IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON THEORETICAL FOUNDATIONS FOR BIG DATA APPLICATIONS: CHALLENGES AND OPPORTUNITIES

Received March 10, 2016, accepted March 25, 2016, date of publication April 4, 2016, date of current version April 21, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2549982

A Tutorial on Secure Outsourcing of Large-scale Computations for Big Data

SERGIO SALINAS[1], (Member, IEEE), XUHUI CHEN[2], (Student Member, IEEE), JINLONG JI[2], (Student Member, IEEE), AND PAN LI[2], (Member, IEEE)

[1]Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS 67260, USA
[2]Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106, USA

Corresponding author: P. Li (lipan@case.edu)

This work was supported by the Division of Computer and Network Systems through the U.S. National Science Foundation under Grant CNS-1149786 and CNS-1343220.

ABSTRACT Today's society is collecting a massive and exponentially growing amount of data that can potentially revolutionize scientific and engineering fields, and promote business innovations. With the advent of cloud computing, in order to analyze data in a cost-effective and practical way, users can outsource their computing tasks to the cloud, which offers access to vast computing resources on an on-demand and pay-per-use basis. However, since users' data contains sensitive information that needs to be kept secret for ethical, security, or legal reasons, many users are reluctant to adopt cloud computing. To this end, researchers have proposed techniques that enable users to offload computations to the cloud while protecting their data privacy. In this paper, we review the recent advances in the secure outsourcing of large-scale computations for a big data analysis. We first introduce two most fundamental and common computational problems, i.e., linear algebra and optimization, and then provide an extensive review of the data privacy preserving techniques. After that, we explain how researchers have exploited the data privacy preserving techniques to construct secure outsourcing algorithms for large-scale computations.

INDEX TERMS Big data, privacy, cloud computing, linear algebra, optimization.

I. INTRODUCTION

The amount of data that is being collected by today's society is exponentially growing. The digital world will increase from 4.4 zetabytes (trillion gigabytes) in 2013 to 44 zetabytes by 2020, more than doubling every two years [1]. Such massive data holds the potential to rapidly advance scientific and engineering knowledge and promote business innovations [2]–[5]. For example, e-commerce companies can improve product recommendations by mining billions of customer transactions [6]; power engineers can monitor the electric grid in real-time based on the enormous amount of sensor data [7]; and financial firms can increase the returns of their portfolios by analyzing the daily deluge of stock market data. Obviously, we have massive data in all these fields, which needs to be stored, managed, and more importantly, analyzed. However, users face an enormous challenge in trying to analyze such huge amounts of data in a timely and cost-effective way.

Specifically, due to the limited computing and RAM (random access memory) capacity of traditional hardware, users are unable to analyze large-scale data sets in a feasible amount of time. In fact, analyzing massive data usually requires vast and sophisticated computing infrastructure. For example, governments and universities have successfully employed supercomputers to solve very heavy computing tasks, such as predicting climate change and simulating protein folding. Unfortunately, because of the high cost of supercomputers, e.g., hundreds of millions of dollars or even more, most users lack access to adequate computing facilities for big data applications. Even small in-house computing clusters are too expensive, and their computing resources may still be insufficient [8], [9].

Recently, cloud computing has been proposed as an efficient and economical way for resource-limited users to analyze massive data sets. In this computing paradigm, cloud users outsource their computing tasks to a cloud server [10]–[14], which contains a large amount of computing resources and offers them on an on-demand and pay-per-use basis [15]. Therefore, users can share the cloud's resources with each other, and avoid purchasing, installing, and maintaining sophisticated and expensive computing hardware and software. For instance, the video streaming company Netflix

employs the cloud to accommodate users' huge and highly dynamic demand in a cost-effective way [16].

Although users recognize the advantages of cloud computing, many of them are reluctant to adopt it due to privacy concerns [17]. To be more prominent, in many cases, users' data is very sensitive and should be kept secret from the cloud for ethical, security, or legal reasons [18]–[20]. For example, outsourcing product recommendations in e-commerce can give unauthorized access to users' shopping habits [21]; a power company's data may reveal the topology of the system, thus enabling attacks on the electric grid [22]; and outsourcing portfolio optimization may compromise financial firms' market research. In fact, users' private data is vulnerable to malicious cloud service providers, who can directly snoop into its users' data, and third-party adversaries, who can launch a number of attacks against the cloud [23]–[27]. Therefore, to enable scientists and engineers to revolutionize their fields through the analysis of large-scale data, it is important to design secure outsourcing tools that preserve their data privacy.

We notice that two types of fundamental mathematical computations frequently appear in large-scale data analytics and computing: linear algebra and optimization. In particular, linear algebra operations are arguably the most fundamental and frequently used computations in big data analysis. For example, least-squares estimation, which is widely used for linear regression, involves the solution of an overdetermined linear system of equations; and iterative solutions for non-linear systems, such as the Newton-Raphson method, usually employ solving a linear system of equations as a subroutine. Besides, to perform more sophisticated analysis, optimization is frequently used in big data applications. For example, scheduling routes in intelligent transportation systems can be done by solving a linear program; and support vector machines, a commonly used machine learning algorithm, employ both linear and quadratic programs.

In this tutorial, we review secure outsourcing techniques for for large-scale data analytics and computing. Specifically, we first introduce linear algebra and optimization problems and their respective solution methods. We then review the most common privacy-preserving techniques for large-scale data sets, and explain how researchers have used them to securely outsource linear algebra and optimization computations.

The rest of this paper is organized as follows. In Section II, we introduce two most fundamental computational problems employed in large-scale data analysis. In Section III, we introduce the cloud computing system architecture and the threat model. In Section IV, we present privacy-preserving techniques for secure outsourcing of large-scale computations. Section V explains how linear algebra and optimization computations can be outsourced to the cloud while preserving data privacy. We finally conclude this paper in Section VI.

II. FUNDAMENTAL COMPUTATIONAL PROBLEMS IN BIG DATA

In this section, we present two most fundamental computational problems, i.e., linear algebra and optimization, employed in large-scale data analysis.

A. LINEAR ALGEBRA

Many problems that involve large-scale data analysis are fundamentally based on solving linear systems of equations (LSEs). For example, in image processing, power flow estimation, and portfolio optimization, least-squares problems can be reformulated as an LSE; in social network analysis and web search engine design, the eigenvector centrality problem is naturally posed as an LSE; and linear regression, which is a very common statistical analysis technique, can also be computed through solving an LSE. We formally define an LSE as follows:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times m}$ is the coefficient matrix, $\mathbf{x} \in \mathbb{R}^{n\times 1}$ is the solution vector, and $\mathbf{b} \in \mathbb{R}^{m\times 1}$ is the constant vector. We assume that $\mathbf{A}$ is square, i.e., $m = n$, non-singular, symmetric, and positive-definite. In fact, given an arbitrary coefficient matrix $\mathbf{A}$, the user can always find an equivalent LSE by employing $\mathbf{A}' = \mathbf{A}^\top\mathbf{A}$ as the coefficient matrix and $\mathbf{b}' = \mathbf{A}^\top\mathbf{b}$ as the constant vector, where matrix $\mathbf{A}'$ is guaranteed to be non-singular, symmetric, and positive-definite.

1) MATRIX INVERSION

To solve the large-scale LSEs in (1) by matrix inversion, a user first finds matrix $\mathbf{A}^{-1}$ such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \tag{2}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, and then computes $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

By computing $\mathbf{A}^{-1}$ once and storing it, the user can efficiently solve several LSEs that share the same coefficient matrix $\mathbf{A}$, but have different constant vectors $\mathbf{b}$. Besides, matrix inversion often appears as a stand alone computation. For instance, in large-scale data visualization, matrix inversion is employed in 3-D rendering processes, such as screen-to-world ray casting and world-to-subspace-to-world object transformations [28].

Unfortunately, since the most efficient matrix inversion algorithms are based on matrix multiplications, the complexity of computing $\mathbf{A}^{-1}$ is too high for the user, particularly in big data applications.

2) ITERATIVE SOLUTION METHODS

Iterative methods can solve LSEs with lower computational complexity by avoiding matrix multiplications. We briefly review two of the most frequently used iterative solution methods: the Jacobi method and the conjugate gradient method (CGM).

The main idea of the Jacobi method is to iteratively update the solution vector based on a transformed coefficient matrix. Specifically, we replace $\mathbf{A}$ with $\mathbf{D}+\mathbf{R}$ in (1), where $\mathbf{D}$ and $\mathbf{R}$ are the matrices containing the diagonal and the off-diagonal elements of $\mathbf{A}$, respectively. Then, by rearranging terms, we reach the following Jacobi iteration to solve (1), i.e.,

$$\mathbf{x}^{k+1} = \mathbf{T}\mathbf{x}^k + \mathbf{c} \tag{3}$$

where $\mathbf{T} = -\mathbf{D}^{-1}\mathbf{R}$ and $\mathbf{c} = \mathbf{D}^{-1}\mathbf{b}$.

To carry out the Jacobi method, the user assigns random values to the initial solution vector $\mathbf{x}^0$, and computes (3) until the following condition is met,

$$||\mathbf{x}^k - \mathbf{x}^{k+1}||_2 \leq \epsilon, \tag{4}$$

where $||\cdot||_2$ is the 2-norm and $0 < \epsilon < 1$ is the tolerance parameter. If the coefficient matrix $\mathbf{A}$ is diagonally dominant, the Jacobi iteration converges in $K \ll n$ iterations. Otherwise, the algorithm may diverge [29].

Another iterative solution method is the conjugate gradient method (CGM), which offers guaranteed convergence if the coefficient matrix is symmetric and positive-definite. To illustrate the main idea of the CGM, we first note that solving (1) is equivalent to solving the following unconstrained optimization program

$$\min f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{A}\mathbf{x} - \mathbf{b}\mathbf{x} \tag{5}$$

when $\mathbf{A}$ is nonsingular, symmetric and positive-definite [30].

The CGM employs a set of vectors $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ that are conjugate with respect to $\mathbf{A}$, that is, at iteration k the following condition is met:

$$\mathbf{p}_k^\top\mathbf{A}\mathbf{p}_i = 0 \quad \text{for} \quad i = 0, \ldots, k-1. \tag{6}$$

Using the conjugacy property of vectors in $\mathcal{P}$, we can find the solution in at most n steps by computing a sequence of solution approximations as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k \tag{7}$$

where α_k is the one-dimensional minimizer of (5) along $\mathbf{x}_k + \alpha_k\mathbf{p}_k$. The minimizer α_k is given by

$$\alpha_k = \frac{-\mathbf{r}_k^\top\mathbf{p}_k}{\mathbf{p}_k^\top\mathbf{A}\mathbf{p}_k} \tag{8}$$

where $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ is called the residual.

Moreover, we can iteratively find the residual based on (7) as follows:

$$\begin{aligned}\mathbf{r}_{k+1} &= \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} \\ &= \mathbf{A}(\mathbf{x}_k + \alpha_k\mathbf{p}_k) - \mathbf{b} = \mathbf{r}_k + \alpha_k\mathbf{A}\mathbf{p}_k.\end{aligned} \tag{9}$$

The CGM finds a new conjugate vector $\mathbf{p}_{k+1}$ at iteration k by a linear combination of the negative residual, i.e., the steepest descent direction of $f(\mathbf{x})$, and the current conjugate vector $\mathbf{p}_k$, that is,

$$\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k \tag{10}$$

where β_{k+1} is chosen in such a way that $\mathbf{p}_{k+1}^\top$ and $\mathbf{p}_k$ meet condition (6).

Since $\mathbf{x}_k$ minimizes $f(\mathbf{x})$ along $\mathbf{p}_k$, it can be shown that $\mathbf{r}_k^\top\mathbf{p}_i = 0$ for $i = 0, 1, \ldots, k-1$ [30]. Using this fact and equation (10), a more efficient computation for (8) can be found, namely,

$$\begin{aligned}\alpha_k &= \frac{-\mathbf{r}_k^\top(-\mathbf{r}_k + \beta_k\mathbf{p}_{k-1})}{\mathbf{p}_k^\top\mathbf{A}\mathbf{p}_k} \\ &= \frac{\mathbf{r}_k^\top\mathbf{r}_k}{\mathbf{p}_k^\top\mathbf{A}\mathbf{p}_k}.\end{aligned}$$

Similarly, using (9), we can find the formulation for β_{k+1}

$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^\top\mathbf{r}_{k+1}}{\mathbf{r}_k^\top\mathbf{r}_k}.$$

The iterations stop when the following stopping criteria is met

$$\sqrt{\mathbf{r}_k^\top\mathbf{r}_k} \leq \nu||\mathbf{b}||_2. \tag{11}$$

where ν is a tolerance value.

To summarize the above, the CGM algorithm is as follows. At iteration $k = 0$, we have

$$\mathbf{r}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b} \tag{12}$$

$$\mathbf{p}_0 = -\mathbf{r}_0 \tag{13}$$

$$k = 0 \tag{14}$$

and at iteration $k \geq 0$ we have the following iterative equations:

$$\alpha_k = \frac{\mathbf{r}_k^\top\mathbf{r}_k}{\mathbf{p}_k^\top\mathbf{A}\mathbf{p}_k} \tag{15}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k\mathbf{A}\mathbf{p}_k \tag{16}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k \tag{17}$$

$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^\top\mathbf{r}_{k+1}}{\mathbf{r}_k^\top\mathbf{r}_k} \tag{18}$$

$$\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k \tag{19}$$

Compared to other methods, e.g., matrix inversion, Gaussian eliminations, QR decomposition, the Jacobi iteration and the CGM offer a feasible algorithm for extremely large-scale systems.

B. OPTIMIZATION

In large-scale data analysis, many computations can be posed as an optimization problem that has a linear objective and affine constraints, i.e., a linear program (LP), or a quadratic objective and affine constraints, i.e., a quadratic program (QP). For example, in machine learning, support vector machines employ both linear and quadratic programs [31]; in financial analysis, linear programs have been successfully used to manage stock portfolios and evaluate risk; and in power grids, linear optimization models are used to control generation dispatch and perform real-time monitoring [32]. In what follows, we introduce linear and quadratic programs, and describe their respective Lagrange dual problems.

1) LINEAR PROGRAM

An optimization problem with linear objective and linear constraints, i.e., an LP, is defined as follows [33]:

$$\min_{\mathbf{x}} \quad z = \mathbf{c}^T\mathbf{x} \tag{20a}$$
$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{20b}$$
$$\mathbf{x} \geq \mathbf{0} \tag{20c}$$

where $\mathbf{c} \in \mathbb{R}^n$ is the cost vector and $\mathbf{x} \in \mathbb{R}^n$ is the variable vector. Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, vector $\mathbf{b}$, and (20c) describe a set of affine constraints. The user aims to find the optimal solution $\mathbf{x}^*$ such that the objective in (20a) is minimized and the constraints (20b) and (20c) are satisfied. We assume that $\mathbf{A}$ is full-rank and the constraints are feasible.

Linear programs can be efficiently solved through iterative solution methods such as the Simplex algorithm [33] and interior point methods [34].

2) QUADRATIC PROGRAM

A QP is an optimization problem that has a quadratic objective and affine constraints [34], i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} - \mathbf{b}^\top\mathbf{x} \tag{21a}$$
$$\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{c} \tag{21b}$$

where $\mathbf{x} \in \mathbb{R}^{n\times 1}$ is the optimization variable, and $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^{n\times 1}$ are the quadratic and affine coefficients, respectively. The matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, and the vector $\mathbf{c} \in \mathbb{R}^{m\times 1}$ define the set of affine linear constraints. Similar to an LP, the user intends to find the optimal solution $\mathbf{x}^*$ such that the objective in (21a) is minimized and the constraints (21b) are satisfied. We assume that the coefficient matrix $\mathbf{Q}$ is positive definite and symmetric, and hence the QP in (21) has a unique solution. We also assume that $\mathbf{A}$ is full-rank.

3) THE LAGRANGE DUAL PROBLEM

The Lagrange dual problem is an equivalent optimization problem that is often easier to solve and provides useful insights into the original problem. For example, as we will see later, the Lagrange dual problem usually has a much simpler constraint set that allows us to employ more efficient solution methods. Therefore, we can solve the Lagrange dual problem to either verify the solution to the original problem or solve the original problem. In what follows, we derive the Lagrange dual problems for both LPs and QPs.

The Lagrange Dual Problem for LPs: To find the dual problem of (20), we first form the Lagrangian $\mathcal{L} : \mathbb{R}^{n\times 1} \times \mathbb{R}^{m\times 1} \times \mathbb{R}^{n\times 1} \to \mathbb{R}$ as follows:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{c}^\top\mathbf{x} + \boldsymbol{\lambda}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) - \boldsymbol{\nu}^\top\mathbf{x} \tag{22}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{m\times 1}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n\times 1}$ are the equality and inequality dual variable vectors, respectively.

We define the Lagrange dual function $g : \mathbb{R}^{m\times 1} \times \mathbb{R}^{n\times 1} \to \mathbb{R}$ as the infimum value of (22) over $\mathbf{x}$ (regarding $\boldsymbol{\lambda} \in \mathbb{R}^{m\times 1}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n\times 1}$), i.e.,

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\nu}) &= \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &= -\mathbf{b}^\top\boldsymbol{\lambda} + \inf_{\mathbf{x}}(\mathbf{c} + \mathbf{A}^\top\boldsymbol{\lambda} - \boldsymbol{\nu})^\top\mathbf{x}. \end{aligned} \tag{23}$$

To solve the Lagrange dual function, we observe that (23) has a lower bound when the coefficient of $\mathbf{x}$ is equal to zero, and otherwise it is unbounded, that is

$$g(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \begin{cases} -\mathbf{b}^\top\boldsymbol{\lambda} & \mathbf{A}^\top\boldsymbol{\lambda} - \boldsymbol{\nu} + \mathbf{c} = \mathbf{0}, \\ -\infty & \text{otherwise.} \end{cases} \tag{24}$$

By using the first case of (24) as the objective function, and requiring $\mathbf{A}^\top\boldsymbol{\lambda} - \boldsymbol{\nu} + \mathbf{c} = 0$ and $\boldsymbol{\nu}$ to be nonnegative, we arrive at the dual problem of (20):

$$\begin{aligned} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}} \quad & g = -\mathbf{b}^T\boldsymbol{\lambda} \\ \text{subject to} \quad & \mathbf{A}^\top\boldsymbol{\lambda} - \boldsymbol{\nu} + \mathbf{c} = \mathbf{0} \\ & \boldsymbol{\nu} \geq \mathbf{0} \end{aligned} \tag{25}$$

which can be further simplified into

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & g = -\mathbf{b}^T\boldsymbol{\lambda} \\ \text{subject to} \quad & \mathbf{A}^\top\boldsymbol{\lambda} + \mathbf{c} \geq \mathbf{0} \end{aligned} \tag{26}$$

Since we have assumed that $\mathbf{A}$ is full-rank and that the affine constraints are feasible, we have that the strong duality holds [34] and thus

$$\mathbf{c}^\top\mathbf{x}^* = -\mathbf{b}^\top\boldsymbol{\lambda}^*. \tag{27}$$

The Lagrange Dual Problem for QPs: Since directly solving the optimization problem in (21) requires expensive solution methods such as interior point methods, it is often preferable to solve the Lagrangian dual problem, which only has non-negativity constraints and can be more efficiently solved.

Following a similar procedure to the above, we first form the Lagrangian $\mathcal{L} : \mathbb{R}^{n\times 1} \times \mathbb{R}^{m\times 1} \to \mathbb{R}$ as follows [34]:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{28}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{m\times 1}$ is the vector of dual variables, and $\mathbf{A}$ and $\mathbf{b}$ are defined in (21a).

We define the Lagrange dual function $g : \mathbb{R}^{m\times 1} \to \mathbb{R}$ as the minimum value of the Lagrangian over $\mathbf{x}$ (for $\boldsymbol{\lambda} \in \mathbb{R}^{m\times 1}$), i.e.,

$$g(\boldsymbol{\lambda}) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \tag{29}$$

To solve the Lagrange dual function, we take the derivative of equation (28) with respect to $\mathbf{x}$ and set it to zero as follows:

$$\mathbf{x}^\top\mathbf{Q} - \mathbf{b}^\top + \boldsymbol{\lambda}^\top\mathbf{A} = \mathbf{0} \tag{30}$$

where we have used the definition of $f(\cdot)$ in (21a). Taking the transpose in (30) and solving for $\mathbf{x}$, we get

$$\mathbf{x} = \mathbf{Q}^{-1}(\mathbf{b} - \mathbf{A}^\top\boldsymbol{\lambda}) \tag{31}$$

Then, by plugging (31) into (28) and considering that $\mathbf{Q}$ is symmetric, we can rewrite the Lagrange dual function as

$$\begin{aligned} g(\boldsymbol{\lambda}) = & -\frac{1}{2}\boldsymbol{\lambda}^\top\mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^\top\boldsymbol{\lambda} \\ & -\boldsymbol{\lambda}^\top(\mathbf{c} - \mathbf{A}\mathbf{Q}^{-1}\mathbf{b}) \\ & -\frac{1}{2}\mathbf{b}^\top\mathbf{Q}^{-1}\mathbf{b} \end{aligned} \tag{32}$$

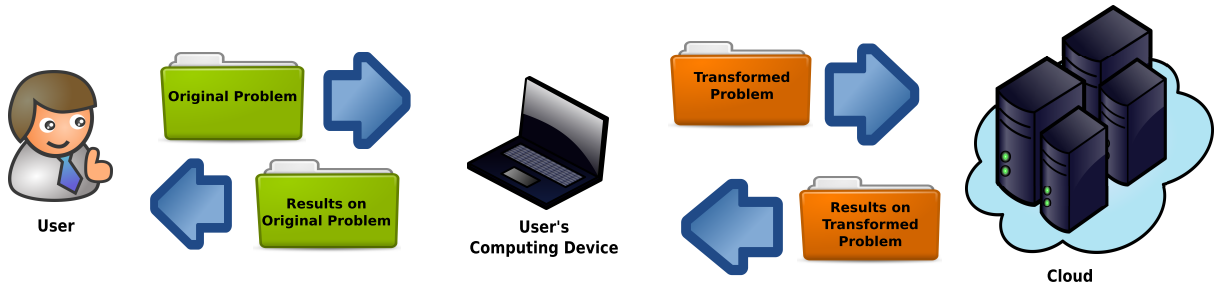


FIGURE 1. A typical architecture for secure outsourcing in big data analysis.

By using equation (32) as the objective function and requiring $\boldsymbol{\lambda}$ to be non-negative, we arrive at the dual problem of the QP in (21), i.e.,

$$\min_{\boldsymbol{\lambda}} \quad g(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^\top \mathbf{P}\boldsymbol{\lambda} + \boldsymbol{\lambda}^\top \mathbf{r} \tag{33a}$$

$$\text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0} \tag{33b}$$

where $\mathbf{P} = \mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^\top$ and it is positive definite and symmetric, and $\mathbf{r} = \mathbf{c} - \mathbf{A}\mathbf{Q}^{-1}\mathbf{b}$. We denote the solution to (33) as $\boldsymbol{\lambda}^*$.

Since the problem (21) is convex and the affine constraints are feasible, then the strong duality holds [34] and we have that

$$\mathbf{x}^* = \mathbf{Q}^{-1}(\mathbf{b} - \mathbf{A}^\top \boldsymbol{\lambda}^*). \tag{34}$$

Moreover, to solve the optimization problem in (33), we can use the Gauss-Seidel iteration as follows:

$$\lambda_j(t+1) = \arg\min_{\lambda_j \geq 0} g(\lambda_1(t+1), \ldots, \lambda_{j-1}(t+1), \lambda_j, \lambda_{j+1}(t), \ldots, \lambda_m(t)) \tag{35}$$

where $j \in [1, m]$. Intuitively, the algorithm updates λ_j (for all $j \in [1, m]$) one at a time, and uses the most recent updates as they become available.

Let $(\lambda_1(t+1), \ldots, \lambda_{j-1}(t+1), \lambda_j, \lambda_{j+1}(t), \ldots, \lambda_m(t))$ be denoted as $\boldsymbol{\lambda}_j(t)$. We can solve equation (35) analytically by taking the partial derivative of $g(\boldsymbol{\lambda}_j(t))$ in (33a) with respect to λ_j and setting it to zero:

$$\frac{\partial g(\boldsymbol{\lambda}_j(t))}{\partial \lambda_j} = r_j + \mathbf{p}_j \boldsymbol{\lambda}_j(t) = 0$$

where r_j is the jth element of $\mathbf{r}$, and $\mathbf{p}_j$ is the jth row of $\mathbf{P}$.

Then, the unconstrained minimum of $g(\cdot)$ along the jth coordinate starting from $\lambda_j(t)$ is attained at

$$\lambda_j'(t+1) = \lambda_j(t) - \frac{1}{p_{j,j}}(r_j + \mathbf{p}_j \boldsymbol{\lambda}_j(t)) \quad \forall j \in [1, m]$$

where t is the iteration index.

Thus, taking into account the non-negativity constraint and holding all the other variables constant, we get that the Gauss-Seidel iteration, when λ_j is updated, is

$$\lambda_j(t+1) = \max\{0, \lambda_j'(t+1)\} = \max\{0, \lambda_j(t) - \frac{1}{p_{j,j}}(r_j + \mathbf{p}_j \boldsymbol{\lambda}_j(t))\}, \tag{36}$$

$$\lambda_i(t+1) = \lambda_i(t) \quad \forall i \neq j. \tag{37}$$

After all the $\lambda_i(t+1)$'s $(1 \leq i \leq m)$ are updated, the GSA iteration proceeds to the next. The iterations continue until the following stopping criteria is satisfied:

$$||\boldsymbol{\lambda}(t+1) - \boldsymbol{\lambda}(t)||_2 \leq \epsilon \tag{38}$$

where $0 < \epsilon < 1$ is the tolerance parameter.

III. SYSTEM ARCHITECTURE

In this section, we present the most common cloud computing architecture for secure outsourcing of large-scale computations, and describe the threat model.

A. SYSTEM ARCHITECTURES

As shown in Fig 1, the system architecture for secure outsourcing of large-scale computations consists of a resource-constrained user and a remote resource-rich cloud server. The user aims to solve one of the large-scale problems described in Section II. However, due to the massive data involved in the problem, the user is unable to solve it by itself in a feasible amount of time. Therefore, the user outsources its most computationally expensive tasks to the cloud.

B. THREAT MODELS

The literature assumes two threat models for the cloud, i.e., a semi-honest model and a malicious model. In the semi-honest model, the cloud attempts to learn the user's private data from its outsourced data and from the results of its own computations. In the malicious model, the cloud has the additional ability to deviate from the proposed protocols or return erroneous results.

Before the cloud can help perform any computations, the user first needs to upload some of its data to the cloud. However, to prevent the cloud from learning private information, the user's data must be well protected. In particular, in the user's data, the non-zero elements' values and positions both carry private information. Besides, we usually need to keep the zeros' in users' data, which can enable more efficient computations, e.g., sparse matrix computations or parallel computations [35]. In the following, we adopt the privacy notion of computational indistinguishability [36], and introduce a few similar but different definitions.

First, in the computational problems described in Section II, we observe that the numerical values in the matrices and vectors are private. Specifically, in LSEs,

the values of the elements in the coefficient matrix $\mathbf{A}$ and constant vector $\mathbf{b}$ are private. Similarly, in LPs, the cost function $\mathbf{c}$, and the constraint set defined by $\mathbf{A}$ and $\mathbf{b}$ should be kept secret from the cloud. In the case of QPs, the objective coefficients $\mathbf{Q}$ and $\mathbf{b}$, as well as the constraint set defined by $\mathbf{A}$ and $\mathbf{c}$ are private. In all the LSE, LP, and QP problems, the solution $\mathbf{x}^*$ is private. We have the following two definitions of computational indistinguishability in value.

Definition 1: Computational Indistinguishability in Value: Let $\mathbf{R} \in \mathbb{R}^{m\times n}$ be a random matrix with entries in its jth column sampled from a uniform distribution with interval $[-R_j, R_j]$ $\forall j \in [1, n]$. Matrices $\mathbf{R}$ and $\mathbf{Q}$ are computationally indistinguishable in value if for any probabilistic polynomial time distinguisher $D(\cdot)$ there exists a negligible function $\mu(\cdot)$ such that [37]

$$|P[D(r_{i,j}) = 1] - Pr[D(q_{i,j}) = 1]| \leq \mu \quad \forall\, i, j \qquad (39)$$

where $r_{i,j}$ is the element in the ith row and jth column of $\mathbf{R}$, and $q_{i,j}$ is the element in the ith row and jth column of $\mathbf{Q}$. Distinguisher $D(\cdot)$ outputs 1 when it identifies the input as a non-uniform distribution in the range $[-R_j, R_j]$, and zero otherwise.

Definition 2: Computational Indistinguishability in Value under a CPA: We say that a matrix transformation scheme has indistinguishable transformations in value under a chosen-plaintext attack (or is CPA-secure in value) if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function μ, such that the probability of distinguishing two matrix transformations in value in a CPA indistinguishability experiment is less than $1/2 + \mu$ [35].

Note that if a matrix transformation scheme results in a matrix satisfying Definition 1, then it satisfies Definition 2 as well.

Moreover, the positions of the non-zero elements in a matrix (i.e., the matrix's structure) contain private information that should also be hidden from the cloud. For example, in power system state estimation, the system matrix contains the topology of the network, which can be used to launch an attack against the grid [22]. To protect a matrix's structure, we can permute its rows and columns in such a way that the non-zero elements occupy positions that are indistinguishable from those of the non-zero elements in another permuted matrix. We give the definition of secure permutation below.

Definition 3: Indistinguishability in Structure under a CPA: We say that a permutation scheme has indistinguishable permutations under a chosen-plaintext attack (or is CPA-secure) if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function μ, such that the probability of distinguishing two permutations in a CPA indistinguishability experiment is less than $1/2 + \mu$ [35].

IV. DATA PRIVACY PRESERVING APPROACHES

To delegate a computing task to the cloud, the user first needs to perform some computations on its data. These computations should require a moderate effort from the user, hide the data from the cloud, and allow the cloud to return a meaningful result. To this end, researchers have proposed mainly two kinds of approaches: cryptography-based techniques and perturbation-based techniques. In this section, we describe the state-of-the-art of privacy-preserving approaches.

A. CRYPTOGRAPHY-BASED APPROACHES

In their seminal work, Gennaro et. al [38] develop the first fully homomorphic encryption (FHE) scheme which allows users to securely outsource any computations to the cloud. However, it requires an expensive preprocessing phase at the user and adds significant computing overhead at the cloud. Although some efforts have been made to improve the practicality of FHE [39], it remains very expensive for big data applications.

To overcome this challenge, a user can employ partially homomorphic encryption (PHE) to conceal its data and allow the cloud to securely carry out a specific computational task. In other words, compared with FHE that supports arbitrary computation on ciphertexts, PHE only allows homomorphic computation of some operations on ciphertexts (e.g., additions, multiplications, quadratic functions, etc.) and hence is usually less complex. For example, consider the Paillier homomorphic cryptosystem, a frequently used PHE. Letting $E(\cdot)$ denote the encryption function, we have that

$$E(m_1 + m_2) = E(m_1) \cdot E(m_2)$$

and

$$E(m_1 \cdot c) = E(m_1)^c$$

where m_1 and m_2 are plaintexts, and c is a constant. Other PHE cryptosystems include El-Gamal [40] and Goldwasser and Micali [41].

Notice that although FHE and PHE schemes satisfy Definition 2, they may still not be applicable for big data applications due to the expensive encryptions/decryptions as well as the complex operations on ciphertexts.

B. PERTURBATION-BASED APPROACHES

Perturbation-based approaches conceal a user's sensitive data by performing linear algebra operations between its private matrices and carefully designed random matrices. In this section, we introduce three main perturbation-based approaches employed to protect the user's privacy: matrix addition, matrix multiplication, and row and column permutations.

1) MATRIX ADDITION

A user can hide the values of the elements in its private matrix $\mathbf{H} \in \mathbb{R}^{m\times n}$ by performing the following matrix addition [37]:

$$\bar{\mathbf{H}} = \mathbf{H} + \mathbf{Z} \qquad (40)$$

where $\mathbf{Z} \in \mathbb{R}^{m\times n}$ is a random matrix, and $\bar{h}_{i,j} = h_{i,j} + z_{i,j}$ $(\forall i \in [1, m],\ j \in [1, n])$.

To reduce the user's computational complexity, the random matrix $\mathbf{Z}$ is formed by a vector outer-product, i.e.,

$$\mathbf{Z} = \mathbf{u}\mathbf{v}^\top \tag{41}$$

where $\mathbf{u} \in \mathbb{R}^{m\times 1}$ is a vector of uniformly distributed random variables and vector $\mathbf{v} \in \mathbb{R}^{n\times 1}$ is a vector of arbitrary positive constants. Note that element $z_{i,j} = u_i v_j$ ($\forall i \in [1, m]$, $j \in [1, n]$), is the product of a random variable and a positive constant, and hence also a random variable.

It has been proven that the above matrix addition scheme is computationally indistinguishable in value as defined in Definition 1 [37]. Note that this scheme does not preserve the zeros in matrix $\mathbf{H}$ during the matrix transformation process.

2) MATRIX MULTIPLICATION

Consider a user's private matrix $\mathbf{H} \in \mathbb{R}^{m\times n}$, with non-zero elements $h_{i,j}$ for $(i, j) \in \mathcal{S}_\mathbf{H}$, where $\mathcal{S}_\mathbf{H}$ is the set of the coordinates of non-zero elements in $\mathbf{H}$. Then, the user can hide $\mathbf{H}$'s non-zero values by performing the following multiplications [35]:

$$\tilde{\mathbf{H}} = \mathbf{DHF} \tag{42}$$

where $\mathbf{D} \in \mathbb{R}^{m\times m}$ is a diagonal matrix with non-zero elements generated by a pseudorandom function, and $\mathbf{F} \in \mathbb{R}^{n\times n}$ is also a diagonal matrix but with non-zero elements set to arbitrary positive constants.

Salinas *et al.* [35] prove that this matrix multiplication scheme is computationally indistinguishable in value under a CPA as defined in Definition 2. Note that this scheme does preserve the zeros in matrix $\mathbf{H}$ during the matrix transformation process.

3) MATRIX PERMUTATION

Although the matrix transformation in equation (42) hides the values of the non-zero elements in $\mathbf{H}$, it reveals their original positions, i.e., $\mathbf{H}$'s structure, which are also private.

By randomly permuting the rows and columns of $\tilde{\mathbf{H}}$, the user can conceal $\mathbf{H}$'s structure. Specifically, the user performs the following multiplications [35]:

$$\hat{\mathbf{H}} = \mathbf{E}\tilde{\mathbf{H}}\mathbf{U} \tag{43}$$

where $\mathbf{E} \in \mathbb{R}^{m\times m}$ and $\mathbf{U} \in \mathbf{R}^{n\times n}$ are pseudorandom permutation matrices, and their elements are defined by

$$e_{i,j} = \delta_{\pi(i),j} \quad \forall i \in [1, m], j \in [1, m]$$
$$u_{i,j} = \delta_{\pi(i),j} \quad \forall i \in [1, n], j \in [1, n]$$

where i and j are the row and column indexes, respectively, and the function $\pi(\cdot)$ maps an original index i to its pseudorandomly permuted index, i.e., $\pi(i) = \hat{e}_i$ (for $i \in [1, m]$) and $\pi(i) = \hat{u}_i$ (for $i \in [1, n]$). Besides, the Kronecker delta function is given by

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j. \end{cases}$$

The user is able to recover the original structure by applying the inverse permutation, i.e.,

$$\hat{\mathbf{H}} = \mathbf{E}^\top\tilde{\mathbf{H}}\mathbf{U}^\top$$

where $\top$ denotes the matrix transpose operation. To reach this result, we have used the orthogonality property of the permutation matrices, i.e., $\mathbf{E}^\top\mathbf{E} = \mathbf{I}$ and $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Note that [35] shows that this matrix permutation scheme is computationally indistinguishable in structure under a CPA as defined in Definition 3.

Therefore, taking advantage of both matrix multiplication and matrix permutation, the user can hide the values of the non-zero elements in $\mathbf{H}$ as well as its structure as follows:

$$\hat{\mathbf{H}} = \mathbf{LHR} \tag{44}$$

where $\mathbf{L} = \mathbf{ED}$ and $\mathbf{R} = \mathbf{FU}$.

V. SECURE OUTSOURCING OF LARGE-SCALE COMPUTATIONS

In this section, we review how researchers have employed the data privacy preserving techniques in Section IV to securely outsource large-scale computations introduced in Section II.

A. LINEAR ALGEBRA

Lei *et al.* [42] construct a method for securely outsourcing matrix inversion described in (2). In particular, the user first conceals the values and structure of a private matrix $\mathbf{A}$ by computing matrix multiplications and permutations, i.e.,

$$\hat{\mathbf{A}} = \mathbf{MAN} \tag{45}$$

where $\mathbf{M}$ and $\mathbf{N}$ are essentially both the multiplication of a random diagonal matrix and a permutation matrix, and then uploads $\hat{\mathbf{A}}$ to the cloud. After computing the inverse matrix $\hat{\mathbf{A}}^{-1}$, the cloud returns the result to the user, who recovers $\mathbf{A}^{-1}$ as follows:

$$\mathbf{A}^{-1} = \mathbf{N}\hat{\mathbf{A}}^{-1}\mathbf{M}.$$

Thus, this secure matrix inversion outsourcing by Lei *et al.* can be used to solve LSEs of the form (1) as described in Section II-A1. Besides, Chen *et al.* [43] employ a similar procedure to solve linear regression problems. Note that neither of these schemes provides formal proof for privacy preservation.

Since it may still be a very challenging task for the cloud to solve LSEs in (1) with the matrix inversion method, researchers have proposed secure outsourcing algorithms for large-scale LSEs based on iterative procedures, which are introduced below.

Wang *et al.* [44] develop an algorithm based on the Jacobi iteration and partial homomorphic encryption, where the user and the cloud collaborate to solve an LSE. Specifically, the user computes matrix $\mathbf{T}$ as described in Section II-A2, encrypts its values using the Paillier cryptosystem described in Section IV-A, and uploads it to the cloud. To protect the

solution vector's values, the user replaces $\mathbf{x}$ with $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{r}$ in (1), where $\mathbf{r}$ is an $n \times 1$ random matrix.

The user and the cloud collaboratively carry out the Jacobi iteration (3) as follows. At the iteration step $k = 0$, the user computes the initial solution vector $\hat{\mathbf{x}}^0 = \mathbf{x}^0 + \mathbf{r}$ and sends it to the cloud, who then computes

$$E((\mathbf{T}\hat{\mathbf{x}}^0)_i) = \prod_{j=1}^{n} E(t_{i,j})^{\hat{\mathbf{x}}_j^0}$$

where $(\mathbf{T}\hat{\mathbf{x}}^0)_i$ is the ith element in $\mathbf{T}\hat{\mathbf{x}}^0$, and $t_{i,j}$ is the element in the ith row and jth column of $\mathbf{T}$. The user downloads $E((\mathbf{T}\hat{\mathbf{x}}^0)_i)$'s and finds $\hat{\mathbf{x}}^1$ by computing the rest of equation (3). At the end of the kth iteration, the user transmits $\hat{\mathbf{x}}^k$ to the cloud for the next iteration $k+1$. The iterations continue until the sequence of solution vectors meets the stopping criteria (4). Finally, to find the solution of (1), the user computes $\mathbf{x}^* = \hat{\mathbf{x}}^* - \mathbf{r}$.

Although encrypting the values of matrix $\mathbf{T}$ is a one-time operation, it is a computationally expensive task, especially in big data applications, and common users may lack resources to compute it. To overcome this problem, the user may employ a private cloud, e.g., a trusted computing infrastructure within its organization, to carry out the encryption. Besides, there is no formal proof for privacy preservation in [44] either.

To relax the assumption of a trusted private cloud and further reduce computational complexity, Salinas *et al.* [37] construct a secure outsourcing algorithm for large-scale LSEs that only employs perturbation-based approaches. In particular, the user generates a masked coefficient matrix $\hat{\mathbf{A}} = \mathbf{A}+\mathbf{Z}$ where $\mathbf{Z}$ is an $n \times n$ random matrix constructed as described in Section IV-B1, and sends it with the initial solution vector $\mathbf{x_0} = \mathbf{q}$ to the cloud, where $\mathbf{q}$ is a random $n \times 1$ vector. Then, the cloud computes the following secure version of (12):

$$\begin{aligned}\mathbf{h}_0 &= \hat{\mathbf{A}}\mathbf{x}_0 \\ &= (\mathbf{A} + \mathbf{Z})\mathbf{x}_0.\end{aligned}$$

Upon receiving $\mathbf{h}_0$, the user computes the residual vector as follows:

$$\begin{aligned}\mathbf{r}_0 &= \mathbf{A}\mathbf{x}_0 - \mathbf{b} \\ &= \mathbf{h}_0 - \mathbf{u}\left(\mathbf{v}^\top \mathbf{x}_0\right) - \mathbf{b}.\end{aligned}$$

At the end of the initialization step, the user sets the conjugate vector $\mathbf{p}_0 = -\mathbf{r}_0$, and transmits it with $\mathbf{r}_0$ to the cloud.

To protect the user's privacy, the cloud carries out computations with the transformed matrix $\hat{\mathbf{A}}$, instead of $\mathbf{A}$, as follows. To compute α_k, the user and the cloud carry out equation (15) in two steps. First, based on the $\mathbf{p}_k$ received from the user, the cloud computes an intermediate vector

$$t_k = \mathbf{p}_k^\top \hat{\mathbf{A}}\mathbf{p}_k = \mathbf{p}_k^\top (\mathbf{A} + \mathbf{Z})\mathbf{p}_k. \tag{46}$$

Second, the user finds α_k using t_k as follows

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{t_k - (\mathbf{p}_k^\top \mathbf{u})(\mathbf{v}^\top \mathbf{p}_k)}.$$

Similarly, the user exploits the cloud's resources to find $\mathbf{r}_{k+1}$. The cloud first calculates an intermediate vector:

$$\mathbf{f}_k = \hat{\mathbf{A}}\mathbf{p}_k = (\mathbf{A} + \mathbf{Z})\mathbf{p}_k \tag{47}$$

which allows the user to compute $\mathbf{r}_{k+1}$ as follows:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k\big(\mathbf{f}_k - \mathbf{u}(\mathbf{v}^\top \mathbf{p}_k)\big).$$

Since equations (17)-(19) only require vector-vector operations, they all can be computed efficiently by the user itself. At the end of the kth iteration, the user transmits $\mathbf{p}_{k+1}$ to the cloud for the next iteration $k+1$. When the condition (11) is met, the algorithm converges and the user recovers the solution vector $\mathbf{x}^*$. Noticeably, [37] proves that the proposed scheme is computationally indistinguishable in value as defined in Definition 1.

B. OPTIMIZATION

1) LINEAR PROGRAMS

To securely outsource an LP of the form in (20), Wang *et al.* [45] propose the following equivalent LP, i.e.,

$$\underset{\hat{\mathbf{x}}}{\text{minimize}} \quad \hat{z} = \hat{\mathbf{c}}^T \hat{\mathbf{x}} \tag{48a}$$

$$\text{subject to} \quad \hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}} \tag{48b}$$

$$\hat{\mathbf{B}}\hat{\mathbf{x}} \geq \mathbf{0} \tag{48c}$$

where $\hat{\mathbf{x}} = \mathbf{N}^{-1}(\mathbf{x} + \mathbf{r})$ and $\mathbf{r}$ is a random $n \times 1$ vector, $\hat{\mathbf{c}} = \gamma \mathbf{N}^\top \mathbf{c}$ (γ is a random number), $\hat{\mathbf{A}} = \mathbf{MAN}$ with $\mathbf{M}$ and $\mathbf{N}$ being random dense matrices, and $\hat{\mathbf{b}} = \mathbf{M}(\mathbf{b} + \mathbf{Ar})$ (with $\mathbf{b} + \mathbf{Ar} \neq \mathbf{0}$). Note that the bound constraint is $\mathbf{Bx} \geq \mathbf{0}$ instead of $\mathbf{x} \geq \mathbf{0}$ in (20c). The authors propose to find a matrix $\boldsymbol{\lambda}$ such that $\boldsymbol{\lambda}\hat{\mathbf{b}} = \mathbf{Br}$, and set $\hat{\mathbf{B}} = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{MA})\mathbf{N}$ for $|\hat{\mathbf{B}}| \neq 0$.

The user can outsource the transformed problem (48a) to the cloud. The cloud then solves (48a) and its Lagrange dual problem, and sends the results to the user, who computes $\mathbf{x}^* = \mathbf{N}\hat{\mathbf{x}}^* - \mathbf{r}$. To verify the correctness of the cloud's computations, the user can check if the objective of (48a) is equal to that of the Lagrange dual problem. No formal proof for privacy preservation is provided. Besides, Wang *et al.* [46] and Nie *et al.* [47] propose similar techniques to securely outsource LPs.

2) QUADRATIC PROGRAMS

To securely outsource a QP of the form in (21), Zhou and Li derive an equivalent QP as follows:

$$\min_{\hat{\mathbf{x}}} \quad \hat{f}(\mathbf{x}) = \frac{1}{2}\hat{\mathbf{x}}^\top \hat{\mathbf{Q}}\hat{\mathbf{x}} - \hat{\mathbf{b}}^\top \hat{\mathbf{x}} \tag{49a}$$

$$\text{subject to} \quad \hat{\mathbf{A}}\hat{\mathbf{x}} \leq \hat{\mathbf{c}} \tag{49b}$$

where $\hat{\mathbf{x}} = \mathbf{N}^{-1}(\mathbf{x} + \mathbf{r})$ is the secure optimization variable, $\mathbf{r}$ is a random $n \times 1$ vector, $\hat{\mathbf{Q}} = \mathbf{N}^\top \mathbf{QN}$, $\hat{\mathbf{b}} = (\mathbf{r}^\top \mathbf{QN} + \mathbf{b}^\top \mathbf{N})^\top$, $\hat{\mathbf{A}} = \mathbf{MAN}$, and $\hat{\mathbf{c}} = \mathbf{M}(\mathbf{b} + \mathbf{Ar})$ (with $\mathbf{b} + \mathbf{Ar} \neq \mathbf{0}$). Matrices $\mathbf{M}$ and $\mathbf{N}$ are random matrices that can be constructed as in equation (44). After receiving the transformed

problem (49), the cloud solves it and returns the solution $\hat{\mathbf{x}}^*$ to the user. The user can find the solution to (21) by computing $\mathbf{x}^* = \mathbf{N}\hat{\mathbf{x}}^* - \mathbf{r}$. In addition, the user can verify the correctness of $\mathbf{x}^*$ by verifying if the Karush-Khun-Tucker conditions hold. Note that this scheme does not provide formal privacy proof.

Salinas *et al.* [35] design a secure outsourcing algorithm by first finding the dual problem of (21), and then concealing the dual problem's matrices via the perturbation-based approaches in Section IV-B. Specifically, in collaboration with the cloud, the user first computes $\mathbf{P}$ and $\mathbf{r}$ in (33) and then conceals them as follows:

$$\hat{\mathbf{P}} = \mathbf{L}\mathbf{R}_p\mathbf{P}\mathbf{L}^{-1} \quad (50)$$

$$\hat{\mathbf{r}} = \mathbf{L}\mathbf{R}_p\mathbf{r} \quad (51)$$

$$\hat{\boldsymbol{\lambda}} = \mathbf{L}\boldsymbol{\lambda}_0 \quad (52)$$

where $\mathbf{L}$ is the same as that in (44), $\mathbf{R}_p > \mathbf{0}$ is a diagonal matrix whose elements are generated by a pseudorandom function, and the elements of $\boldsymbol{\lambda}_0$ are non-negative which satisfies the constraints of (21). After the cloud receives the above matrix and vector, it carries out the Gauss-Seidel iteration in (36) as follows:

$$\hat{\lambda}_k(t+1) = \max\{0, \hat{\lambda}_k(t) - \frac{1}{\hat{p}_{k,k}}(\hat{r}_k + \hat{\mathbf{p}}_k\hat{\boldsymbol{\lambda}}(t))\}$$

where $\hat{\lambda}_k$ and $\hat{r}_k$ are the kth elements of $\hat{\boldsymbol{\lambda}}$ and $\hat{\mathbf{r}}$, respectively, $\hat{\mathbf{p}}_k$ is the kth row of $\hat{\mathbf{P}}$, and the iteration step is denoted by t. To allow the cloud to enforce the non-negativity constraints in (37), the user also shares the signs of the elements in $\mathbf{L}$. The iterations continue until the stopping criteria in (38) is satisfied.

The user can find the solution to the original QP problem by first computing $\boldsymbol{\lambda}^* = \mathbf{L}^{-1}\hat{\boldsymbol{\lambda}}^*$ and then $\mathbf{x}^* = \mathbf{Q}^{-1}(\mathbf{b} - \mathbf{A}\boldsymbol{\lambda}^*)$ where $\mathbf{Q}^{-1}$ can be efficiently calculated with the help of the cloud. Furthermore, the authors have developed a parallel computing algorithm that can enable the cloud to complete the Gauss-Seidel iterations in a parallel manner. Finally, the user verifies the correctness of the computation by checking whether the Karun-Khun-Tucker optimality conditions hold.

Moreover, the authors formally prove that the proposed scheme is CPA-secure both in value and in structure as defined in Definition 2 and Definition 3, respectively.

C. SECURITY VALIDATION

In addition to constructing secure outsourcing algorithms, previous works describe the security properties of their schemes with respect to the threat models presented in Section III-B. Specifically, in works that are based on homomorphic encryption such as [44], the authors base the security properties of their algorithms on the employed cryptosystem. To show the security of outsourcing schemes based on perturbation techniques, the works in [42], [45], and [48] briefly describe how an attacker would need an infeasible amount of computing resources and time to extract information from the user's outsourced matrices and vectors, but no formal proof for privacy is given. The works in [35] and [37] state theorems about the security of their outsourcing algorithms and formally show that they are computationally indistinguishable.

D. COMPLEXITY ANALYSIS

To preserve the computational gains offered by cloud computing, secure outsourcing schemes require the user to perform computing tasks that are less expensive than solving the original computations. In this section, we review the computational complexity of solving linear algebra and optimization problems, and report the user's computational complexity in the secure outsourcing algorithms.

We define the computational complexity as the number of floating-point (flops) operations (additions, subtractions, multiplications, and divisions), bitwise operations, and encryptions that the party needs to perform. We note that an encryption takes many flops, and a flop takes some bitwise operations.

Matrix inversion has complexity $\mathcal{O}(n^\rho)$ for $2 < \rho < 3$, given an $n \times n$ matrix. In the secure outsourcing scheme for matrix inversion by Lei *et al.* [42], the user computes sparse matrix multiplications, which have complexity $\mathcal{O}(n^2)$.

In general, solving an LSE takes complexity $\mathcal{O}(n^3)$. In the scheme by Wang *et al.* [44], the user only performs vector computations with complexity $\mathcal{O}(n)$ and outsources $\mathcal{O}(n^2)$ Paillier encryptions to a private cloud. The secure outsourcing algorithm by Salinas *et al.* [37] requires the user to perform operations with complexity $\mathcal{O}(n^2)$, and do not need the help of a private cloud.

Solving LPs and QPs requires $\mathcal{O}(n^3)$ computing operations. The algorithm for secure outsourcing of linear programs presented by Wang *et al.* [45] requires the user to perform operations with complexity $\mathcal{O}(n^\rho)$ for $2 < \rho < 3$. The user's complexity when securely outsourcing QPs using the algorithms proposed by Zhou *et al.* [48] and Salinas *et al.* [35] is $\mathcal{O}(n^2)$.

VI. CONCLUSIONS

Scientists and engineers have the opportunity to revolutionize their fields of study by analyzing the massive data collected by today's society. To analyze such large-scale data sets, cloud computing has been proposed as a cost-effective and practical computing paradigm. However, since the data carries private information, it should be kept secret from the cloud and external attackers for ethical, security, or legal reasons. Moreover, we notice that many large-scale data analysis techniques are based on two most fundamental computational problems, i.e., linear algebra and optimization. In this tutorial, we have described the most frequently employed linear algebra and optimization computations in large-scale data analysis. We have presented an extensive tutorial of privacy-preserving techniques, and how they are used to securely outsource large-scale computations.

REFERENCES

[1] J. F. Gantz, D. Reinsel, S. Minton, and V. Turner. (2014). The digital universe of opportunities: Rich data and the increasing value of the Internet of Things. IDC. [Online]. Available: http://idcdocserv.com/1678

[2] National Research Council, *Frontiers in Massive Data Analysis*. Washington, DC, USA: The National Academies Press, 2013.

[3] Q. Hardy. (2015). *IBM, G.E. and Others Create Big Data Alliance*. [Online]. Available: http://bits.blogs.nytimes.com/2015/02/17/ibm-g-e-and-others-create-big-data-alliance/

[4] *Big Data: Seizing Opportunities, Preserving Values*. [Online]. Available: https://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf

[5] A. McAfee and E. Brynjolfsson, *Big Data: The Management Revolution*. Boston, MA, USA: Harvard Business Review, Oct. 2012.

[6] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[7] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power Energy Mag.*, vol. 7, no. 2, pp. 52–62, Mar./Apr. 2009.

[8] President's Council of Advisors on Science and Technology. (May 2014). *Big Data and Privacy: A Technological Perspective*. [Online]. Available: http://www.whitehouse.gov/sites/default/files/microsites/ostp/PCAST/pcast_big_data_and_privacy_-_may_2014.pdf

[9] T. Kraska, "Finding the needle in the big data systems haystack," *IEEE Internet Comput.*, vol. 17, no. 1, pp. 84–86, Jan./Feb. 2013.

[10] Y. Simmhan *et al.*, "Cloud-based software platform for big data analytics in smart grids," *Computing Sci. Eng.*, vol. 15, no. 4, pp. 38–47, Jul./Aug. 2013.

[11] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan, "Computational solutions to large-scale data management and analysis," *Nature Rev. Genet.*, vol. 11, no. 9, pp. 647–657, Sep. 2010.

[12] H. Demirkan and D. Delen, "Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud," *Decision Support Syst.*, vol. 55, no. 1, pp. 412–421, 2013.

[13] E. Kohlwey, A. Sussman, J. Trost, and A. Maurer, "Leveraging the cloud for big data biometrics: Meeting the performance requirements of the next generation biometric systems," in *Proc. IEEE World Congr. Services (SERVICES)*, Washington, DC, USA, Jul. 2011, pp. 597–601.

[14] U. Kang, D. H. Chau, and C. Faloutsos, "Pegasus: Mining billion-scale graphs in the cloud," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Kyoto, Japan, Mar. 2012, pp. 5341–5344.

[15] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 311–336, Mar. 2011.

[16] Amazon Web Services. (2015). *Netflix Case Study*. [Online]. Available: https://aws.amazon.com/solutions/case-studies/netflix/

[17] Cloud Security Allliance. (2011). *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0*. [Online]. Available: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

[18] A. Gumbus and F. Grodzinsky, "Era of big data: Danger of descrimination," *SIGCAS Comput. Soc.*, vol. 45, no. 3, pp. 118–125, Sep. 2015.

[19] H. K. Patil and R. Seshadri, "Big data security and privacy issues in healthcare," in *Proc. IEEE Int. Congr. Big Data*, Anchorage, AK, USA, Jun./Jul. 2014, pp. 762–765.

[20] D. Eckhoff and C. Sommer, "Driving for big data? Privacy concerns in vehicular networking," *IEEE Security Privacy*, vol. 12, no. 1, pp. 77–79, Jan. 2014.

[21] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2008, pp. 111–125.

[22] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proc. ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, 2009, pp. 21–32.

[23] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, Nov. 2009, pp. 199–212.

[24] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.

[25] D. Perez-Botero, J. Szefer, and R. B. Lee, "Characterizing hypervisor vulnerabilities in cloud computing servers," in *Proc. Int. Workshop Secur. Cloud Comput.*, Hangzhou, China, 2013, pp. 3–10.

[26] R. Miao, R. Potharaju, M. Yu, and N. Jain, "The dark menace: Characterizing network-based attacks in the cloud," in *Proc. Conf. Internet Meas.*, Tokyo, Japan, 2015, pp. 169–182.

[27] A. Duncan, S. Creese, M. Goldsmith, and J. S. Quinton, "Cloud computing: Insider attacks on virtual machines during migration," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Melbourne, VIC, Australia, Jul. 2013, pp. 493–500.

[28] S. F. F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," Mitsubishi Electr. Res. Lab., Cambridge, MA, USA, Tech. Rep. TR-97-19, 1997.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[30] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[31] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[32] A. Monticelli, *State Estimation in Electric Power Systems: A Generalized Approach*. Norwell, MA, USA: Kluwer, 1999.

[33] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*. New York, NY, USA: Springer, 2007.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[35] S. Salinas, C. Luo, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale quadratic programs," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Xi'an, China, 2016.

[36] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. London, U.K.: Chapman & Hall, 2008.

[37] S. Salinas, C. Luo, X. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr./May 2015, pp. 1035–1043.

[38] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. 30th Annu. Conf. Adv. Cryptol. (CRYPTO)*, Santa Barbara, CA, USA, 2010, pp. 465–482.

[39] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, Santa Barbara, CA, USA, 2010, pp. 483–501.

[40] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[41] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. ACM Symp. Theory Comput.*, San Francisco, CA, USA, 1982, pp. 365–377.

[42] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, p. 1, Jan./Jun. 2013.

[43] F. Chen, T. Xiang, X. Lei, and J. Chen, "Highly efficient linear regression outsourcing to a cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 499–508, Oct. 2014.

[44] C. Wang, Q. Wang, K. Ren, and J. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, Jun. 2013.

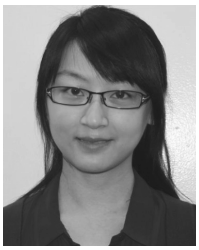
[45] C. Wang, K. Ren, and J. Wang, "Secure optimization computation outsourcing in cloud computing: A case study of linear programming," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 216–229, Jan. 2016.

[46] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, "Privacy-assured outsourcing of image reconstruction service in cloud," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 1, pp. 166–177, Jun. 2013.

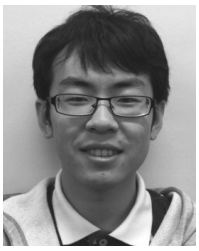
[47] H. Nie, X. Chen, J. Li, J. Liu, and W. Lou, "Efficient and verifiable algorithm for secure outsourcing of large-scale linear programming," in *Proc. IEEE 28th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Victoria, BC, Canada, May 2014, pp. 591–596.

[48] L. Zhou and C. Li, "Outsourcing large-scale quadratic programming to a public cloud," *IEEE Access*, vol. 3, pp. 2581–2589, Dec. 2015.

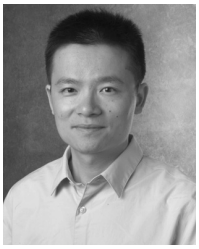
SERGIO SALINAS (M'09) received the B.S. degree in telecommunications engineering from Jackson State University, Jackson, in 2010, and the Ph.D. degree in electrical and computer engineering from Mississippi State University, Starkville, MS, in 2015. He is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS. His research interests include security and privacy in cyberphysical systems, cloud computing, and big data.



XUHUI CHEN (S'14) received the B.E. degree in information engineering from Xidian University, China, in 2012, and the M.E. degree in electrical and computer engineering from Mississippi State University, Starkville, in 2015. She is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Case Western Reserve University. Her research interests include big data, and mobile cloud computing.



JINLONG JI (S'16) received the B.E. and M.E. degrees from Xidian University, China, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Case Western Reserve University. His research interests include cybersecurity, big data computing, and smart health systems.



PAN LI (S'06–M'09) received the B.E. degree in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in 2009. Since Fall 2015, he has been with the Department of Electrical Engineering and Computer Science, Case Western Reserve University. He was an Assistant Professor with the Department of Electrical and Computer Engineering, Mississippi State University, in 2009 and 2015. His research interests include network science and economics, energy systems, security and privacy, and big data. He has served as an Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS–COGNITIVE RADIO SERIES and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, a Feature Editor of the IEEE WIRELESS COMMUNICATIONS, and a Technical Program Committee Co-Chair for Ad-hoc, Mesh, Machine-to-Machine and Sensor Networks Track, IEE VTC 2014, Physical Layer Track, Wireless Communications Symposium, WTS 2014, the Wireless Networking Symposium, and the IEEE ICC 2013. He received the NSF CAREER Award in 2012 and is a member of ACM.