multiplications from the shares it receives. To provide stronger security, the authors extend the approach by using real and fake shares in combination with homomorphic encryption, where $t+1$ real and t fake shares are used with secret sharing set up for threshold t. However, this leads to the server performing at least $4t+2$ matrix multiplications in the base case, which significantly increases the server's computational burden and is not desired.

A solution that avoids the above performance issue, but offers weaker security guarantees is given in [86]. During the initial transformation, matrix $\mathbf{X}$ is modified to $\mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$, while $\mathbf{Y}$ is modified to $\mathbf{P}_2\mathbf{Y}\mathbf{P}_3^{-1}$, where $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are randomly permuted diagonal matrices multiplied by random values. More precisely, for $k = 1, 2, 3$ define $\mathbf{P}_k(i, j) = \alpha_{ki}\delta_{\pi_k(i),j}$, where for each k α_{ki} is a vector of random values, $\delta_{i,j}$ is the Kronecker delta, and $\pi_k(i)$ refers to a random permutation with inputs i. Randomness of both the permutations and random vectors is treated as the client's secret keys. Upon receiving $\mathbf{M}_K$ from the server, the client recovers the result as $\mathbf{M} = \mathbf{P}_1^{-1}\mathbf{M}_K\mathbf{P}_3$. To achieve checkability, the client uses the Monte Carlo algorithm (also called Freivalds's algorithm [61]) as follows: The client randomly generates a 0-1 vector $\mathbf{r}$ and computes $\mathbf{P} = \mathbf{X} \times \mathbf{Y}\mathbf{r} - \mathbf{M} \times \mathbf{r}$; the process is repeated with several new vectors $\mathbf{r}$. Any non-zero $\mathbf{P}$ indicates verification failure. The overhead is $O(n^2)$ for the client and $O(n^3)$ for the server, which is the same as in [8], but in practice it is several times lower because of the hidden constants.

Another thread of research studies secure protocols based on HE. Because secure matrix multiplication does not actually require FHE, Mohassel [97] studies verification of homomorphic matrix multiplication using partially HE. The work discusses several additively HE such as Paillier, Goldwasser-Micali, and El Gamal (with messages in the exponent) as well as BGN-style encryption that allows for multiple additions and at most one multiplication over ciphertexts. The author proved that if the underlying HE scheme satisfies associativity and distinctiveness, the simple algorithm developed in that work can be used to verify the result of matrix multiplication in $O(n^2)$ time. Block-wise representation of matrices is also suggested for large inputs.

Maintaining the same asymptotic complexities for both the client and the server, Zhang and Blanton [149] introduce new features in their proposed schemes by dividing the overall computation into several steps and associating a key with each stage. This way the client can efficiently determine the portion of the computation where the server fails to compute correctly. In their approach, the ability to successfully detect faulty elements in the returned matrix with probability 1 is called *deterministic verification*. The idea behind the transformation is to encode the elements of input matrices $\mathbf{X}$ and $\mathbf{Y}$ into two other matrices $\mathbf{X}'$ and $\mathbf{Y}'$ using a setting that admits bilinear maps. The elements of $\mathbf{X}'$ and $\mathbf{Y}'$ also encode secret relationships that the client generates and can efficiently compute for the product matrix. Security is proved under a variant of the Decisional Diffie-Hellman assumption. Fiore and Gennaro [59] proposed a solution with public verifiability

Table 2. Comparison of secure matrix multiplication outsourcing schemes.

Scheme	Underlying technique	Algebraic structure	Input matrices	Verification
[8]	Secret Sharing	Finite field $\mathbb{Z}_p$	Square matrices	Nondeterministic
[97]	Additive HE	Finite field $\mathbb{Z}_N$	Square matrices	Nondeterministic
[86]	Permutation	Infinite field $\mathbb{R}$	Any matrices	Nondeterministic
[59]	PRF	$\mathbb{Z}_p$	Any matrices	Deterministic
[149]	Additive HE	$\mathbb{Z}_p$	Any matrices	Deterministic

and achieve outsourcing of matrix-vector multiplication using pseudo-random functions (PRFs) with closed form efficiency. By applying the construction to each column of the input matrix, matrix multiplication can be performed with the client's work being linear the input size, as in other constructions. The authors also realize a multi-function verifiable computation scheme using a variant of PRFs. A comparison of several properties of matrix multiplication schemes is given in Table 2. Additional comparison of efficiency is provided in Section 8.

4.2.2 Matrix Inversion. Matrix inversion can be performed using several approaches which include Gaussian elimination, Newton's method [106], Cayley-Hamilton method [85], Blockwise inversion [15], and others. Using a divide-and-conquer technique, matrix inversion can be accomplished by computing the inverses of two half-sized matrices and performing six half-sized matrix multiplications. Hence, the time complexity of performing matrix inversion is the same as that of computing matrix multiplication [42].

Following their own work on matrix multiplication [86], Lei et al. [88] proposed a protocol for outsourcing matrix inversion operations that uses a similar transformation technique. During the transformation phase, the original matrix $\mathbf{X}$ is transformed into $\mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$, where $\mathbf{P}_1$ and $\mathbf{P}_2$ are randomly permuted diagonal matrices multiplied by random coefficients. Thus, the position of every element of $\mathbf{X}$ is randomly reorganized in the transformed matrix and is further protected by multiple random values. Using the Monte Carlo algorithm, random vectors are used in the verification step to assess correctness of the result.

In [97], Mohassel also proposes an algorithm for outsourcing matrix inversion that uses his solution for privacy-preserving matrix multiplication as a building block. To obtain the inverse of $\mathbf{X}$, the client first generates a random matrix $\mathbf{R}$. Then the client outsources matrix multiplication of $\mathbf{X}$ and $\mathbf{R}$ to the server as previously described to obtain $\mathbf{XR}$. Once the client recovers $\mathbf{XR}$, she sends it to the server who performs matrix inversion $(\mathbf{XR})^{-1} = \mathbf{X}^{-1}\mathbf{R}^{-1}$ and sends it to the client. The client verifies correctness of the received result using a similar approach to the one described before the client generates a random vector $\mathbf{r}$ and checks whether $(\mathbf{X}^{-1}\mathbf{R}^{-1})(\mathbf{XRr}) = \mathbf{r}$ holds. Finally, the client runs outsourced matrix multiplication of $\mathbf{R}$ and $\mathbf{X}^{-1}\mathbf{R}^{-1}$ to obtain the final result $\mathbf{X}^{-1}$.

It can be seen that the design of securely outsourcing matrix inversion operations borrows techniques from solutions to secure matrix multiplication outsourcing. The basic strategy is to use a random matrix to randomly permute and blind the elements of the original matrix.

4.2.3 Matrix Factorization. Unlike matrix multiplication and inversion, matrix factorization is a term that encompasses a number of different computational tasks such as non-negative matrix factorization (NMF), singular value decomposition (SVD), eigenvalue decomposition (EVD), etc., which we consequently describe.

NMF involves factorization of matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ with rank $r \leq min\{m, n\}$, where m is the number of features and n is the number of observations, from m-dimensional space into r-dimensional subspace. In other words, we decompose $\mathbf{X}$ as $\mathbf{X} = \mathbf{WH}$, where $\mathbf{W} \in \mathbb{R}^{m \times r}$ is known as the basis matrix and $\mathbf{H} \in \mathbb{R}^{r \times n}$ is called the coefficient matrix. These three matrices are required to have no negative elements. In the case of large-scale data, say, 1 million dimensions, it is desirable to find a small set of representative dimensions, say, 10 or 100 dimensions [121]. Hence, the problem can be approximated and approximate NMF is defined as using subspace of dimension r' such that $\mathbf{X} \approx \mathbf{WH}$, where $\mathbf{W} \in \mathbb{R}^{m \times r'}$ and $\mathbf{H} \in \mathbb{R}^{r' \times n}$. The approximate NMF problem is equivalent to minimizing the distance between $\mathbf{X}$ and $\mathbf{WH}$ under a certain cost function. Lin [91] gave an iterative algorithm for solving this problem in $O(Nmnr')$ time, where N is the number of iterations which can be set to a very large value.

Addressing the problem of outsourcing computation-intensive NMF, Duan et al. [55] proposed an outsourcing scheme that uses multiplication of permuted matrices. The client first transforms $\mathbf{X}$ into $\mathbf{X}' = \mathbf{PXQ}$, where $\mathbf{P}$ and $\mathbf{Q}$ are permutation matrices, and sends $\mathbf{X}'$, dimension r', and a stopping parameter ϵ to the server. The server returns $\mathbf{W}'$ and $\mathbf{H}'$ as the solution. In the verification phase, the client runs one iteration of the algorithm locally using $\mathbf{W}'$ and $\mathbf{H}'$ as the initial values and obtains a new solution $\mathbf{W}''$ and $\mathbf{H}''$. After computing the target function (i.e., the cost function) on both of the solutions, the client treats the received solution as valid if the difference between the two results of target function evaluation is below ϵ. If the received solution passes verification, the client recovers the solution to the original problem as $\mathbf{W} = \mathbf{P}^{\mathrm{T}}\mathbf{W}''$ and $\mathbf{H} = \mathbf{H}''\mathbf{Q}^{\mathrm{T}}$, where $\mathbf{A}^{\mathrm{T}}$ denotes the transpose of matrix $\mathbf{A}$.

EVD is another type of matrix factorization defined for square matrices. A symmetric matrix $\mathbf{X}$ can be factorized as $\mathbf{X} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ are n eigenvalues and $\mathbf{Q} = (q_1, \ldots, q_n)$ are the n eigenvectors associated with $\lambda_1, \ldots, \lambda_n$ [99]. In an outsourcing scheme, the client wants to get eigenvalues and the corresponding orthonormal eigenvectors of matrix $\mathbf{X}$. SVD is a different matrix factorization problem. It is defined as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where $\mathbf{U} \in \mathbb{R}^{m \times n}$ has orthonormal columns, $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix, and $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix [80].

Zhou et al. [153] designed protocols for outsourcing EVD and SVD of a matrix to a malicious cloud. For EVD, a client who would like to obtain eigenvalues and the corresponding orthonormal eigenvectors of matrix $\mathbf{X}$ first transforms it into $\mathbf{X}' = \mathbf{A}(\alpha\mathbf{X} + s\mathbf{I})\mathbf{A}^{\mathrm{T}}$, where α and s are random values and $\mathbf{A}$ is a random orthonormal matrix chosen by the client to achieve input privacy and $\mathbf{I}$ is the identity matrix. Hence, an eigenvalue λ' of $\mathbf{X}'$ should satisfy $\lambda' = \alpha\lambda + s$, where λ is an eigenvalue of $\mathbf{X}$. This transformation can guarantee the eigenvectors of the original matrix are orthonormal as long as the eigenvectors returned from the server are orthonormal. The server then returns $\mathbf{M}'$ and Λ' as eigenvectors and eigenvalues of $\mathbf{X}'$. To achieve efficient verification, this work applies the Monte Carlo algorithm (which was also used in [88]).

For SVD, three steps are required to compute the solution in [153] given matrix $\mathbf{X}$: (i) computing $\mathbf{Y} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{Y}^{\mathrm{T}}$; (ii) computing orthonormal eigenvectors and the corresponding eigenvalues of $\mathbf{Y}$, from which the singular values of $\mathbf{Y}$ can be computed as the square root of the eigenvalues; and (iii) computing orthonormal eigenvectors which form the columns of matrix $\mathbf{V}$ of $\mathbf{Y}^{\mathrm{T}}$. Let $\mathbf{U}$ and $\mathbf{V}$ denote the matrices the columns of which are the eigenvectors of $\mathbf{Y}$ and $\mathbf{Y}^{\mathrm{T}}$, respectively. Similar to the transformation used for EVD outsourcing, in SVD outsourcing the client is instructed to compute $\mathbf{X}' = \mathbf{A}(\alpha\mathbf{X})\mathbf{B}$, where α is a positive value randomly chosen by the client, and $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ are two distinct randomly chosen orthonormal matrices. It can be proved that the singular values of $\mathbf{X}'$ is equal to α multiplied by the singular values of $\mathbf{X}$. Also, the relationship between eigenvectors $\mathbf{U}'$ and $\mathbf{V}'$ associated with running the above three-step process on $\mathbf{X}'$ and eigenvectors of $\mathbf{U}$ and $\mathbf{V}$ computed using the above process for $\mathbf{X}$ satisfies $\mathbf{U} = \mathbf{A}^{\mathrm{T}}\mathbf{U}'$ and $\mathbf{V} = \mathbf{B}^{\mathrm{T}}\mathbf{V}'$.

Table 3 summarizes secure matrix factorization outsourcing techniques.

4.2.4 Matrix Determinant Computation. Determinant computation (DC) is known as another fundamental computation task widely used in scientific and engineering applications, especially in

Table 3. Comparison of secure matrix factorization outsourcing schemes

Scheme	Task	Masking technique	Masking matrices	Verification algorithm
[55]	NMF	Permutation matrices	Invertible	One more iteration
[153]	EVD	Random matrices	Orthonormal	Monte Carlo algorithm
[153]	SVD	Random matrices	Orthonormal	Monte Carlo algorithm

statistics [120]. The best currently known time complexity of DC is that of matrix multiplication. For large matrices, DC can be sped up by outsourcing the problem to a cloud server. It can be accomplished using the design of Lei et al. [87] where data privacy is achieved through LU decomposition. More precisely, given an $n \times n$ original matrix $\mathbf{X}$, the client first computes an $(m+n) \times (m+n)$ matrix $\mathbf{T}$ represented as

$$\mathbf{T} = \begin{bmatrix} \mathbf{X} & \mathbf{O} \\ \mathbf{B} & \mathbf{D} \end{bmatrix}$$

where $\mathbf{O}$ is an $m \times n$ zero matrix, $\mathbf{D}$ is a diagonal matrix with elements $(d_1, \ldots, d_m)$, $\mathbf{B}$ is a null matrix, and m is chosen by the client to aid data protection. The client then transforms $\mathbf{X}$ into $\mathbf{X}' = \mathbf{P}_1\mathbf{T}\mathbf{P}_2^{-1}$, where $\mathbf{P}_1$ and $\mathbf{P}_2$ are random permutation matrices, and sends it to the server. The server can use any suitable LU decomposition algorithm to compute the determinant of $\mathbf{X}'$ $\det(\mathbf{X}')$ together with the lower and upper triangular matrices $\mathbf{L}$ and $\mathbf{U}$, such that $\mathbf{X}' = \mathbf{LU}$. The client can then compute $\det(\mathbf{X})$ as $\det(\mathbf{X}) = \det(\mathbf{X}')\det(\mathbf{P}_2)/(\det(\mathbf{P}_1)\det(\mathbf{D}))$. The determinants of $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{D}$ can be efficiently computed by the client because they are special-form matrices. The verification design follows the Monte Carlo algorithm, and the client computes $\mathbf{L} \times (\mathbf{U}\mathbf{r}) - \mathbf{X}'\mathbf{r}$ using a randomly chosen vector $\mathbf{r}$ of n bits (as before, the computation is repeated several times with different $\mathbf{r}$ to provide sufficient correctness guarantees).

Note that the work [97] also gives an algorithm to compute the minimal polynomial of a matrix in $O(n^2 \log n)$ time using as building blocks matrix multiplication and inversion also present in that work. Given matrix $\mathbf{X}$, its rank and determinant can be computed from the constant coefficient of the minimal polynomial of the product $\mathbf{DUXL}$, where $\mathbf{U}$ and $\mathbf{L}$ are $n \times n$ random Toeplitz matrices and $\mathbf{D}$ is a random diagonal matrix.

4.2.5 Brief Discussion. Matrix operations have close relationships with the layers above and below it in the computational hierarchy of Figure 2. For example, matrix multiplication and determinant computation can be performed using a combination of scalar operations. However, most of solutions for secure outsourcing of matrix operations assume a single user-single server architecture, while schemes on outsourcing scalar operations often employ a different setup. Computing the inverse of a matrix can be accomplished by solving a system of linear equations as described in the next subsection. Furthermore, in practical large-scale eigenvalue methods, factorization of general matrices can be obtained by a back-substitution procedure in the QR algorithm [60]. This computation amounts to solving a system of linear equations. Note that there are more matrix operations than those listed in this section such as, e.g., matrix rank computation, matrix transpose, LU decomposition, Frobenius inner product, and others. These operations are either too trivial to be considered for outsourcing or are used as components of the tasks that we discussed. Additional analysis of the security properties offered by matrix operations schemes and their strength is available in Section 6.

4.3 Systems of Linear Equations

A system of linear equations (SLE), or a linear system, is a collection of two or more linear equations over the same set of variables. Solving a system of linear equations amounts to finding the values of the variables satisfying all linear equations [47]. Among the algorithms for solving a linear system, one approach is to formulate the problem in a matrix form as $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ is a vector with n unknown variables, $\mathbf{A}$ is a square matrix of coefficients of full rank n, and $\mathbf{b}$ is an n-element vector with constant terms. Because matrix $\mathbf{A}$ is non-singular, the unique solution to the system can be determined as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Note that outsourcing the inverse of a matrix is a computational task at a lower level. Using it for this problem would result in the client outsourcing to the server computation of the inverse of $\mathbf{A}$ by means of LU decomposition and then locally computing $\mathbf{A}^{-1}\mathbf{b}$. This requires the client to do $O(n^2)$ work, but this approach is still computationally expensive for the server for large systems of linear equations. When the size of the data is large, a better solution is to use an iterative approach which is more efficient for the server in terms of both memory consumption and CPU time. The idea is to start with an initial approximation of the solution and refine it through an iterative process [7]. When the approximated solution becomes sufficiently accurate, it is output as the solution to the linear system.

4.3.1 Secure Outsourcing Solutions. Wang et al. [129] were the first to provide an outsourcing solution for a large system of linear equations using the iterative approach. Given a SLE $\mathbf{Ax} = \mathbf{b}$, the coefficient matrix is represented as the sum of a diagonal matrix $\mathbf{D}$ and the remaining matrix $\mathbf{R}$, i.e., $\mathbf{A} = \mathbf{D} + \mathbf{R}$. This allows us to rewrite the equation $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ as $\mathbf{x} = -\mathbf{D}^{-1}\mathbf{Rx} + \mathbf{D}^{-1}\mathbf{b}$. The iterative equation for the kth iteration can then be written as $\mathbf{x}^{(k)} = \mathbf{Tx}^{(k-1)} + \mathbf{c}$, where $\mathbf{T} = -\mathbf{D}^{-1}\mathbf{R}$ and $\mathbf{c} = \mathbf{D}^{-1}\mathbf{b}$. The protocol then instructs the client to compute and encrypt $\mathbf{T}$ using additively HE to provide data privacy (note that it is easy to compute $\mathbf{D}^{-1}$ and $\mathbf{D}^{-1}\mathbf{R}$) and send the corresponding ciphertext $Enc(\mathbf{T})$ to the server. To protect vector $\mathbf{b}$, the client additionally chooses a random vector $\mathbf{r}$ and rewrites the original problem as $\mathbf{Ay} = \mathbf{b}'$, where $\mathbf{y} = \mathbf{x} + \mathbf{r}$ and $\mathbf{b}' = \mathbf{b} + \mathbf{r}$. Thus, the computation in the kth iteration can now be revised as $\mathbf{y}^{(k)} = \mathbf{Ty}^{(k-1)} + \mathbf{c}'$, where $\mathbf{c}' = \mathbf{D}^{-1}\mathbf{b}'$. Now the task $\Phi = (\mathbf{A}, \mathbf{b})$ has been transformed into $\Phi' = (\mathbf{T}, \mathbf{c}')$, and the client outsources the computation to the server with an initial guess $\mathbf{y}^{(0)}$ for vector $\mathbf{y}$. In the kth iteration, the server computes $Enc(\mathbf{T} \cdot \mathbf{y}^{(k-1)})$ using homomorphic properties of the encryption scheme and sends it to the client. The client decrypts the ciphertext and sets $\mathbf{y}^{(k)} = \mathbf{Ty}^{(k-1)} + \mathbf{c}'$. If the convergence condition is not met, the client sends $\mathbf{y}^{(k)}$ to the server for another iteration. At the end of the computation, the client recovers the result as $\mathbf{x} = \mathbf{y} - \mathbf{r}$. As far as verification of correctness goes, in addition to checking the convergence condition $||\mathbf{y}^{(k-1)} - \mathbf{y}^{(k)}|| \leq \epsilon$, the client is also instructed to verify that $||\mathbf{Ay}^{(k)} - \mathbf{b}'|| \leq \epsilon$ in order to avoid accepting the solution if a malicious (or lazy) server reuses and returns the ciphertext computed in the previous iteration. Furthermore, to detect server's other arbitrary deviations from the computation, the client tests correctness of $\mathcal{L}$ iterations in a batch. To do so, the client chooses $\mathcal{L}$ random numbers α_i of length l and checks whether $\mathbf{T}(\sum_{k=1}^{\mathcal{L}} \alpha_k \cdot \mathbf{y}^{(k)}) = \sum_{k=1}^{\mathcal{L}} \alpha_k \cdot \mathbf{z}^{(k)}$, where $\mathbf{z}^{(k)}$ is the decryption of the ciphertext returned by the server in the kth iteration. It has been proved that the equation verifies correctness of the output with error probability at most 2^{-l}.

Following this first work, several new protocols have been proposed to improve its properties. Chen et al. [35] mention that the solution of [129] may leak several columns of $\mathbf{T}$ through different iterations, in violation of input privacy. The proposed solution is to protect $\mathbf{x}$ using matrix $\mathbf{D}_2$ and use a left multiplication matrix $\mathbf{D}_1$ and a random permutation π of rows for both $\mathbf{A}$ and $\mathbf{b}$. The matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are chosen to be diagonal for efficiency reasons. Hence, the task $\Phi = (\mathbf{A}, \mathbf{b})$ is transformed into $\Phi' = (\mathbf{A}', \mathbf{b}')$, where $\mathbf{A}' = \mathbf{D}_1\pi(\mathbf{A})\mathbf{D}_2$, $\mathbf{b}' = \mathbf{D}_1\pi(\mathbf{b})$, and $\pi(\cdot)$ is a random permutation of the rows of its input. The server computes the solution $\mathbf{y}$ using any SLE solver and sends it to the client. The client checks whether $\mathbf{A'y} = \mathbf{b'}$ holds and recovers $\mathbf{x}$ as $\mathbf{x} = \pi^{-1}(\mathbf{D}_2\mathbf{y})$. Neither HE nor interactions between the client and the server are involved in this protocol.

Using a similar protocol, Chen et al. [37] consider using sparse matrices to protect the original dense coefficient matrix $\mathbf{A}$. In this solution, given $\Phi = (\mathbf{A}, \mathbf{b})$, the client computes $\mathbf{T} = \mathbf{MAN}$ and $\mathbf{d} = \mathbf{M}(\mathbf{Ar} + \mathbf{b})$, where $\mathbf{M}$ and $\mathbf{N}$ are sparse matrices and $\mathbf{r}$ is a random blinding vector, and sends the task $\Phi' = (\mathbf{T}, \mathbf{d})$ to the server. The server solves the SLE and sends the solution $\mathbf{y}$ to the client, which the client consequently verifies as $\mathbf{Ty} = \mathbf{d}$. This protocol avoids client-server interaction and achieves deterministic verification. Building on the work of [37], Nie et al. [102] propose a scheme that works even when the outsourced SLE has no solution. In this case, the client can construct a linear programming problem to determine if the SLE has a single unique solution or not. To claim that the SLE has no solution, the server has to show that a positive optimal objective value exists in the corresponding auxiliary problem.

Salinas et al. [113] looked into the challenges for memory-constrained devices and developed a protocol for outsourcing solving an SLE while maintaining low computational and storage complexities. The authors noticed that an instance of an SLE problem $\mathbf{Ax} = \mathbf{b}$ can be transformed to an unconstrained quadratic program which minimizes $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A'x} - \mathbf{b'x}$ and hence can be efficiently solved by the conjugate gradient method (CGM). To be able to use CGM, the client needs to set $\mathbf{A'} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{b'} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$. Because computing $\mathbf{A'}$ has high complexity, the client outsources this multiplication. In particular, the client masks $\mathbf{A}$ as $\mathbf{A'} = \mathbf{A} + \mathbf{Z}$, where $\mathbf{Z}$ is a pseudorandom matrix computed as the outer product of some vectors and is proved to be computationally indistinguishable from a matrix with uniformly chosen random elements, and sends it to the server. Once the client recovers $\mathbf{A'}$ from the server's response, it masks $\mathbf{A'}$ using the same mechanism as above and they engage in multiple iterations of CGM on protected $\mathbf{A'}$. The server's work during each iteration consists of matrix-vector multiplications, while the client performs only vector multiplications. This work also puts forward the notion of external memory I/O cost, which is the number of external memory accesses when the data is too large to reside in local memory, and shows that its external memory I/O cost is lower than in previous schemes. Verification, however, is not discussed, and thus the solution might be vulnerable to server attacks described in [129]. Lastly, combining the transformations of [37] and [113] described in this section, Yu et al. [147] proposed an SLE outsourcing protocol that protects the position of zero elements in the coefficient matrix $\mathbf{A}$.

4.3.2 Brief Discussion. Computation at this level of the computational hierarchy primarily follow an iterative design. Most of the strategies for securely solving an SLE in the literature are transformation-based. The techniques are similar to those used for matrix operations. Privacy protection measures focus on the intermediate results in an iterative protocol between a client and a server as in [129]. In some schemes, precision of the final output cannot be controlled by the user and depends on the online SLE solver; this problem is eliminated by the technique in [35, 37]. Also, the solution given in [113] is particularly suited for extremely large tasks. Lastly, the scheme given in [102] studies verification of outsourced computation of SLE when no solution exists and utilizes linear programming for that task, which shows a tight relationship between SLE and computational tasks at the top level of the computational hierarchy.

4.4 Mathematical Optimization Tasks

Mathematical optimization is a computational task for finding a maximum or minimum value of an objective function given several constraints, represented by equalities and inequalities [96]. The model is known as linear programming (LP) when both the objective function and constraints are linear. Another typical model of optimization problems is quadratic programming (QP), where the objective is a quadratic function and the constraints are all linear functions over the decision variables. Mathematical optimization is widely used in a number of industries including transportation, energy, manufacturing, financial analysis, and others.

4.4.1 Linear Programming. Wang et al. [127] were the first to provide practical mechanisms for secure outsourcing of large-scale LP computation. That work defined a generalized form of linear programming which is:

$$\min \mathbf{c}^{\mathrm{T}}\mathbf{x} \text{ subject to } \mathbf{Ax} = \mathbf{b}, \mathbf{Bx} \geq \mathbf{0}$$

This formulation can be represented as a tuple $\Phi = (\mathbf{A}, \mathbf{B}, \mathbf{b}, \mathbf{c})$ composed of an $m \times n$ matrix $\mathbf{A}$, n-element vectors $\mathbf{c}$ and $\mathbf{b}$, and an $n \times n$ non-singular matrix $\mathbf{B}$. To protect output privacy, the solution applies a random affine transformation on the variable $\mathbf{x}$, defined by $\mathbf{y} = \mathbf{M}^{-1}(\mathbf{x} + \mathbf{r})$, where $\mathbf{M}$ is a random $n \times n$ non-singular matrix and $\mathbf{r}$ is an n-element vector. To protect input privacy, a randomly generated $m \times m$ non-singular matrix $\mathbf{Q}$ is used to hide the elements of $\mathbf{A}$ and $\mathbf{b}$. Furthermore, the inequality constraint $\mathbf{B}$ is protected by a randomly generated $n \times m$ matrix λ, which must satisfy $|\mathbf{B} - \lambda\mathbf{A}| \neq 0$ and $\lambda\mathbf{b} = \mathbf{0}$. To protect the objective function, a real-valued positive scalar γ is used to replace $\mathbf{c}$ by $\gamma\mathbf{c}$. Thus, the LP problem Φ is transformed into:

$$\Phi' = (\mathbf{QAM}, \mathbf{BM} - \lambda\mathbf{QAM}, \mathbf{Q}(\mathbf{b} + \mathbf{Ar}), \gamma\mathbf{c}^{\mathrm{T}}\mathbf{M})$$

by the client and is outsourced to the server. Applying the transformation to the entire problem enables the cloud server to use existing algorithms and tools for LP solvers, such as Simplex and Interior Point methods [49], to perform the task.

During verification, three different cases are considered according to the solution provided by the server: 1) There is an optimal solution with finite objective value. The server must provide an optimal solution $(\mathbf{s}, \mathbf{t})$ to the dual problem, and the client verifies whether the following conditions are satisfied:

$$\gamma\mathbf{c}^{\mathrm{T}}\mathbf{My} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar})\mathbf{s}, \mathbf{QAMy} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar}), \mathbf{BMy} - \lambda\mathbf{QAMy} \geq 0, \text{ and}$$
$$\mathbf{QAMs} + (\mathbf{BMy} - \lambda\mathbf{QAMy})\mathbf{t} = \gamma\mathbf{c}^{\mathrm{T}}\mathbf{M}.$$

2) The server claims that Φ' is infeasible. The server needs to demonstrate that a positive optimal objective value exists in the following auxiliary problem:

$$\text{Minimize } z \text{ subject to } -\mathbf{1}z \leq \mathbf{QAMy} - \mathbf{Q}(\mathbf{b} + \mathbf{Ar}) \leq \mathbf{1}z, \mathbf{BMy} - \lambda\mathbf{QAMy} \geq -\mathbf{1}z.$$

3) The server claims that the problem Φ' is unbounded. The server is required to show that the following problem has optimal objective value of 0:

$$\text{Minimize } z \text{ subject to } -\mathbf{1}z \leq \mathbf{QAMs} + (\mathbf{BM} - \lambda\mathbf{QAM})\mathbf{t} - \gamma\mathbf{c}^{\mathrm{T}}\mathbf{M} \leq \mathbf{1}z, \mathbf{t} \geq -\mathbf{1}z.$$

All of these tasks can be accomplished by the server using the information contained in Φ'. Because solving the auxiliary LP problem with optimal solutions is asymptotically the same as solving the original LP problem (which is $\Omega(n^3)$ for n variables), the verification process does not significantly increase the server's work. Wang et al. [128] formally show security of this scheme against ciphertext-only attacks. There is also analysis of security and efficiency tradeoffs between their design and FHE.

Using a similar setup, Nie et al. [101] provided a new secure outsourcing algorithm in order to lower the client's work. Compared to the solution given in [127], this work replaces random matrices $\mathbf{M}$ and $\mathbf{Q}$ with sparse matrices. The design follows a similar outsourcing and verification process. The client outsources two transformed tasks to the server and holds two separate keys associated with the transformations. The client's transformation and verification time is $O(n^2)$ because of matrix-vector multiplications. Chen et al. [35] reformulate the LP problem in a 'standard' form given in [26], where the constraint $\mathbf{Bx} \geq \mathbf{0}$ is replaced with $\mathbf{x} \geq \mathbf{0}$. That work also claimed that some of the linear transformations used in [127] are not necessary and that γ must be positive, as negative values (allowed in [127]) can lead to incorrect results.

4.4.2 Quadratic Programming. A QP problem can be represented in the following form:

$$\min \tfrac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{p}^{\mathrm{T}}\mathbf{x} \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{B}\mathbf{x} \leq \mathbf{c}$$

where $\mathbf{Q}$ is a positive definite $n \times n$ matrix, $\mathbf{A}$ and $\mathbf{B}$ are full rank matrices of size $m \times n$ and $k \times n$, respectively, $\mathbf{x}$ and $\mathbf{p}$ are n-element vectors, $\mathbf{b}$ is an m-element vector, and $\mathbf{c}$ is an k-element vector. Thus, an QP problem can be represented as $\Phi = (\mathbf{Q}, \mathbf{p}, \mathbf{A}, \mathbf{b}, \mathbf{B}, \mathbf{c})$. To securely outsource an QP problem, Zhou et al. [152] designed a protocol to transform Φ into a protected form and verify the result returned by the server through Karush-Kuhn-Tucker (KKT) conditions which are necessary and sufficient for the optimal solution. Similar to the solution in [127], this work uses transformation $\mathbf{x} = \mathbf{N}\mathbf{y} + \mathbf{r}$, where $\mathbf{N}$ is an $n \times n$ random matrix and $\mathbf{r}$ is a random vector of size n. This transformation is used for the feasible region because unlike [127] that covered all possibilities including infeasible and unbounded cases, this work makes an assumption that the input problem is feasible. Also, a randomly generated $m \times m$ matrix $\mathbf{M}$ is used to hide the equality constraints $(\mathbf{A}, \mathbf{b})$ and a $k \times k$ matrix $\mathbf{L}$ is used to hide the inequality constraints $(\mathbf{B}, \mathbf{c})$. Hence, the problem Φ is transformed into $\Phi' = (\mathbf{Q}', \mathbf{p}', \mathbf{A}', \mathbf{b}', \mathbf{B}', \mathbf{c}')$, where $\mathbf{A}' = \mathbf{MAN}$, $\mathbf{b}' = \mathbf{M}(\mathbf{b} - \mathbf{Ar})$, $\mathbf{B}' = \mathbf{LBN}$, $\mathbf{c}' = \mathbf{L}(\mathbf{c} - \mathbf{Br})$, $\mathbf{Q}' = \mathbf{N}^{\mathrm{T}}\mathbf{QN}$, and $\mathbf{p}' = (\mathbf{r}^{\mathrm{T}}\mathbf{QN} + \mathbf{p}^{\mathrm{T}}\mathbf{N})^{\mathrm{T}}$, and it is formulated as:

$$\min \tfrac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{Q}'\mathbf{y} + \mathbf{p}'^{\mathrm{T}}\mathbf{y} \text{ subject to } \mathbf{A}'\mathbf{y} = \mathbf{b}', \mathbf{B}'\mathbf{y} \leq \mathbf{c}'.$$

The objective function $(\mathbf{Q}, \mathbf{p})$ is transformed in $(\mathbf{Q}', \mathbf{p}')$, where $\mathbf{Q}' = \mathbf{N}^{\mathrm{T}}\mathbf{QN}$ retains the property of $\mathbf{Q}$ that it is positive definite. The server is instructed to solve the transformed QP problem along with its dual problem. To demonstrate correctness, the necessary and sufficient KKT condition for the optimal solution are formulated as follows: if $\mathbf{x}^*$ is the optimal solution, then there exist an m-element vector $\alpha^* \geq 0$ and an k-element vector $\beta^* \geq 0$ satisfying $\mathbf{Q}\mathbf{x}^* + \mathbf{p} + \mathbf{A}^{\mathrm{T}}\alpha^* + \mathbf{B}^{\mathrm{T}}\beta^* = 0$, $\mathbf{A}\mathbf{x}^* = \mathbf{b}$, and $\mathbf{B}\mathbf{x}^* \leq \mathbf{c}$. Once the result is returned by the server, the client verifies whether the KKT conditions hold. For a task with n variables, the client's overhead is $O(n^2)$, while it requires $\Omega(n^3)$ time to solve a QP problem directly.

Salinas et al. [114] treat the problem of QP outsourcing in a simpler form with only inequality constraints as in:

$$\min \tfrac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{x} \text{ subject to } \mathbf{A}\mathbf{x} \leq \mathbf{c}$$

The solution requires the client to transform the problem into its dual form: $\min g(\lambda) = \frac{1}{2}\lambda^{\mathrm{T}}\mathbf{P}\lambda + \lambda^{\mathrm{T}}\mathbf{r}$ subject to $\lambda \geq 0$, where $\mathbf{P} = \mathbf{AQ}^{-1}\mathbf{A}^{\mathrm{T}}$ is a positive definite and symmetric matrix, $\mathbf{r} = \mathbf{c} - \mathbf{AQ}^{-1}\mathbf{b}$, and λ is the vector of dual variables. Because $\mathbf{P}$ is expensive to compute, the client securely outsources matrix multiplications to the server, which allows the client to obtain $\mathbf{P}$ and $\mathbf{r}$. The client then transforms $\mathbf{P}$ into its protected form $\mathbf{P}' = \mathbf{EDR}_P\mathbf{P}(\mathbf{ED})^{-1}$, and $\mathbf{r}$ into $\mathbf{r}' = \mathbf{EDR}_P\mathbf{r}$, where $\mathbf{E}$ is pseudorandom orthogonal permutation matrix and $\mathbf{D}$ and $\mathbf{R}_P$ are diagonal matrices with randomly generated non-zero elements. Furthermore, the initial value of λ is concealed as $\lambda' = \mathbf{ED}\lambda_0$. The cloud server is instructed to carry out the (iterative) Gauss-Seidel algorithm on protected data, until the stopping criteria are met. Once the client receives the result, it verifies its correctness by checking the appropriate KKT condition. The paper shows that two transformations of the same problem are computationally indistinguishable from one another under a chosen-plaintext attack, but this does not exclude the possibility of revealing information about the original problem (such as the number of zero elements).

Another type of mathematical optimization, called Convex Separable Programming, is considered in [90]. Because this is a non-linear problem, the proposed solution was to first linearize the non-linear parts of the computation by inserting grid points. By doing so, the client can form a number of LP problems as a transformed version of the original problem. Using random matrices, the client can outsource these tasks securely to the cloud. The paper also reasons about the best tradeoff between accuracy and the time necessary for computation convergence.

4.4.3 Brief Discussion. The LP outsourcing solutions surveyed in this section all use random transformation as the main design idea and defend against malicious servers. They can be distinguished by the way the transformations are performed. For example, [127] uses a non-singular dense matrix $\mathbf{Q}$ to hide the equality constraints, a randomly generated matrix λ to hide the inequality constraints, and a non-singular dense matrix $\mathbf{M}$ to hide the feasible area of the problem. [35], on the other hand, uses a diagonal positive definite matrix to hide the feasible area. This saves client's matrix multiplications, but this scheme is only applicable when the inequality constraints are specified in the form $\mathbf{x} \geq \mathbf{0}$ instead of $\mathbf{Bx} \geq \mathbf{0}$. In the design of [101], sparse matrices are used to hide the constraints and a random vector is used to hide the final result. Matrix multiplications becomes cheap, but it is difficult to prove privacy protection of this scheme.

This concludes our survey of outsourcing common functions categorized in Figure 2. Additional analysis of their security properties and performance is available in Sections 6 and 7. Because the hierarchy of Figure 2 cannot cover all schemes available in the literature, we continue with other specialized computations in different domains in Section 5.

5 APPLICATION-ORIENTED SECURE COMPUTATION OUTSOURCING

In this section, we cover several research areas where secure computation outsourcing finds its applications. Compared to descriptions in Section 4, this section provides high-level overview rather than detailed coverage of related work due to distinctness of each problem domain and variations of the techniques.

5.1 Machine Learning and Data Mining Tasks

Machine learning is a field that enables computers to learn without being explicitly programmed. This covers techniques such as regression, support vector machines (SVM), association rule learning, and others, which we consequently discuss.

Different forms of regression are extensively used in practical applications as a model to predict a real-valued output using an input data set [16], which can be computationally intensive for large data sets. Nikolaenko et al. [103] developed a solution for privacy-preserving ridge regression where distributed data is contributed by many clients. Two servers are used to perform the computation on behalf of the clients, one of which is termed as the evaluator and the other as the crypto service provider (CSP), and the evaluator learns the output of the computation. The solution uses a combination of HE and garbled circuits. In the first phase of the computation, the users submit encrypted data and the evaluator leverages homomorphic properties of the encryption scheme and the evaluator performs the first part of the computation that uses only linear operations. In the second phase, with the help of the CSP, the evaluator performs the remaining portion of the computation with many non-linear operations. The CSP primarily performs offline work, but in the most efficient version it also needs to participate in one round of computation. The basic solution can be extended to deter misbehavior of both the evaluator and CSP.

A linear regression (LR) model is typically defined as $\mathbf{y} = \mathbf{X}\beta + \epsilon$, where $\mathbf{y}$ is a vector of size m, $\mathbf{X}$ is an $m \times n$ matrix, β is a vector of size n and ϵ is the error term. The goal is to compute coefficients β of the linear function while minimizing ϵ. When least-squared error estimation is used, the solution can be computed as $\beta = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$. Hence, the objective of LR outsourcing is to obtain β while protecting $\mathbf{X}$, $\mathbf{y}$, and β. Note that the significant difference between LR and SLE tasks is that with LR the final optimal result minimizes the squared error and does not necessarily satisfy all of the equations [143]. Chen et al. [34] designed two protocols for LR outsourcing for different application requirements. In the first protocol, the client transforms the original LR problem $\Phi = (\mathbf{X}, \mathbf{y})$ into $\Phi' = (\mathbf{AXD}, \mathbf{Ay})$, where $\mathbf{A}$ is a random orthogonal matrix of size $m \times m$ and $\mathbf{D}$ is a random diagonal matrix of size $n \times n$, and sends Φ' to the server, who solves the LR problem and obtains β'. After receiving the results, the client recovers β as $\beta = \mathbf{D}\beta'$. The client also checks whether each element in $\mathbf{y} - \mathbf{X}\beta$ is small enough prior to accepting the result. The client's cost of this protocol is $m^2n + 2mn + 2m$ if the Gram-Schmit algorithm is used to generate the orthogonal matrix. The second protocol modifies $\mathbf{A}$ to be a diagonal matrix, and $\mathbf{A}$ is randomly filled with values k or $-k$. This protocol requires the client to do $3mn + 2m$ work, which lowers the computation cost if m is large, but provides lower security guarantees. One of the two protocols can be selected based on how large m is compared to n.

Support vector machines (SVM) are supervised learning models and the associated algorithms for classification with state-of-the-art performance. Given a training data set, the problem is to find an optimal hyperplane for dividing the data set into two categories according to the kernel-defined feature space [46]. Given m training data instances $(\mathbf{x}_i, y_i)$, where each $\mathbf{x}_i$ is an n-element vector and y_i is either 1 or -1 indicating the class to which point $\mathbf{x}_i$ belongs, the output of the SVM is a decision function obtained via quadratic programming. An SVM problem is often solved in its dual form to achieve optimality. Besides the training phase, the most time consuming component is searching the cost and kernel parameters using a brute force search, especially when dealing with large training data sets.

To realize privacy-preserving outsourcing of SVM computation, Lin et al. [92] proposed to use random transformations during the training phase. However, this technique cannot preserve certain relationships among data points, such as the dot product and Euclidean distance. To address this issue, to solution is to introduce a reduced kernel matrix K, whose entries combine training points with random vectors, denoted as $K_{i,j} = k(\mathbf{x}_i, \mathbf{r}_j)$. The randomness is used to provide security protection of intermediate computations. The client (i.e., the data owner in this work) first transforms the training points using a random transformation as $\mathbf{c}_i = \mathbf{M}\mathbf{x}_i$, where $\mathbf{M}$ is a random nonsingular $n \times n$ matrix and $i = 1, \ldots, m$. In addition, the random vectors $\mathbf{r}_j$ are transformed as $\mathbf{s}_j = (\mathbf{M}^{\mathrm{T}})^{-1}\mathbf{r}_j$. This transformation is used to protect data privacy at the initial phase before the problem is formed. At this point the transformed training instances transformed random vectors are sent to the server for securely computing kernel matrices. The server can carry out parameter searching and model training over the transformed data. The transformation ensures that the generated classifier can only be known to the client. After receiving the trained SVM classifier, the client can outsource transformed query data to the server for classification decisions while achieving data privacy.

Association rule mining is a rule-based machine learning method for finding frequent patterns, correlations, associations and other useful relations among variables in large databases, as illustrated in [107]. Wong et al. [138] developed an outsourcing protocol that maps an original item to a number of possible values using a substitution cipher. This offer stronger security protection compared to one-to-one mapping of deterministic ciphers. However, verification is not supported in this scheme. A follow-up work [139] introduced an artificial itemset planting (AIP) technique for constructing an audit environment to add checkability to their previous work.

There are also publications on secure query processing over large data sets that reside with the server, which can be called an outsourced database model. Wong et al. [140] treated k-nearest neighbor (kNN) queries over outsourced encrypted databases. The work proposed an encryption scheme with special properties called asymmetric scalar-product-preserving encryption (ASPE). The solution requires that the database owner and a client who queries it share the ASPE secret key. There were several subsequent studies on this topic. For example, Yao et al. [146] studied

Table 4. An overview of surveyed machine learning and data mining results.

Task	Scheme	Threat model	Result verification	Security protection method	Client's work
Regression	[103]	Malicious	High probability	HE+garbled circuits	Fair
	[34]	Malicious	Deterministic	Random transformation	Low
SVM training	[92]	Honest-but-curious	None	Random transformation	Very low
Association rule mining	[138]	Honest-but-curious	None	Transaction transformation	Low
	[139]	None	Probabilistic	None	Low
kNN	[140]	Honest-but-curious	None	Splitting and artificial attributes	Low

secure nearest neighbor queries, where a client can perform NN queries with query privacy and database privacy. The solution leverages special partitions and the Voronoi diagram of the database. Elmehdwi et al. [57] provided a solution to secure kNN queries with accurate results secure under chosen-plaintext attacks. It assumed two non-colluding cloud servers, one server generates the key and the other server stores all uploaded (encrypted) data tuples. Query, pattern, and database privacy are guaranteed with lightweight client's overhead. Li et al. [89] consequently extend the single data owner setting to multiple data owners.

Pattern matching amounts to searching a text for all occurrences of a specific pattern. To provide efficient and secure pattern matching outsourcing, Zou et al. [151] builds a solution that uses a new privacy-preserving outsourced discrete Fourier transform (OFFT) protocol as a building block. This building block is used to realize secure outsourced polynomial evaluation, which is consequently used to design the final outsourced pattern matching protocol. Security is shown in the semi-honest model, where text privacy is achieved even if the server and the pattern owner collude and pattern privacy is achieved even if the server and the text owner collude. The solution has low computation and communication cost.

An overview of the publications surveyed in this subsection is given in Table 4. Efficiency of client's transformations and verification is listed as a design goal for most of these schemes. With the exception of the schemes for regression, these protocols usually achieve privacy protection in the presence of honest-but-curious adversaries and cannot detect incorrect results returned by the server. Some techniques such as those used in [34] and [92] are similar to the techniques for hiding client's data discussed in the previous section.

5.2 Image Processing Tasks

As mentioned earlier, large data sets may include different forms of content such as text, audio, images, etc. The number and size of such data sets is rapidly growing today. Outsourcing image, and more generally, media processing to the cloud is promising, but faces critical challenges of data privacy. Several researchers studied this area, and we describe a number of advances in this field.

Wang et al. [130] proposed an outsourced image recovery service, where reconstructing images from compressed samples is considered to be a time-consuming task. The image content should be protected when images are being stored and during the recovery process. In this construction, the data owner transforms compressed raw image samples and sends them to the server for storage.

Upon receiving a reconstruction request from a client, the server reconstructs the corresponding images from its samples. Weak clients are able to recover the image efficiently. The signal can always be accurately recovered from its compressed samples by solving the following LP problem:

$$\min \mathbf{1}^{\mathrm{T}} \cdot \mathbf{g} \text{ subject to } \mathbf{y} = \mathbf{F} \cdot \mathbf{g}, \mathbf{g} \geq 0$$

where $\mathbf{F}$ is an $m \times 2n$ coefficient matrix, $\mathbf{g}$ is an $2n$-element vector of real-valued variables and $\mathbf{y}$ is an m-element sample vector. During problem transformation, the solution $\mathbf{g}$ is protected as $\mathbf{g}'$, where $\mathbf{g} = \mathbf{Q}\mathbf{g}' - \mathbf{e}$, $\mathbf{Q}$ is a random $2n \times 2n$ invertible matrix, and $\mathbf{e}$ is a $2n$-element vector; $\mathbf{y}$ is protected as $\mathbf{y}' = \mathbf{P}(\mathbf{y} + \mathbf{F}\mathbf{e})$, where $\mathbf{P}$ is a random $m \times m$ invertible matrix; $\mathbf{F}$ is protected as $\mathbf{F}' = \mathbf{P}\mathbf{F}\mathbf{Q}$; and a permutation matrix π and a random $2n \times m$ matrix $\mathbf{M}$ are additionally used to further hide the inequality constraints. The client can recover the result by computing $\mathbf{g} = \mathbf{Q}\mathbf{g}' - \mathbf{e}$. The client's work is $O(n^{\rho})$, while the time for solving the LP problem is $O(n^3)$. A follow-up work [150] proposed a parallel outsourcing and reconstruction solution of lower cost.

Image feature detection is a popular computational task for social network providers because it can improve their clients' experience. Because of significant computational resources associated with image feature detection, there is a need for this task to be outsourced. To address this issue, Qin et al. [109] designed a practical privacy-preserving solution for outsourcing Scalar Invariant Feature Transform (SIFT). SIFT consists of two phases: scale-space extrema detection and feature descriptor generator. In this solution, the client owns private image data and uses three non-colluding servers, two of which play the role of generators and the remaining one plays the role of a comparer. The client first divides the image matrix $\mathbf{M}$ into two matrices $\mathbf{C}_1 = \mathbf{P}$ and $\mathbf{C}_2 = (\mathbf{M} - \mathbf{P}) \bmod q$, where $\mathbf{P}$ is a random matrix with positive entries less than q, and sends them together with several control parameters to the generators. Each generator performs convolution and subtraction operations on its matrix as prescribed by the algorithm to generate scale-space. Consecutive operations require comparisons, to be run by the comparer, but pixel values needs to be protected before the comparer can observe intermediate results. Thus, the generators divide the scale-space into cubes, each cube is protected using order-preserving encryption, and the cubes are consistently permuted by the generators. After the comparer computes the extrema, the corresponding information is sent to the generators who proceed with the second phase of the computation and generate feature descriptors. Finally, the result is sent to the client. This transformation-based technique exhibits better performance than HE-based scheme such as [81]. Qin et al. [110] provided a full security proof for confidentiality of pixel values and descriptive feature vectors used in that work. Wang et al. [132] provided an improved SIFT outsourcing solution that preserved additional features of the original SIFT. It used two servers who utilize garbled circuits for secure comparisons and resulted in a more efficient solution that prior results. Lastly, another follow-up work [82] improves the solution to preserve characteristics of the original SIFT (such as robustness to a number of image variations) and offers lower computational and communication cost compared to prior work.

Working on secure outsourcing of a different popular feature descriptor called Histogram of Oriented Gradients (HOG), Wang et al. [135] considered two different server settings. In the single-server setting, the solution uses SwHE with the packing single-instruction multiple-data (SIMD) technique. The client encrypts image I and outsources it to the server. The server exploits homomorphic properties of the encryption to build scale spaces and consequently the image descriptor using a modified computation carefully designed to be carried out on encrypted data. The client receives and recovers the feature descriptor using decryption. To improve performance of the single-server solution when the size of images is large, the two-server solution lets the client split the image into two random shares and uses secure batched comparison run by the servers without learning each others' inputs. This solution is faster for the client and has accuracy closer to the original HOG.

Speeded-up Robust Features (SURF) is another widely used feature extraction algorithm, which can be viewed as improved version of SIFT. Wang et al. [133] proposed a protocol that uses two non-colluding servers and, similar to other work, executes SURF computation on images split into random shares. The solution includes two new protocols for secure packed multiplication and comparison using SwHE and SIMD. The former enables the servers to perform computation such as the second order Gaussian derivative and Hessian's determinant, while the latter is primarily used for dominant orientation assignment computation of the points of interest. Client's work in this solution is significantly lower than in Bai et al.'s scheme [11], which uses HE to construct an interactive multi-round protocol for SURF outsourcing.

Among the publications on secure outsourcing of image feature detection, the design of [109] is one of the few that uses transformation-based techniques to protect privacy of the query image. Also, result verification is not considered in the papers surveyed in this subsection, and often more than one cloud server is used for outsourcing for efficiency reasons.

5.3 Biometric Computation

Biometric computations entail analysis of data related to human characteristics such as genomic data, fingerprints, retina and iris patterns, written signatures, and others. There are many algorithms for analyzing biometric data suitable for large-scale computation. In addition, computation with sensitive genomic data such as DNA sequences is a large research direction in bioinformatics.

Secure outsourcing of private sequence comparisons was studied in early work by Atallah and Li [9] which allowed two clients, who hold a private sequence of size m and n, respectively, to offload computation and communication burden associated with dynamic programming, which has cost $O(mn)$. Blanton et al. [19] later improved that work to allow the servers to compute the edit distance and the edit path using only $O(m+n)$ memory (as opposed to $O(mn)$ memory for edit path computation in [9]). The proposed solution is non-interactive for the clients, requires them to do only work linear in their input size, and uses garbled circuits in a new non-black-box way with two servers. Performance is substantially improved compared to that in [9]. Blanton and Aliasgari [17] proposed an outsourcing solution for error-resilient DNA searching via oblivious evaluation of a finite automaton. The protocol allows a client's private DNA sequence to be evaluated for the presence of a service provider's private pattern that corresponds to a certain genetic test.

Motivated by expensive computation required for all-pairs biometric comparisons [31], Blanton et al. [21] designed a mechanism for secure and verifiable computation outsourcing of generic biometric comparisons using several distance metrics (Hamming distance, Euclidean distance, and set intersection cardinality). In this work, a client with a large data set of pre-collected biometric samples outsources the computation of all-pairs comparisons and statistical analysis to multiple servers. The solution defends against a lazy adversary who may skip a portion of the computation, but does not intentionally corrupt the result. The protocol for biometric comparisons (i.e., distance computation) introduces a number of fake items inserted in random positions which are used as checkpoints and verified by the client. The server cannot distinguish fake items from real data because the data is protected. If the client additionally requests statistical information about the computed distances (in the form of a histogram), this information is computed using a randomized order of distances and different regions for real and fake item distances. The client performs a couple of checks on the returned histogram, and the solution guarantees that lazy servers cannot pass the verification with a larger probability than what the client sets.

To realize the exact computations used in biometric comparisons rather than standard distance metrics, a follow-up study [18] focused on the problem of secure outsourcing of iris identification. In this work, a database of encrypted iris codes is held by one or more servers, and a client with another iris code queries the database for matches. In the single-server model, privacy of both

Table 5. An overview of surveyed biometric computation literature.

Task	Scheme	Threat model	Result verification	Security protection methods	Local efficiency
DNA comparisons	[9]	Non-colluding servers	None	Splitting and HE	Slow
	[17]	Honest-but-curious	None	HE	Slow
	[19]	Honest-but-curious	None	Random transformation + garbled circuits	Fast
	[21]	Malicious	Probabilistic	Adding fake items	Very fast
Iris-based recognition	[18]	Malicious	Probabilistic	Secret sharing	Fast
	[131]	Honest-but-curious	None	Random transformation	Very fast

the database entries and client's queries is achieved using predicate encryption, but this solution suffers from inefficiency for large databases. The multi-server model relies on secure multi-party computation using secret sharing with at least three servers. Using a similar setting, the protocol given in [40] requires only two servers and can compute both Euclidean and Hamming distances. In their protocol, the entity's biometric database is encrypted using FHE, and the biometric image data is encrypted using additive HE.

The scheme of Yuan and Yu [148] was designed to enable privacy-preserving biometric identification for large databases, formulated as the same functionality as that of kNN outsourcing in [140], and claimed to offer stronger security protection than [140]. It was intended to sustain collusion of the server and a client who should be unable to learn information about the private database. However, it was later shown to be vulnerable to such collusions, in which case it becomes possible to remove the randomness [131]. To improve security properties of prior biometric identification protocols, Wang et al. [131] proposed a new efficient scheme, in which feature vectors are masked by multiplying them with random matrices. The solution is proved secure even when the server and the client collude.

A brief summary of the papers surveyed in this subsection is given in Table 5. Note that most of the schemes in this subsection rely on multiple servers. One exception is a single-server solution from [18], but its performance is not practical for a large database. An efficient single-server solution of biometric matching outsourcing was given in [131]. Lastly, the verification solution from [21] will work with one or more servers, as long as input biometrics can be properly protected. Similar to image processing outsourcing, efficient verification of biometric tasks is difficult and the available results are limited.

5.4 Graph Algorithms

Graphs are widely used in many different application domains, including social networks [100], online knowledge discovery [41], computer networks [122], and many others. In many cases, graphs are very large and represent sensitive data, as, for example, is the case for online social networks. In order to perform computation on large graphs efficiently and simultaneously address data privacy concerns, many publications have studied privacy-preserving querying of encrypted outsourced graphs.

Chase and Kamara [33] introduced the notion of structured encryption, where encrypted structured data can be privately queried using a specific token created using the secret key. Several query functions can be supported by this design, including vertex neighbor queries, vertex adjacency tests, and page ranking queries on a labeled graph. However, one of the most fundamental graph operations—the shortest distance between two vertices—is not supported. Hence, several publications have consequently studied privacy-preserving shortest path computation.

Blanton et al. [20] study the shortest path and several other classical graph algorithms, such as breadth-first search (BFS), depth-first search (DFS), minimum spanning tree (MST), etc. This work designed data-oblivious algorithms—defined as having data-independent memory accesses—for these problems so that they are suitable for secure execution in outsourced environments. A client first splits the graph, represented using an adjacency matrix **M**, and distributes it to the cloud servers. The rows of the graph are randomly permuted, and the developed algorithms proceed by revealing the working row of the matrix, without disclosing information about the node to which the row corresponds or information about graph connectivity. In the BFS algorithm, all nodes are marked as white, gray, or black as in the conventional algorithm and are obliviously and privately updated after processing one node (or one matrix row) using its adjacency information without explicitly maintaining a node queue as in the conventional algorithm. Upon computation completion, the servers return their shares of the result to the client. The authors also described other similar protocols for several graph problems including the single-source single-destination shortest path. Their solution first uses BFS to compute the distances from source to all other nodes in the graph and then iterates starting from the destination node to retrieve information about its parent. The length of the path can be protected by always performing the maximum number of iterations and ignoring all nodes once the source has been reached. Another work by Aly et al. [5] treated similar algorithms using secure multi-party computation techniques, but results in higher asymptotic complexities for the servers.

Following this line of work, Wang et al. [137] provided asymptotically better algorithms for special types of sparse graphs by integrating shortest path queries with oblivious data structures. However, these techniques are ORAM-based and require large bandwidth and expensive offline computations, which is generally not suitable for computation outsourcing. Wu et al. [141] presented a cryptographic design for the use of navigation on city streets, which is the most popular application of the shortest path problem. This work focuses on privacy of both the client's location and the server's routing database. The solution is based on private information retrieval and garbled circuit evaluation over a compressed graph, where the directions are represented by binary values during the preprocessing phase. Meng et al. [95] proposed a graph encryption scheme, which allows approximate shortest distance queries on large-scale graphs to be performed on encrypted data. The protocol, called GRECS, is provably secure against any semi-honest server. To achieve relatively efficient computation, the solution leverages symmetric-key operations. To lower communication overhead, this work also proposes another scheme which incorporate SwHE, sacrificing computational speed. To achieve both low computation and communication, they provide another protocol which has some additional leakage as a tradeoff. Recent work of Wang et al. [134] improves these results in two aspects: On the one hand, this work provides a mechanism for computing the exact shortest distance in the new Secure Graph DataBase encryption scheme (SecGDB) instead of estimating the length of the shortest path. The graph is represented using adjacency lists, and the solution combines additively HE and garbled circuits to implement Dijkstra's shortest path algorithm. On the other hand, this solution for the first time supports queries over dynamic graphs. To achieve this property, the solution uses another encrypted data structure populated with neighbor information of nodes in adjacency lists, which can support modifications homomorphically on ciphertexts. Moreover, the solution also maintains query history as an auxiliary data structure that stores previously queried results. As a result, it helps provide better amortized time complexity over multiple queries.

To summarize, graph-based secure computation outsourcing schemes can be divided into two categories: publications on securely outsourcing graph algorithms to the server (such as in [20]) and on executing protected queries over protected graphs stored on the server (such as in [134]). The goal of the publications in the first category is to protect the graph, including its structure, and the techniques include random permutation, adding fake nodes, and others. A refined data-oblivious algorithm is used to support queries. Publications in the second category are usually specific to shortest path queries. Computation often proceeds on Boolean values, while HE and garbled circuits provide sufficient privacy with relatively practical performance.

5.5 Cryptographic Tasks

Modern public key cryptography has pervasive applications. Devices such as RFID tags are computationally incapable of carrying out expensive cryptographic operations. Hohenberger and Lysyanskaya [79] were the first to study the question of securely outsourcing modular exponentiation, which is known to be a computational bottleneck in many discrete-logarithm-based cryptographic systems. The algorithm reveals only the size of the input to the servers and requires $O(\log^2 n)$ client's work (i.e., a constant number of modulo multiplications) for an n-bit exponent. Chen et al. [38] later proposed a new efficient and verifiable algorithm for outsourcing modular exponentiations that has advantages over prior work.

Garbled circuits are one of the most common techniques for secure function evaluation, but they require significant resources because garbling and evaluation of a single Boolean gate involves cryptographic operations. To lower the cost of circuit evaluation for devices with limited resources, Carter et al. [32] introduced a scheme for outsourcing this task to servers who are assumed to be malicious. Their protocol maintains input and output privacy for both participants, i.e., a weak evaluator who outsources its work and a more computationally capable garbler who can carry out its task, while significantly reducing computation and communication requirements for the weak client. Kerschbaum [83] consequently designed a scheme to additionally allow circuit generation to be outsourced under the assumption that the generator might also be computationally weak.

6 SECURITY ANALYSIS, COMPARISON AND DISCUSSION

6.1 System Model Variants

Before we discuss security treatment of secure computation outsourcing in the literature, we briefly summarize and categorize existing solutions of large-scale secure computation outsourcing by their system architecture. In addition to the common system model already described in Subsection 2.1, for the ease of presentation and fairness of the comparison we briefly introduce three specific variations, which are: 1) a system with three entities including a data owner, a service provider and a data user; 2) a system with two or more cloud servers executing a computation task outsourced by a client with no interaction between the two servers; and 3) a system with two or more cloud servers that collaboratively compute a task outsourced by a client. The cloud servers in 2) and 3) are usually assumed to be non-colluding. Although slightly different, the architectures still align well with the secure computation outsourcing model.

Schemes of the first type are usually presented in publications on outsourcing machine learning or bioinformatics tasks. The interaction often follows the data as a service (DaaS) model, where a data user requests the result of a certain computational task over a certain data set. The data owner is an entity who has complete control over the data and can authorize or deny access to portions of it. The data should be protected from both external service providers and in part from data users. In this system model type, the verification phase sometimes is not even applicable. Although data owner and data user should be considered separately when considering privacy protection in some schemes, these two entities can be viewed as a single entity, or client, in certain outsourcing scenarios such as [92]. This model is almost the same as the original system model of Subsection 2.1 and our analysis of security-related properties such as input/output privacy, verification, etc. below also applies to these schemes.

Solutions of the second type, with two (or more) non-interacting cloud servers are usually found in publications on outsourcing cryptographic tasks. For example, earlier studies on outsourcing modular exponentiations assumed two non-colluding cloud servers to which a client outsources tasks on two correlated inputs without letting each of the servers know the original input. The outputs of the servers are used to verify each other's computation. These schemes are still in the scope of computation outsourcing because a heavy computation task is alleviated on the client side, which is the main difference between secure computation outsourcing and secure multi-party computation.

The third variant of the system model also involves two or more cloud servers which are now required to interact in the protocols. This model is widely used in studies of outsourcing multimedia-related tasks such as image feature extraction and also in outsourcing graph-based computations. A common strategy to outsourcing a task in this case is to split the original data into random matrices locally and distribute them to separate servers. The servers then compute the function on their own private input and interact to communicate their outputs. Additional processing in the form of comparison, aggregation or integration is securely conducted by an additional server or the client.

6.2 Input and Output Privacy Protection

Recall from Section 2.3 that a fundamental security property sought of secure computation outsourcing schemes is that of privacy protection of the data handled by cloud servers. This was formulated as *input privacy*, i.e., inability of a cloud server to derive information about the input data that it receives (in a protected form) and uses in the computation and also *output privacy* that specifically refers to the server's inability to learn information about the result of the computation. To meet this security objective, ideally the server should be unable to learn any information about the data it receives or computes. This is formally modeled as the server's inability to distinguish outsourced data from randomly generated data of the same size using statistical or computational notion of indistinguishability. This general formulation is applicable to constructions with a single server or multiple servers, including the setting where the servers communicate.

Many constructions in the literature meet this definition of data privacy. Examples include [97] and [149]. There are, however, other publications that relax this definition of data privacy and allow some data leakage about private data in order to improve performance of their constructions. In this section, we outline common formulations of data privacy and attacks on data privacy in the context of computation outsourcing.

As stated in Section 2, local data transformation or encryption prior to outsourcing is a necessary step for achieving input/output privacy. Because constructions based on cryptographic techniques and those based on custom transformations to fit the needs of specific problems often have significantly different characteristics, in what follows, we analyze these two categories of techniques separately. Furthermore, because FHE can be used for general computational tasks with strong privacy guarantees, we will only discuss approaches that utilize partially HE.

We start with a rough classification of the publications based on the techniques they employ which is given in Table 6. Transformation-based techniques usually use a key in the form of random invertible matrices or random numbers to hide the original dataset. Early publications such as [10] deems it impractical to provide a systematic analysis of data privacy of transformation-based

Table 6. Comparison of two different outsourcing strategies.

Technique	Representative work	Advantages	Disadvantages
Transformation-based	[127] [34]	More efficient	Problem-specific, less security
Transformation-based + partially HE	[129] [149]	More secure and general	Inefficient

techniques because data disguises are problem-dependent. Furthermore, early attacks against privacy of outsourced data, such as the *statistical attack* of [10] which traverses the output of specific random generators to analyze the transformed data, are considered weak and can be defeated using complex probability distributions or by refreshing the key.

More recently, it became common to see the data privacy property formulated as security under a *ciphertext-only attack* with certain constraints. Note that because of the non-interactive nature of computation outsourcing, it is meaningful to formulate data privacy using encryption terminology. This property requires that, given encrypted input, it is infeasible for the server to obtain or deduce any information about the original data. In works that adopt this definition, it is either shown that the distribution of the transformed data is statistically close to that of a data sampled uniformly at randomly, or the server's advantage in distinguishing the transformed data from random values is negligible. An example of constraints that may come with this formulation of security can be found in the work of [129]. Its focus is on iterative linear equation outsourcing, and input privacy can be achieved if the number of iterations is limited by a certain value.

Taking the transformation technique of [127] as an example of constructions secure against a ciphertext-only attack, we can see that construction adds randomly chosen vector $\mathbf{r}$ to input $\mathbf{x}$ as a masking layer. This renders different forms of analysis attacks ineffective when only a ciphertext is available to the attacker. It has been later shown in [128] that if the entries of $\mathbf{r}$ are uniformly chosen from interval $\mathbf{I} = [-2^{\kappa}, 2^{\kappa}]$, where κ is a security parameter, the statistical distance between $\mathbf{x} + \mathbf{r}$ and a vector $\hat{\mathbf{r}}$ randomly and uniformly sampled from $\mathbf{I}$ is a negligible function. In other words, the server's views $\mathbf{x} + \mathbf{r}$ and $\hat{\mathbf{r}}$ are statistically indistinguishable. Hence, protection of individual sensitive matrix and vector values (e.g., $\mathbf{b}$ and $\mathbf{b}' = \mathbf{Q}(\mathbf{b} + \mathbf{x}\mathbf{r})$) is achieved.

The above analysis is based on the assumption of that a random transformation is used only once. Later it has been observed that using the same random transformation multiple times may allow the attacker to accumulate specific plaintext/ciphertext pairs until the original matrix could be recovered by solving a linear equation or certain information about original matrices can be learned. For example, several schemes lead to data leakage if users use a single matrix $\mathbf{M}$ to hide their data vectors $\mathbf{x}$ as $\mathbf{M}\mathbf{x}$. If a client applies the same transformation matrix $\mathbf{M}$ as the secret key to hide another data vector $\mathbf{y}$, the server can recover information about the vector in the form of $\mathbf{y}^{-1}\mathbf{x}$ by multiplying $(\mathbf{M}\mathbf{y})^{-1}$ with $\mathbf{M}\mathbf{x}$. Therefore, under the assumption that the same transformation key can be used to transform different inputs, it becomes meaningful to consider security under a *known-plaintext attack*. Given knowledge of the plaintext, the objective of the attacker then becomes to recover the secret key. It can be easily shown that many schemes from the literature become vulnerable in this security model. This is often because they apply addition and/or multiplication of random matrices to protect the original data matrix and these operations are distributive.

The above definitions of ciphertext-only and known-plaintext attacks capture the notion of no information about the input being revealed to the server. A number of publications relax these definitions and allow the server to learn certain information about the data. For example, transformations that multiply input by random matrices can reveal the number of zero elements in the original data or even the location of zero elements in the input matrix, which is proposed in [147]. There are also solutions that preserve certain properties of the input matrix after the transformation, such as retaining the matrix sign or rank, which also amounts to information leakage. Note that the property of rank-preserving is often necessary when encrypting the coefficient matrices in linear systems and linear programming. These constructions would not satisfy the requirements of security of indistinguishability but is still secure against ciphertext-only attack.

To better preserve input privacy, several studies apply encryption schemes with limited homomorphic properties. For example, [97] design computation over ciphertexts encrypted using different HE techniques to outsource matrix multiplication. Because the server receives multiple transmissions from the client, data privacy is defined with respect to all the messages the server sees during the protocol, as we originally define using the notion of indistinguishability. Other publications such as privacy-preserving sequence comparisons in [9] and ridge regression in [103] utilize multiple servers and would fall under the same formulation of data privacy as above. Similar levels of privacy protection can be found in [149], and many others. The properties of constructions for outsourcing fundamental functions and their detailed comparison can be found in Table 7. Table 9 explains some abbreviations and notation used in Table 7.

Most of the outsourcing schemes we have discussed also aim to protect output privacy. Because the transformation process is often symmetric, output privacy in these schemes is usually achieved at the same security strength as that of input privacy. In contrast, there are also a number of existing schemes that only protect input privacy or provide a limited form of output protection. For example, information about output may be exposed to a cloud server in the form of revealing records that matched the client's query. This is present in outsourcing biometric-related and data mining related tasks, such as searching a set of iris codes for a match [18] and searching the k-nearest neighbors over the outsourced database [57].

As a brief summary, we observe the following trends: 1) For transformation-based schemes to achieve the security guarantees put forward in the respective publications, the masking material in the form of vectors and matrices should be randomly chosen anew for each input. 2) As encryption aids privacy protection, it is easier to achieve input privacy if semantically secure encryption is used. 3) When multiple servers are utilized, input privacy cannot be guaranteed if they collude.

6.3 Ensuring Correctness through Verification

Checkability, or verifiability, is one of the key requirements of secure computation outsourcing. As previously described, it refers to the user's ability to detect server misbehavior. Current designs can be divided into two cases: the ones with deterministic verification and with non-deterministic verification.

The designs with *deterministic verification* can always provide deterministic checkability. One example can be found in [127] which deals with linear programming computation. Compared to many outsourced tasks that correspond to fundamental functions whose output falls in a single case, the LP problem may not have an optimal solution. This raises the challenge of designing the verification procedure. Besides verifying the returned optimal solution, the user has to ensure that the 'infeasible' or 'unbounded' output is honestly reported by the cloud server. Hence, the server is instructed to return a proof Γ that includes different options for different cases. In the case of feasible LP, a dual optimal solution should be included in the proof. Then the user can validate correctness of the solution if both the primary and dual objective values are equal. In the case that the server wants to show infeasibility of the original task, it has to solve the auxiliary LP problem. Then both the optimal solution of the auxiliary problem and its proof of optimality should be included in the proof Γ. The unbounded case is treated similarly.

Table 7. Protocol design choices in outsourcing of fundamental functions.

Task	Scheme	Threat model	Technique	Interaction	Verification dependence	Security strength
Matrix multiplication	[14]	Malicious	TF + HE	Once	p	COA(NS)
	[8]	Malicious	SS	Vrf	t	IND
	[86]	Malicious	TF	No	l	COA(NS)
	[97]	Malicious	TF + HE	No	Deterministic	IND
	[149]	Malicious	TF + PRF + HE	No	Deterministic	IND
		Semi-honest	TF + PRF + HE	No	Deterministic	IND
	[59]	Malicious	PRFC	No	Deterministic	IND
Matrix inversion	[88]	Malicious	TRF	No	l	COA(NS)
Nonnegative matrix factorization	[55]	Malicious	TF	Vrf	Deterministic	COA(NS)
Matrix eigen-decomposition	[153]	Malicious	TF	No	l	COA(NS)
Singular value decomposition	[153]	Malicious	TF	No	l	COA(NS)
Matrix determinant	[87]	Malicious	TF	No	l	COA(NS)
System of Linear Equations	[129]	Semi-honest	TF + HE	Solv, Vrf	Deterministic	COA
		Malicious	TF + HE	Solv	l	COA
	[35]	Malicious	TF	No	Deterministic	COA(NS)
	[37]	Malicious	TF	No	Deterministic	COA(NS)
Linear programming	[127]	Malicious	TF	No	Deterministic	COA
	[101]	Malicious	TF	No	Deterministic	COA(NS)
Quadratic programming	[114]	Malicious	TF	Yes	Deterministic	IND
	[152]	Malicious	TF	No	Deterministic	COA(NS)

The design with *non-deterministic verification* can guarantee correctness with certain probability. The probability of a client accepting a wrong result can be adjusted using security parameters to balance performance and the probability of error. One typical non-deterministic verification example is the design of [129], where computation of the solution to a linear equation proceeds in iterations until the convergence criterion is satisfied. In its verification phase, to check correctness of $\mathcal{L}$ iterations in a batch, the client randomly selects $\mathcal{L}$ l-bit numbers to be used as the coefficients in a linear combination. Correctness of the received answer in $\mathcal{L}$ iterations can then be tested by checking whether the linear combination of the returned vectors with the specified random coefficients equals to the value that the client expects. The equality always holds when the output in each iteration is correctly computed. It can also be proved that a wrong result even with only one incorrect value in the received vector will be undetected with probability less than 2^{-l}.

The design of a verification process is highly related to efficiency. A poor verification mechanism can be costly to the client and result in computation comparable to executing the task itself. Hence, a careful design of verification is quite needed and was developed in many outsourcing schemes as we discussed. Although non-deterministic verification shows its weakness on the accuracy of misbehavior detection compared to deterministic approaches, it can be more flexible in practice when efficiency is the most important metric. Furthermore, non-deterministic verification often provides sufficient guarantees in the long run when the dataset is large enough.

7 PERFORMANCE EVALUATION

For a given computational task, client's performance can be evaluated by comparing the encryption/decryption or transformation overhead as well as task verification cost to computation overhead of performing the original task. Because of the original motivation of outsourcing large-scale computation, performance speed-up is a necessary requirement for outsourcing schemes. Thus, Table 8 summarizes performance speed-up for different schemes, except those that use FHE, whose encryption overhead can be very high (notation can be found in Table 9). For example, in [88] and related schemes, the transformation uses a random matrix for hiding the original data. Client's computation is dominated by several matrix additions and matrix-vector multiplications, which take $O(n^2)$ time. As described in Subsection 6.1, HE is another common data protection technique used in secure computation outsourcing constructions. For example, it can be found in the design of [129], where a client performs a matrix-vector multiplication before encrypting the elements of the original matrix. If the number of computation iterations is less than the data size, the client's work is $O(n^2)$ including $O(n^2)$ encryptions.

To make a fair comparison, another issue that should be taken into consideration is the external memory I/O operations when the data is large and cannot reside in local memory, which was first discussed in [113]. In [88] and similar schemes, the cost of encryption is $4n^2$ I/O memory operations, decryption requires another $2n^2$ memory I/O operations, while the verification procedure needs another n^2 memory I/O operations. Hence, the memory I/O usage of this design is around $7n^2$ in total. In the scheme of [129], if all interactive operations occur within memory, the memory I/O usage is also around $7n^2$. For comparison, the scheme in [113] for outsourcing the same task uses vector operations on the client side instead of matrix-vector operations, and as a result achieves better memory I/O usage requiring $4n^2$ memory I/O operations, without considering verification.

8 OPEN ISSUES AND CHALLENGES

Although extensive research has already solved many challenges in secure outsourcing of large-scale computational tasks, a number of interesting and key research problems remain to be fully explored. We present several open problems below to stimulate further research:

Adaptive and flexible encryption design. As this survey demonstrates, there is a large number of schemes for outsourcing fundamental functions which can be used by many applications in engineering and other disciplines. Currently available outsourcing solutions mostly focus on one or two encryption (or transformation) approaches for a given computational task. The security parameter is often not explicitly given and the level of security is not tunable, but rather design choices were made to provide a tradeoff between efficiency and security. Engineering tasks, however, may have diverse requirements with some of them emphasizing fast computation over security, while others favoring stronger security guarantees. Thus, designs that allow for adaptive selection of security parameters and models with formalized analysis may be an interesting direction to explore.

Table 8. Efficiency comparison of schemes for outsourcing fundamental functions.

Task	Scheme	Enc/Dec time	Task time	Verification time
Matrix multiplication	[14]	$O(n^2)$	$O(n^\rho)$	$O(n^2)$
	[8]	$O(t^2n^2)$	$O(tn^\rho)$	$O(n^2)$
	[86]	$O(mn + ns + ms)$	$O(msn)$	$O(msl)$
	[97]	$O(n^2)$	$O(n^\rho)$	$O(n^2)$
	[149]	$O(mn + ns + ms)$	$O(msn)$	$O(ms)$
		$O(mn + ns + ms)$	$O(msn)$	$O(1)$
	[59]	$O(\max(m, n)s)$	$O(mns)$	$O(\max(m, n)s)$
Matrix inversion	[88]	$O(n^2)$	$O(n^\rho)$	$O(ln^2)$
Nonnegative matrix factorization	[55]	$O(\max(m, n)^2)$	$O(imnr)$	$O(mnr)$
Matrix eigen-decomposition	[153]	$O(n^2)$	$O(ln^2)$	$\Omega(n^3)$
Singular value decomposition	[153]	$O(n^2)$	$O(ln^2)$	$O(n^3)$
Matrix determinant	[87]	$O(n^2)$	$O(n^\rho)$	$O(ln^2)$
System of linear equations	[129]	$O(in + n^2)$	$O(n^\rho)$	$O(n^2)$
		$O(in + n^2)$		$O(ln^2)$
	[35]	$O(n^2)$		$O(n^2)$
	[37]	$O(\lambda n^2)$		$O(n^2)$
Linear programming	[127]	$O(n^\rho)$	$\Omega(n^3)$	$O(n^2)$
	[101]	$O(\lambda n^2)$		$O(n^2)$
Quadratic programming	[114]	$O(\max(mn, n^2))$	$\Omega(n^3)$	$O(\max(mn, n^2))$
	[152]	$O(n^2)$		$O(n^2)$

Parallel computations. In some of the existing schemes, encryption of different portions of the original data is not correlated. Hence, the encryption can be carried out in parallel to reduce client's computation time. This topic is rarely discussed in existing literature. Additionally, designing a parallel computing algorithm may open new interesting optimization possibilities and computational savings on the cloud side. Future solutions should take this design factor into consideration, with the goal of achieving larger time savings for the client.

Secure computation outsourcing with dynamic data. Dynamic data analysis is becoming more and more popular in modern data mining research, such as social network involvement and Twitter keywords tracking. Besides transforming or encrypting a computational task in its entirety, there is a need to be able to handle streaming or quickly changing data, where the speed of the response may be prioritized over its precision. Outsourcing schemes are often necessary when a fast real-time response is required, such as for traffic monitoring or route planning. Security and efficiency properties may need to be revisited for such dynamic environments.

9 CONCLUSIONS

In this survey, we give a systematic overview of existing solutions for securely outsourcing large-scale computations, including fundamental functions such as scalar product and matrix operations, as well as computation for specific applications such as optimization problems, data mining, graph algorithms, etc. Efficiency of client's computation and proper data confidentiality protection from

Table 9. Abbreviations and notation used in Tables 7 and 8.

Abbreviations		Notation	
CPA	chosen-plaintext attack	n	dimensions of a square input matrix or the number of columns in the first non-square matrix
COA	ciphertext-only attack		
IND	computationally indistinguishable	m	the number of rows in the first non-square matrix
		s	the number of rows in the second non-square matrix
Vrf	verification	t	secret sharing threshold
TF	transformation-based	ρ	the power in the asymptotic complexity of matrix multiplication
HE	homomorphic encryption	l	the number of iterations in the verification process
PRF	pseudorandom function	p	modulus size used in homomorphic encryption
PRFC	PRF with closed form efficiency	r	dimension parameter for matrix factorization
SS	Shamir's secret sharing	i	the number of iterations needed in the computation
Solv	solving computational task	λ	the upper bound on the number of non-zero elements in each matrix row
NS	not provably secure	p	the number of rows in the constraint matrix for optimization problems

the cloud server conducting the task are the two most important goals that prominent schemes from the literature aim to achieve. Additionally, verifiability of the computed result becomes an essential property for state-of-the-art secure computation outsourcing solutions in the presence of servers who are not fully trusted. We also identified tradeoffs between security and efficiency among different application domains. Following the literature review, we also comment on a number of open problems in the hope that this could help to shape future research directions in the area of securely outsourcing large-scale computations.

ACKNOWLEDGMENTS

This work was supported in part by grants CNS-1262277 and CNS-1319090 from the U.S. National Science Foundation and in part by Research Grants Council of HK (Project No. CityU C1008-16G). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] A. Adshead. 2014. Data set to grow 10-fold by 2020 as internet of things takes off. http://www.computerweekly.com/news/2240217788

[2] E. Aguiar, Y. Zhang, and M. Blanton. 2014. An Overview of Issues and Recent Developments in Cloud Computing and Storage Security. In *High Performance Cloud Auditing and Applications*, B.-Y. Choi, K. Han, and S. Song (Eds.). Springer.

[3] Jacob Alperin-Sheriff and Chris Peikert. 2013. Practical bootstrapping in quasilinear time. In *Advances in Cryptology–CRYPTO 2013*. Springer, 1–20.

[4] Jacob Alperin-Sheriff and Chris Peikert. 2014. Faster bootstrapping with polynomial error. In *CRYPTO*. Springer, 297–314.

[5] A. Aly, E. Cuvelier, S. Mawet, and M. Pereira, O.and Van Vyve. 2013. Securely solving simple combinatorial graph problems. In *International Conference on Financial Cryptography and Data Security*. 239–257.

[6] Amazon. 2017. What is Cloud Computing? https://aws.amazon.com/what-is-cloud-computing/

[7] A. Amritkar, E. de Sturler, K. Świrydowicz, D. Tafti, and K. Ahuja. 2015. Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver. *J. Comput. Phys.* 303 (2015), 222–237.
[8] M. Atallah and K. Frikken. 2010. Securely outsourcing linear algebra computations. In *ASIACCS*. ACM.
[9] M. Atallah and J. Li. 2005. Secure outsourcing of sequence comparisons. *IJISP* 4, 4 (2005), 277–287.
[10] M. Atallah, K. Pantazopoulos, J. Rice, and E. Spafford. 2002. Secure outsourcing of scientific computations. *Advances in Computers* 54 (2002), 215–272.
[11] Y. Bai, L. Zhuo, B. Cheng, and Y. Peng. 2014. Surf feature extraction in encrypted domain. In *ICME*. IEEE.
[12] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. 2013. Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 478–492.
[13] M. Bellare, V. T. Hoang, and P. Rogaway. 2012. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security*. ACM, 784–796.
[14] D. Benjamin and M. Atallah. 2008. Private and cheating-free outsourcing of algebraic computations. In *Privacy, Security and Trust, 2008*. IEEE, 240–245.
[15] D. Bernstein. 2005. *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*. Vol. 41. Princeton University Press Princeton.
[16] C. Bishop. 2006. Pattern recognition. *Machine Learning* 128 (2006), 1–58.
[17] M. Blanton and M. Aliasgari. 2010. Secure outsourcing of DNA searching via finite automata. In *IFIP Annual Conference on Data and Applications Security and Privacy*. 49–64.
[18] M. Blanton and M. Aliasgari. 2012. Secure outsourced computation of iris matching. *JCS* 20, 2-3 (2012).
[19] M. Blanton, M. Atallah, K. Frikken, and Q. Malluhi. 2012. Secure and efficient outsourcing of sequence comparisons. In *European Symposium on Research in Computer Security*. 505–522.
[20] M. Blanton, A. Steele, and M. Alisagari. 2013. Data-oblivious graph algorithms for secure computation and outsourcing. In *ASIA CCS*. ACM, 207–218.
[21] M. Blanton, Y. Zhang, and K. Frikken. 2013. Secure and verifiable outsourcing of large-scale biometric computations. *ACM Transactions on Information and System Security (TISSEC)* 16, 3 (2013), 11.
[22] Bluewaters. 2013. Blue Waters history. https://bluewaters.ncsa.illinois.edu/blue-waters
[23] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. 2014. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*. 533–556.
[24] G. Bonnoron, C. Fontaine, G. Gogniat, V. Herbert, V. Lapôtre, V. Migliore, and A. Roux-Langlois. 2017. Somewhat/Fully Homomorphic Encryption: Implementation Progresses and Challenges. In *C2SI*. Springer, 68–82.
[25] J. Bos, K. Lauter, J. Loftus, and M. Naehrig. 2013. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme.. In *IMA Int. Conf.* 45–64.
[26] S. Boyd and L. Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[27] Z. Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*.
[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *The 3rd Innovations in Theoretical Computer Science Conference*. ACM, 309–325.
[29] Z. Brakerski and V. Vaikuntanathan. 2011. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *FOCS 2011, IEEE.*
[30] Z. Brakerski and V. Vaikuntanathan. 2014. Lattice-based FHE as secure as PKE. In *ITCS*. ACM, 1–12.
[31] H. Bui, M. Kelly, C. Lyon, M. Pasquier, D. Thomas, P. Flynn, and D. Thain. 2009. Experience with BXGrid: a data repository and computing grid for biometrics research. *Cluster Computing* 12, 4 (2009), 373–386.
[32] H. Carter, B. Mood, P. Traynor, and K. Butler. 2016. Secure outsourced garbled circuit evaluation for mobile devices. *Journal of Computer Security* 24, 2 (2016), 137–180.
[33] M. Chase and S. Kamara. 2010. Structured encryption and controlled disclosure. In *Asiacrypt*. 577–594.
[34] F. Chen, T. Xiang, X. Lei, and J. Chen. 2014. Highly efficient linear regression outsourcing to a cloud. *IEEE Transactions on Cloud Computing* 2, 4 (2014), 499–508.
[35] F. Chen, T. Xiang, and Y. Yang. 2014. Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud. *J. Parallel and Distrib. Comput.* 74, 3 (2014), 2141–2151.
[36] X. Chen. 2016. Introduction to secure outsourcing computation. *Synthesis Lectures on Information Security, Privacy, & Trust* 8, 2 (2016), 1–93.
[37] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. Wong. 2015. New algorithms for secure outsourcing of large-scale systems of linear equations. *Transactions on Information Forensics and Security* 10, 1 (2015).
[38] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou. 2014. New algorithms for secure outsourcing of modular exponentiations. *IEEE Transactions on Parallel and Distributed Systems* 25, 9 (2014), 2386–2396.
[39] J. Cheon, J. Coron, J. Kim, M. Lee, T. Lepoint, M. Tibouchi, and A. Yun. 2013. Batch fully homomorphic encryption over the integers. In *EUROCRYPT*. 315–335.

[40] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang. 2014. Outsourceable two-party privacy-preserving biometric authentication. In *The 9th ASIACCS*. ACM, 401–412.

[41] D. Cook, L. Holder, G. Galal, and R. Maglothin. 2001. Approaches to parallel graph-based knowledge discovery. *J. Parallel and Distrib. Comput.* 61, 3 (2001), 427–446.

[42] T. Cormen. 2009. *Introduction to algorithms.* MIT press.

[43] J. Coron, T. Lepoint, and M. Tibouchi. 2014. Scale-invariant fully homomorphic encryption over the integers. In *International Workshop on Public Key Cryptography.* 311–328.

[44] J. Coron, A. Mandal, D. Naccache, and M. Tibouchi. 2011. Fully homomorphic encryption over the integers with shorter public keys. In *Annual Cryptology Conference.* 487–504.

[45] J. Coron, D. Naccache, and M. Tibouchi. 2012. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT.* 446–464.

[46] C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[47] C. Cullen. 2012. *Matrices and linear transformations.* Courier Corporation.

[48] W. Dai, Y. Doröz, and B. Sunar. 2014. Accelerating NTRU based homomorphic encryption using GPUs. In *HPEC.* IEEE.

[49] G. Dantzig and M. Thapa. 2006. *Linear programming 2: theory and extensions.*

[50] D. Demirel, L. Schabhüser, and J. Buchmann. 2017. *Privately and Publicly Verifiable Computing Techniques-A Survey.* Springer. 1–58 pages.

[51] Y. Doröz, Y. Hu, and B. Sunar. 2014. Homomorphic AES Evaluation using NTRU. IACR Cryptology ePrint Archive 2014/039.

[52] W. Du and M. Atallah. 2001. Privacy-preserving cooperative statistical analysis. In *ACSAC.* IEEE, 102–110.

[53] W. Du, M. Murugesan, and J. Jia. 2010. Uncheatable grid computing. In *Algorithms and theory of computation handbook.* Chapman & Hall/CRC, 30–30.

[54] W. Du and Z. Zhan. 2002. Building decision tree classifier on private data. In *The IEEE international conference on Privacy, security and data mining-Volume 14.* Australian Computer Society, Inc., 1–8.

[55] J. Duan, J. Zhou, and Y. Li. 2016. Secure and Verifiable Outsourcing of Nonnegative Matrix Factorization (NMF). In *The 4th ACM Workshop on Information Hiding and Multimedia Security.* ACM, 63–68.

[56] T. ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[57] Y. Elmehdwi, B. Samanthula, and W. Jiang. 2014. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *ICDE.* IEEE, 664–675.

[58] L. Fang, W. Ng, and W. Zhang. 2014. Encrypted scalar product protocol for outsourced data mining. In *CLOUD.* IEEE, 336–343.

[59] D. Fiore and R. Gennaro. 2012. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *CCS.* ACM, 501–512.

[60] J. Francis. 1962. The QR transformation: part 2. *Comput. J.* 4, 4 (1962), 332–345.

[61] R. Freivalds. 1979. Fast probabilistic algorithms. *Mathematical Foundations of Computer Science* (1979).

[62] T. S. Fun and A. Samsudin. 2016. A Survey of Homomorphic Encryption for Outsourced Big Data Computation. *TIIS* 10, 8 (2016), 3826–3851.

[63] R. Gennaro, C. Gentry, and B. Parno. 2010. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO.* 465–482.

[64] A. Gentry, C.and Sahai and B. Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013.* 75–92.

[65] C. Gentry. 2009. *A fully homomorphic encryption scheme.* Ph.D. Dissertation. Stanford University.

[66] C. Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *STOC.*

[67] C. Gentry and S. Halevi. 2011. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS.* IEEE, 107–109.

[68] C. Gentry and S. Halevi. 2011. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT.* 129–148.

[69] C. Gentry, S. Halevi, and N. Smart. 2012. Better bootstrapping in fully homomorphic encryption. In *International Workshop on Public Key Cryptography.* 1–16.

[70] C. Gentry, S. Halevi, and N. Smart. 2012. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT.*

[71] C. Gentry, S. Halevi, and N. Smart. 2012. Homomorphic evaluation of the AES circuit. In *CRYPTO 2012.*

[72] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to play any mental game. In *STOC.* ACM.

[73] S. Goldwasser, Y. Kalai, and G. Rothblum. 2008. Delegating computation: interactive proofs for muggles. In *STOC.* ACM, 113–122.

[74] P. Golle and I. Mironov. 2001. Uncheatable distributed computations. In *RSA.* 425–440.

[75] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. 2015. Leveled fully homomorphic signatures from standard lattices. In *STOC*. ACM, 469–477.

[76] S. Halevi and V. Shoup. 2014. HElib-An Implementation of homomorphic encryption.

[77] S. Halevi and V. Shoup. 2015. Bootstrapping for helib. In *EUROCRYPT*. 641–670.

[78] V. Herbert and C. Fontaine. 2017. Software Implementation of 2-Depth Pairing-based Homomorphic Encryption Scheme. IACR Cryptology ePrint Archive 2017/91.

[79] S. Hohenberger and A. Lysyanskaya. 2005. How to securely outsource cryptographic computations. In *TCC*. 264–282.

[80] R. Horn and C. Johnson. 2012. *Matrix analysis*. Cambridge university press.

[81] C. Hsu, C. Lu, and S. Pei. [n. d.]. Homomorphic encryption-based secure SIFT for privacy-preserving feature extraction. In *IS&T/SPIE Electronic Imaging*.

[82] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren. 2016. Securing SIFT: privacy-preserving outsourcing computation of feature extractions over encrypted image data. *IEEE Transactions on Image Processing* 25, 7 (2016), 3411–3425.

[83] F. Kerschbaum. 2015. Oblivious outsourcing of garbled circuit generation. In *SAC*. ACM, 2134–2140.

[84] Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan. 2016. SHIELD: scalable homomorphic implementation of encrypted data-classifiers. *IEEE Trans. Comput.* 65, 9 (2016), 2848–2858.

[85] L. Kondratyuk and M. Krivoruchenko. 1992. Superconducting quark matter in SU (2) colour group. *Zeitschrift für Physik A Hadrons and Nuclei* 344, 1 (1992), 99–115.

[86] X. Lei, X. Liao, T. Huang, and F. Heriniaina. 2014. Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. *Information Sciences* 280 (2014), 205–217.

[87] X. Lei, X. Liao, T. Huang, and H. Li. 2015. Cloud computing service: The case of large matrix determinant computation. *IEEE Transactions on Services Computing* 8, 5 (2015), 688–700.

[88] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu. 2013. Outsourcing large matrix inversion computation to a public cloud. *IEEE Transactions on Cloud Computing* 1, 1 (2013).

[89] F. Li, R. Shin, and V. Paxson. 2015. Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners. In *ACM Workshop on Cloud Computing Security*. 53–64.

[90] W. Liao, W. Du, S. Salinas, and P. Li. 2016. Efficient Privacy-Preserving Outsourcing of Large-Scale Convex Separable Programming for Smart Cities. In *HPCC/SmartCity/DSS*. IEEE, 1349–1356.

[91] C. Lin. 2007. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks* 18, 6 (2007), 1589–1596.

[92] K. Lin and M. Chen. 2010. Privacy-preserving outsourcing support vector machines with random transformation. In *The 16th SIGKDD*. ACM, 363–372.

[93] A. López-Alt, E. Tromer, and V. Vaikuntanathan. 2012. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*. ACM, 1219–1234.

[94] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. 2004. Fairplay – A Secure Two-Party Computation System. In *USENIX Security Symposium*.

[95] X. Meng, S. Kamara, K. Nissim, and G. Kollios. 2015. GRECS: graph encryption for approximate shortest distance queries. In *CCS*. ACM, 504–517.

[96] M. Minoux. 1986. *Mathematical programming: theory and algorithms*. John Wiley & Sons.

[97] P. Mohassel. 2011. Efficient and Secure Delegation of Linear Algebra. IACR Cryptology ePrint Archive 2011/605.

[98] M. Naehrig, K. Lauter, and V. Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Cloud computing security workshop*. ACM, 113–124.

[99] E. Nering. 1970. *Linear algebra and matrix theory*. John Wiley & Sons.

[100] M. Newman, D. Watts, and S. Strogatz. 2002. Random graph models of social networks. *The National Academy of Sciences* 99, suppl 1 (2002), 2566–2572.

[101] H. Nie, X. Chen, J. Li, J. Liu, and W. Lou. 2014. Efficient and verifiable algorithm for secure outsourcing of large-scale linear programming. In *AINA*. IEEE, 591–596.

[102] H. Nie, H. Ma, J. Wang, and X. Chen. 2014. Verifiable algorithm for secure outsourcing of systems of linear equations in the case of no solution. In *BWCCA*. IEEE, 572–577.

[103] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. 2013. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy*. 334–348.

[104] P. Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*. 223–238.

[105] M. Paindavoine and B. Vialla. 2015. Minimizing the number of bootstrappings in fully homomorphic encryption. In *SAC*. Springer, 25–43.

[106] V. Pan and J. Reif. 1985. Efficient parallel solution of linear systems. In *STOC*. ACM, 143–152.

[107] G. Piatetsky-Shapiro. 1991. Discovery, analysis, and presentation of strong rules. *KDD* (1991), 229–238.

[108] T. Pöppelmann and T. Güneysu. 2013. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *SAC*. 68–85.

[109] Z. Qin, J. Yan, K. Ren, C. Chen, and C. Wang. 2014. Towards efficient privacy-preserving image feature extraction in cloud computing. In *ACMMM*. ACM, 497–506.

[110] Z. Qin, J. Yan, K. Ren, C. Chen, and C. Wang. 2016. SecSIFT: Secure Image SIFT Feature Extraction in Cloud Computing. *TOMM* 12, 4s (2016), 65.

[111] R. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[112] Sergio Salinas, Xuhui Chen, Jinlong Ji, and Pan Li. 2016. A tutorial on secure outsourcing of large-scale computations for big data. *IEEE Access* 4 (2016), 1406–1416.

[113] S. Salinas, C. Luo, X. Chen, and P. Li. 2015. Efficient secure outsourcing of large-scale linear systems of equations. In *INFOCOM*. IEEE, 1035–1043.

[114] S. Salinas, C. Luo, W. Liao, and P. Li. 2016. Efficient secure outsourcing of large-scale quadratic programs. In *ACM ASIACCS*. ACM, 281–292.

[115] J. Sen. 2013. Security and privacy issues in cloud computing. *Architectures and Protocols for Secure Information Technology Infrastructures* (2013), 1–45.

[116] R. Sion. 2008. Towards secure data outsourcing. In *Handbook of Database Security*. 137–161.

[117] N. Smart and F. Vercauteren. 2010. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*. 420–443.

[118] N. Smart and F. Vercauteren. 2014. Fully homomorphic SIMD operations. *Designs, codes and cryptography* (2014), 1–25.

[119] E. Songhori, S. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar. 2015. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 411–428.

[120] J. Stevens. 2012. *Applied multivariate statistics for the social sciences*. Routledge.

[121] Z. Sun, T. Li, and N. Rishe. 2010. Large-scale matrix factorization using mapreduce. In *ICDMW*. IEEE.

[122] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian. 2001. Computer-attack graph generation tool. In *DISCEX*, Vol. 2. IEEE.

[123] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya. 2016. Ensuring security and privacy preservation for cloud data services. *ACM Computing Surveys (CSUR)* 49, 1 (2016), 13.

[124] P. Thibodeau. 2016. U.S. to have 200-petaflop supercomputer by early 2018. http://www.computerworld.com/article/3086178/high-performance-computing

[125] J. Vaidya and C. Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *SIGKDD*. ACM.

[126] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *EUROCRYPT*. 24–43.

[127] C. Wang, K. Ren, and J. Wang. 2011. Secure and practical outsourcing of linear programming in cloud computing. In *INFOCOM*. IEEE, 820–828.

[128] C. Wang, K. Ren, and J. Wang. 2016. Secure optimization computation outsourcing in cloud computing: A case study of linear programming. *IEEE Trans. Comput.* 65, 1 (2016), 216–229.

[129] C. Wang, K. Ren, J. Wang, and Q. Wang. 2013. Harnessing the cloud for securely outsourcing large-scale systems of linear equations. *IEEE Transactions on Parallel and Distributed Systems* 24, 6 (2013).

[130] C. Wang, B. Zhang, K. Ren, and J. Roveda. 2013. Privacy-assured outsourcing of image reconstruction service in cloud. *IEEE Transactions on Emerging Topics in Computing* 1, 1 (2013), 166–177.

[131] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang. 2015. CloudBI: Practical privacy-preserving outsourcing of biometric identification in the cloud. In *ESORICS*. 186–205.

[132] Q. Wang, S. Hu, K. Ren, J. Wang, Z. Wang, and M. Du. 2016. Catch me in the dark: Effective privacy-preserving outsourcing of feature extractions over image data. In *INFOCOM*. IEEE, 1–9.

[133] Q. Wang, S. Hu, J. Wang, and K. Ren. 2016. Secure Surfing: Privacy-Preserving Speeded-Up Robust Feature Extractor. In *ICDCS*. IEEE, 700–710.

[134] Q. Wang, K. Ren, M. Du, Q. Li, and A. Mohaisen. 2017. SecGDB: Graph Encryption for Exact Shortest Distance Queries with Efficient Updates. In *Financial Cryptography*.

[135] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren. 2016. SecHOG: Privacy-Preserving Outsourcing Computation of Histogram of Oriented Gradients in the Cloud. In *ASIA CCS*. ACM, 257–268.

[136] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar. 2012. Accelerating fully homomorphic encryption using GPU. In *HPEC*. IEEE, 1–5.

[137] X. Wang, K. Nayak, C. Liu, T. Chan, E. Shi, E. Stefanov, and Y. Huang. 2014. Oblivious data structures. In *ASIA CCS*. ACM, 215–226.

[138] W. Wong, D. Cheung, E. Hung, B. Kao, and N. Mamoulis. 2007. Security in outsourcing of association rule mining. In *The 33rd VLDB*. VLDB Endowment, 111–122.

[139] W. Wong, D. Cheung, E. Hung, B. Kao, and N. Mamoulis. 2009. An audit environment for outsourcing of frequent itemset mining. *The VLDB Endowment* 2, 1 (2009), 1162–1173.

[140] W. Wong, D. Cheung, B. Kao, and N. Mamoulis. 2009. Secure knn computation on encrypted databases. In *SIGMOD*. ACM, 139–152.

[141] D. Wu, J. Zimmerman, J. Planul, and J. Mitchell. 2016. Privacy-preserving shortest path computation. *NDSS* (2016).

[142] G. Xu, G. Amariucai, and Y. Guan. 2017. Delegation of computation with verification outsourcing: Curious verifiers. *IEEE Transactions on Parallel and Distributed Systems* 28, 3 (2017), 717–730.

[143] X. Yan and X. Su. 2009. *Linear regression analysis: theory and computing*. World Scientific.

[144] A. Yao. 1982. Protocols for secure computations. In *FOCS*. IEEE, 160–164.

[145] A. Yao. 1986. How to generate and exchange secrets. In *FOCS*. IEEE, 162–167.

[146] B. Yao, F. Li, and X. Xiao. 2013. Secure nearest neighbor revisited. In *ICDE*. IEEE, 733–744.

[147] Y. Yu, Y. Luo, D. Wang, S. Fu, and M. Xu. 2016. Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations. In *ICC*. IEEE, 1–6.

[148] J. Yuan and S. Yu. 2013. Efficient privacy-preserving biometric identification in cloud computing. In *INFOCOM*. IEEE.

[149] Y. Zhang and M. Blanton. 2014. Efficient secure and verifiable outsourcing of matrix multiplications. In *International Conference on Information Security*. 158–178.

[150] Y. Zhang, J. Zhou, L. Zhang, F. Chen, and X. Lei. 2015. Support-set-assured parallel outsourcing of sparse reconstruction service for compressive sensing in multi-clouds. In *SocialSec*. IEEE, 1–6.

[151] J. Zhou, C., and X. Dong. 2016. PPOPM: More Efficient Privacy Preserving Outsourced Pattern Matching. In *ESORICS*. 135–153.

[152] L. Zhou and C. Li. 2015. Outsourcing large-scale quadratic programming to a public cloud. *IEEE Access* 3 (2015).

[153] L. Zhou and C. Li. 2016. Outsourcing Eigen-Decomposition and Singular Value Decomposition of Large Matrix to a Public Cloud. *IEEE Access* 4 (2016), 869–879.

Received June 2017; revised September 2017; accepted November 2017