2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems

Efficient Privacy-preserving Outsourcing of Large-scale Convex Separable Programming for Smart Cities

Weixian Liao, Wei Du
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, OH, USA
wxl393, wxd108@case.edu

Sergio Salinas
Department of Electrical Engineering and Computer Science
Wichita State University
Wichita, KS, USA
salinas@cs.wichita.edu

Pan Li
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, OH, USA
lipan@case.edu

***Abstract*—One of the most salient features of smart city is to utilize big data to make our lives more convenient and more intelligent. This is usually achieved through solving a series of large-scale common and fundamental problems such as linear systems of equations, linear programs, etc. However, it is a very challenging task for resource-limited clients and small companies to solve such problems as the data volume keeps increasing. With cloud computing, an alternative is to solve complex problems by outsourcing them to the cloud. Nonetheless, data privacy is one of the main concerns. Many previous works on privacy-preserving outsourcing are based on cryptographic techniques like homomorphic encryption and have very high computational complexity, which may not be practical for big data applications. In this paper, we design an efficient privacy-preserving outsourcing algorithm based on arithmetic operations only for large-scale convex separable programming problems. Specifically, we first develop an efficient transformation scheme to preserve data privacy. Then we linearize the convex functions with arbitrary accuracy and solve the problem by outsourcing it to the cloud. The client can efficiently verify the correctness of the returned results to prevent any malicious behavior of the cloud. Implementations on Amazon Elastic Compute Cloud (EC2) platform show that the proposed scheme provides significant time savings.**

***Keywords*-convex separable programming; cloud computing; privacy; smart city;**

I. Introduction

Smart cities play a key role in transforming people's lives by improving sectors including transportation, water, power system, healthcare, education, etc. [1]. Many technologies that enable such innovative services employ big data analytics. In particular, we have witnessed a skyrocketing volume of data generated by a great variety of sources, such as online transactions, social networking activities, healthcare records, emails and so forth. Analyzing such data has brought far-reaching impacts on smart cities. For example, comprehensive and insightful analysis of the intensive physiological data from patients can effectively assist physicians in diagnosing diseases [2]. Manufacturers integrate big data analytics into every stage of their business processes, including predicting next possible popular products, optimizing product prices against competitors, forecasting market demands and formulating production plans so as to achieve maximum profits [3]. Besides, in social networks, monitoring and analyzing interactions among users can improve the accuracy of targeted advertising and recommendation systems [4].

Although big data analytics can bring tremendous opportunities and profits, it is difficult for customers and small companies to compute and analyze huge volumes of data by themselves, which requires extensive computing resources and hence very expensive capital investments [5]. As a result, it is in dire need to find effective approaches to analyze large-scale data sets in a more efficient and economical way. Fortunately, cloud computing, characterized by rich computation resources and the pay-per-use paradigm, can help resource-limited clients perform large-scale data computation and analytics [6]–[8]. In particular, clients can offload heavy computation tasks to the cloud and enjoy vast computation resources in a cost-effective manner. It has become widely utilized in various types of environments and supported clients to solve pressing issues more timely. For example, a financial corporation's proprietary trading algorithm can involve intensive computation for realtime high frequency trading. It is very impractical for the company to run it with limited computational resources which leads to delayed response to the market and may cause inestimable losses. However, expeditious cloud outsourcing and computing can be an effective alternative to address this issue [9]. To give another example, smart grid companies can also choose to outsource complex power distribution schemes to the cloud, which can save noticeable computational resources and improve energy efficiency.

In spite of the enormous benefits brought by the cloud, there exist some serious concerns from clients, one of which is data privacy. Clients' data often contains sensitive information, such as individuals' medical records, companies' proprietary information, engineering and scientific models, etc. Since the leakage of such information may cause serious problems, a good way of protecting clients' data is to let the clients send concealed data instead of real data to the cloud. The second issue is the verifiability of the results returned by the cloud. It is possible that the cloud may unintentionally or intentionally return invalid results. For example, if the software incurs some hardware failures or expensive cost during the operation, a malicious cloud may send incorrect results to the client. Consequently, a privacy-preserving outsourcing protocol should be developed in a manner that enables the client to check the correctness of the returned results. The last challenge is the computational efficiency. The additional burden incurred by the privacy-preserving scheme should be as little as possible. Otherwise there will be no incentive for the client to seek help from the cloud. To overcome those challenges, there have been some existing schemes that are designed and applied to encrypt and outsource basic mathematical problems. For example, our previous works in [5], [10], [11] study the secure outsourcing of linear systems of equations and quadratic programming problems. Zhou *et al.* provide a privacy-preserving outsourcing tool that focuses on a quadratic programming problem [12]. Outsourcing methods for modular exponentiation, image reconstruction and linear regression are also reported in [13]–[15], respectively. Moreover, there are privacy-preserving outsourcing schemes for matrix operations, including matrix inversion [16], matrix determinant [17], and matrix multiplication [18].

However, the privacy-preserving outsourcing of large-scale convex separable programming (CSP) has not been studied before, which is one of the most common and fundamental nonlinear programming problem in many engineering and scientific computations. Particularly, many real-world applications are essentially CSPs, such as cost optimization, industry control, and resource allocation [19], which are critical for smart cities. To this end, in this paper we propose an efficient privacy-preserving outsourcing algorithm for solving large-scale CSPs. Specifically, we consider a CSP where the objective function and constraints are composed of convex functions. Firstly we develop an efficient transformation scheme to preserve a vector's privacy. Due to the characteristics of CSPs, we are able to linearize the convex functions with arbitrary accuracy, which results in solving a series of privacy-preserving large-scale linear programming problems in the cloud. To ensure the returned results' integrity, we adopt a light-weight scheme to effectively verify the correctness of the final results. Our main contributions in this paper are summarized as follows:

- We develop a privacy-preserving scheme to efficiently outsource large-scale CSPs. To the best of our knowledge, this is the first study in the literature to investigate this problem.
- Our privacy-preserving scheme is based on very efficient arithmetic operations instead of heavy computations like homomorphic encryptions.
- We show that the proposed algorithm protects the privacy of the client's data and the final results.
- Experimental results show that the proposed algorithm achieves noticeable time savings.

The rest of the paper is organized as the follows. Section II introduces the system architecture, threat model, and privacy definition. In Section III, we propose a privacy-preserving transformation algorithm to protect a vector's privacy. Section IV presents an efficient privacy-preserving algorithm to solve the transformed linearized CSP problem. In Section V, we evaluate the performance of the proposed algorithm through implementations on the Amazon Elastic Compute Cloud (EC2) platform and finally conclude the paper in Section VI.

II. Problem Formulation

In this section, we introduce our system architecture, the threat model, and privacy definition.

A. System Architecture

We consider a two-party computing architecture for large-scale CSP problems as shown in Fig. 1, where a client has a resource-limited computing device and a remote cloud server has abundant computing capabilities. The client tries to solve a large-scale CSP problem with the help of the cloud by outsourcing the most computationally complex tasks to the cloud to find the optimal solution while preserving his/her privacy. A large-scale CSP problem can be formulated as follows:

$$\textbf{P1:} \qquad \textbf{Min} \quad F = \sum_{j=1}^{n} f_j(x_j),$$

$$\textbf{s.t.} \qquad \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i, i = 1, \cdots, m \qquad (1)$$

$$x_{jL} \leq x_j \leq x_{jU}, j = 1, \cdots, n \qquad (2)$$

where F is a nonlinear separable function. $f_j(x_j)'s$ and $g_{ij}(x_j)'s$ are general convex functions. $b_i's$ are constants. x_{jL} and x_{jU} are lower and upper bounds for x_j. Problem **P1** is said to be a separable programming (SP) problem because all the variables, i.e., $x_j, j \in [1, n]$, are mathematically independent in the objective and constraint functions [20].

CSPs [19] are a special class of optimization problems, which arise frequently in practical applications such as time-dependent optimization in smart city applications. For example, in an industrial resource utilization problem, each variable x_j represents the resource utilization in the time period j, and the results of the resource utilization or profits are additive over time. Thus, this problem can be formulated as a CSP problem where decision variables are subject to practical constraint functions. Another example is smart grid operations in smart cities. Particularly, the objective function can be minimizing zero minus the revenue of a big company with regards to monthly energy consumptions

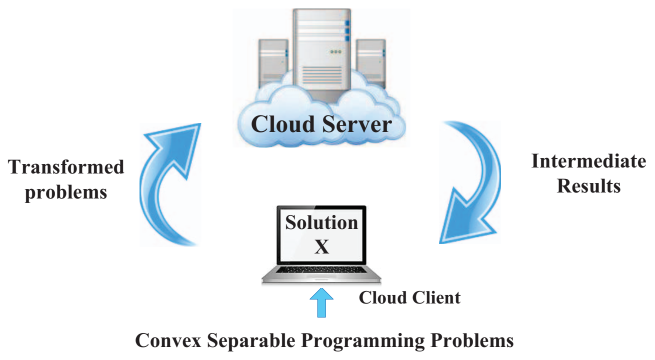


Figure 1. The architecture for privacy-preserving outsourcing of large-scale CSPs

at different sub-companies, while the total energy cost at different months are upper-bounded. Obviously, this problem can be formulated as a CSP as well.

B. Threat Model

We consider that the cloud server is malicious and knows the proposed secure outsourcing algorithm. Specifically, the cloud tries to extract knowledge from the client's data and the final results. The cloud may even try to deviate from the proposed protocols and return erroneous results so as to save computing resources. We also consider that in the problem **P1**, the objective functions $f_j(x_j)$'s and the constraint functions $g_{ij}(x_j)$'s all contain sensitive information that should not be revealed to the cloud. The optimal solution x_j's and the optimal value optimal value of the objective function should not be known by the cloud either.

C. Privacy Definition

In this study, we adopt the definition of computational indistinguishability [21] in our privacy-preserving outsourcing scheme design. We first introduce the definition of probability ensemble as follows.

Definition 1. Probability Ensemble: Let $\mathcal{I}$ be a countable set. A probability ensemble by $\mathcal{I}$ is a collection of random variables denoted by $\{X_i\}_{i\in\mathcal{I}}$.

$\mathcal{I}$ can be either a set of natural numbers, i.e., $\mathbb{N}$, or an efficiently computable subset $\{0,1\}^n$. With Definition 1, we can formally give the definition of computational indistinguishability between two ensembles.

Definition 2. Computational Indistinguishability: Let $\{X_n\}_{n\in\mathbb{N}}$ be a probability ensemble with entries following a uniform distribution on $[-R, R]$. Another probability ensemble $\{Y_n\}_{n\in\mathbb{N}}$ is said to be computationally indistinguishable from $\{X_n\}_{n\in\mathbb{N}}$ if for every probabilistic polynomial-time distinguisher D, there exists a negligible function, i.e., $\mu(\cdot)$ such that

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| \le \mu(\cdot) \tag{3}$$

where distinguisher $D(\cdot)$ outputs 1 when it identifies that the input x (or y) does not follow a uniform distribution, and zero otherwise.

III. A Privacy-preserving Transformation Scheme for Vector Privacy

Before we delve into the details of the privacy-preserving outsourcing algorithm for the CSP problem, we first develop a privacy-preserving vector transformation scheme.

Specifically, the client needs to conduct transformation on the original data before delegating the complex computations to the cloud. The transformation should: 1) conceal the data from the cloud, 2) impose light computational burden on the client, 3) allow the cloud to return verifiable results. Therefore, we design a light-weight privacy-preserving transformation scheme based on addition and multiplication operations only, which ensures the computational indistinguishability in Definition 2.

In particular, the client hides the private variable vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ by adding a randomly generated vector $\mathbf{r} \in \mathbb{R}^{n\times 1}$ as follows:

$$\mathbf{y} = \mathbf{x} + \mathbf{r} \tag{4}$$

where $y_j = x_j + r_j$ for any $j \in [1, n]$, and y_j, x_j and r_j are the jth element of vector $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{r}$, respectively. Here we assume that x_j is in the range $[-K, K]$ where $K = 2^k (k > 0)$ is a positive constant. In addition, r_j $(j \in [1, n])$ is uniformly distributed on $[-L, L]$ with the corresponding probability density function as follows:

$$f_r(r_j) = \begin{cases} \frac{1}{2L}, & -L \le r_j \le L \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $L = 2^{k+l} (l > 0)$ is a positive constant. We obtain the following theorem that vectors $\mathbf{r}$ and $\mathbf{y}$ are computationally indistinguishable.

Theorem 1. Let $\mathbf{r}^{n\times 1}$ be a random vector with its elements uniformly distributed in the interval $[-L, L]$. Then vectors $\mathbf{r}$ and $\mathbf{y} = \mathbf{x} + \mathbf{r}$ are computationally indistinguishable.

Proof: According to Definition 2, we need to prove that any polynomial-time distinguisher D cannot distinguish y_j from r_j for $j \in [1, n]$ except with non-negligible success probability, where y_j and r_j are the jth element of the vector $\mathbf{y}$ and $\mathbf{r}$ respectively. The best strategy for a polynomial-time distinguisher D is to follow the rules: D outputs 0 or 1 with the same probability of $\frac{1}{2}$ if the chosen element, i.e., y_j, is within the range $[-L, L]$, and outputs 1 if y_j is in the range $(-\infty, -L)$ or (L, ∞).

Therefore, the success probability of the distinguisher when the input is $y_j = x_j + r_j$ can be calculated as:

$$\begin{aligned} & Pr[D(y_j) = 1] \\ = & \ \frac{1}{2}Pr[-L \le x_j + r_j \le L] \\ + & \ Pr[x_j + r_j < -L] + Pr[x_j + r_j > L] \\ = & \ \frac{1}{2}\big(1 - Pr[x_j + r_j < -L] - Pr[x_j + r_j > L]\big) \\ + & \ Pr[x_j + r_j < -L] + Pr[x_j + r_j > L] \end{aligned} \quad (6)$$

Recall that x_j is in the range $[-K, K]$ and that r_j is sampled from a uniform distribution with probability density function in (5). We have that

$$\begin{aligned} & Pr[x_j + r_j > L] \\ & = Pr[r_j > L - x_j] \le Pr[r_j > L - K] = \frac{K}{2L}. \end{aligned} \quad (7)$$

Similarly, we can have that

$$\begin{aligned} & Pr[x_j + r_j < -L] = Pr[r_j < -L - x_j] \\ & \le Pr[r_j < -L + K] = \frac{K}{2L}. \end{aligned} \quad (8)$$

Consequently, the success probability of the distinguisher D follows the inequality:

$$Pr[D(y_j) = 1] \le \frac{1}{2} + \frac{K}{2L}. \quad (9)$$

On the other hand, when the input is r_j, following the procedures above we can obtain that:

$$Pr[D(r_j) = 1] = \frac{1}{2}. \quad (10)$$

According to equation 3 in Definition 2, we get for all $j \in [1, n]$:

$$|Pr[D(y_j) = 1] - Pr[D(r_j) = 1]| \le \frac{K}{2L}. \quad (11)$$

Note that $K = 2^k$ and $L = 2^{k+l}$. Thus, we can obtain that

$$\mu(l) = \frac{K}{2L} \le \frac{2^k}{2^{k+l+1}} = \frac{1}{2^{l+1}}, \quad (12)$$

which is negligible when l is large. ■

IV. Privacy-preserving Outsourcing Scheme Design

In this section, we develop a privacy-preserving outsourcing scheme for large-scale CSPs. Note that the original CSP problem **P1** is a nonlinear problem. Our main idea is to firstly linearize the nonlinear functions in **P1** with arbitrary accuracy and obtain a series of linear programming problems denoted by **P2**. After that, we propose the privacy-preserving outsourcing scheme for solving the large-scale CSP problem.

A. Linearization of a General Nonlinear Function

Consider a general continuous nonlinear function $h(t)$ where $t \in [t_a, t_b]$. We use a linear approximation function, i.e., $\hat{h}(\cdot)$, to approximate the original function $h(t)$. Specifically, by inserting k grid points, denoted by $\{t_v | v = 1, \cdots, k\}$, a continuous nonlinear function $h(t)$ can be approximated by [19]:

$$\text{for} \quad t = \sum_{v=1}^{k} t_v \lambda_v, \quad h(t) \approx \sum_{v=1}^{k} h(t_v)\lambda_v \quad (13)$$

where $\lambda_v (v \in [1, k])$ is the coefficient for the grid point t_v, and

$$\sum_{v=1}^{k} \lambda_v = 1, \lambda_v \ge 0, v = 1, \cdots, k \quad (14)$$

B. Linearization of Problem *P1*

Based on the linearization method presented in (13) and (14), we can transform the original problem **P1** into **P2**, i.e.,

$$\begin{aligned} \textbf{P2:} \quad & \min_{\{\lambda_{jv} | j \in [1,n], v \in [1,k_j]\}} \quad \sum_{j=1}^{n} \sum_{v=1}^{k_j} f_j(\hat{x}_{jv})\lambda_{jv}, \\ \textbf{s.t.} \quad & \sum_{j=1}^{n} \sum_{v=1}^{k_j} g_{ij}(\hat{x}_{jv})\lambda_{jv} \le b_i, i = 1, \cdots, m \quad (15) \\ & \sum_{v=1}^{k_j} \lambda_{jv} = 1, j = 1, \cdots, n \quad (16) \\ & \lambda_{jv} \ge 0, v = 1, \cdots, k_j, j = 1, \cdots, n \quad (17) \end{aligned}$$

where k_j is the number of grid points for the variable x_j, and x_{jv}'s ($v \in [1, k_j]$) are the grid points for the variable x_j. Since **P2** is a linear programming problem, we can solve it with existing techniques such as interior point method [22].

C. Optimal Solver for Original Large-scale CSP Problem

Since the accuracy of the linear approximations heavily depends on the number of grid points for each variable, there is a tradeoff between the accuracy and the convergence speed. That is, when we increase the number of the grid points to improve the approximation accuracy, the size of the approximation problem **P2** increases dramatically. Considering that the problem is already a large-scale problem, how to optimally choose the number of grid points is very critical. In what follows, we describe how to find the optimal number of grid points.

We solve this problem in an iterative manner. Assume that at the dth iteration, we solve the **P2** and let $\hat{\lambda}_{jv}$'s be the optimal solution to **P2**. Furthermore, let $s_i \ge 0$ and t_j denote the optimal Lagrangian multipliers for constraints (15) and (16), respectively. Then the solution set can be denoted by $\Omega = \{\hat{\lambda}_{jv}, s_i, t_j | i \in [1, m], j \in [1, n], v \in [1, k_j]\}$. Next, the question is whether adding a new grid point can achieve a better approximation and the minimum objective function value would further decrease. Therefore, we have the following theorem.

Theorem 2. Let $\Omega = \{\hat{\lambda}_{jv}, s_i, t_j | i \in [1, m], j \in [1, n], v \in [1, k_j]\}$ be the solution set to the problem **P2** and $\hat{x}_{jv}$'s, $(v \in [1, k, j = 1, \cdots, n)$ be the corresponding grid points. Consider that functions f_j and g_{ij} are convex functions. Denote by $\psi_j(\hat{x}_j)$ a function as follows:

$$\psi_j(\hat{x}_j) = f_j(\hat{x}_j) + \sum_{i=1}^{m} s_i g_{ij}(\hat{x}_j) + t_j \tag{18}$$

for $j = 1, \cdots, n$, where $\hat{x}_j = \sum_{v=1}^{k_j} \hat{x}_{jv} \hat{\lambda}_{jv}$. Then we have

1) If $\forall j = 1, \cdots, n, \psi_j(\hat{x}_j) \geq 0$, then $\Omega = \{\hat{\lambda}_{jv}, s_i, t_j\}, i = 1, \cdots, m, j = 1, \cdots, n$ is an optimal solution to problem **P1**, and the optimal objective function value is $\sum_{j=1}^{n} f_j(\hat{x}_j)$.
2) Otherwise, if $\psi_j(\hat{x}_j) \leq 0$, denote by $\hat{x}_{j(k_j+1)} = \sum_{v=1}^{k_j} \hat{x}_{jv} \hat{\lambda}_{jv}$ a new grid point. Then we will obtain a new approximating linear programming problem with a minimum objective value no higher than the current $\sum_{j=1}^{n} f_j(\hat{x}_j)$.

Proof: Due to the space limits, we omit the proof and please refer to [20] for detailed proof. ∎

Theorem 2 helps us determine the optimal number of grid points to find the final solution. The whole algorithm for solving the SP problem is summarized as Algorithm 1.

Algorithm 1 Efficient Solver for SP Problem

```
Input: P2, initial grid points $\hat{x}_{j0}$ and $k_j = 1$
1: Solve P2
2: Solve subproblem (18) and obtain $\psi_j(\hat{x}_j)$'s and $\hat{\lambda}_{jv}$'s ($v \in [1, k_j]$)
3: For ($\psi_j(\hat{x}_j) < 0$)
     Add new grid point $\hat{x}_{j(k_j+1)} = \sum_{v=1}^{k_j} \hat{x}_{jv}\hat{\lambda}_{jv}$ with $\hat{\lambda}_{j(k_j+1)} = 0$, and set $k_j = k_j + 1$
     Update P2, solve P2 and obtain $\hat{\lambda}_{jv}, v = 1, \cdots, k_j, j = 1, \cdots, n$
     Solve subproblem (18) and update $\psi_j(\hat{x}_j)$ for all $j \in [1, n]$
   end
Output: $\hat{x}_j = \sum_{v=1}^{k_j} \hat{x}_{jv}\hat{\lambda}_{jv}$ and $\sum_{j=1}^{n} f_j(\hat{x}_j)$
```

D. A Privacy-preserving Outsourcing Algorithm

In what follows, we develop a privacy-preserving outsourcing algorithm to solve the large-scale CSP problem with the help of the cloud.

Particularly, as shown in Section IV-B, the client linearize the original problem **P1** into **P2**. Considering that P2 is a large-scale problem and computationally prohibitive for the client to solve by itself, **P2** will be outsourced to the cloud for solutions. To protect client's data privacy, we conduct some transformations based on the proposed scheme in Section III.

Recall that the original problem **P1** is linearized and transformed into **P2** before outsourcing to the cloud. Only $f_j(x_{jv})$'s are sent to the cloud while x_{jv}'s are kept privately by the client. Therefore, the client has protected the privacy of $f_j(x_j)$'s and $g_{ij}(x_j)$'s. However, the problem **P2** still contains sensitive information, i.e., λ_{jv}'s $(j \in [1, n], v \in [1, k_j])$, which can enable the cloud to obtain the final optimal objective function value of **P2** and hence of **P1** as well. To address this issue, we hide the vectors of $\Lambda_{jk_j} = \{\lambda_{jv}, v = 1, \cdots, k_j\}$ for every $j \in [1, \cdots, n]$ by adding a random vector $\mathbf{r_j}$:

$$\bar{\Lambda}_{jk_j} = \Lambda_{jk_j} + \mathbf{r_j}, j = 1, \cdots, n \tag{19}$$

where $\bar{\Lambda}_{jk_j}$ is a vector with elements denoted by $\bar{\lambda}_{jv} = \lambda_{jv} + r_{jv}$ $(v = 1, \cdots, k_j)$ and $\mathbf{r_j}$ is constructed following Section III. In addition, the lower bounds on λ_{jv}'s, will become $\bar{\lambda}_{jv} \geq r_j$, in which r_{jv}'s still contain private information that can not be revealed to the cloud. Consequently, we transform the constraint (17) by:

$$\mathbf{A}\bar{\Lambda}_{jk_j} \geq \mathbf{A}\mathbf{r_j}, j = 1, \cdots, n \tag{20}$$

where $\mathbf{A}$ is a $k_j \times k_j$ positive random matrix with elements being sampled from a uniform distribution $(0, L')$. Note that each element of $\mathbf{Y}_j$ will be masked by k_j random number and hence can be protected.

Now we can transform the problem **P2** into a privacy-preserving problem **P3**:

$$\textbf{P3:} \quad \textbf{Min} \quad \sum_{j=1}^{n}\sum_{v=1}^{k_j} f_j(x_{jv})\bar{\lambda}_{jv},$$

$$\textbf{s.t.} \quad \sum_{j=1}^{n}\sum_{v=1}^{k_j} g_{ij}(x_{jv})\bar{\lambda}_{jv} \leq b_i + \sum_{j=1}^{n}\sum_{v=1}^{k_j} g_{ij}(x_{jv}) r_{jv}, i = 1, \cdots, m \tag{21}$$

$$\sum_{v=1}^{k_j} \bar{\lambda}_{jv} = 1 + \sum_{v=1}^{k_j} r_{jv}, j = 1, \cdots, n \tag{22}$$

$$\mathbf{A}\bar{\Lambda}_{jk_j} \geq \mathbf{A}\mathbf{r_j}, j = 1, \cdots, n \tag{23}$$

in which $\sum_{j=1}^{n}\sum_{v=1}^{k_j} g_{ij}(x_{jv}) r_{jv}$ and $\sum_{v=1}^{k_j} r_{jv}$ are calculated by the client and then sent to the cloud as constraints. Since **P3** is a standard linear programming problem, the cloud can use the existing solvers such as the interior point methods [22] to obtain the result. Similar to that in Algorithm 1, after the cloud sends back the solutions, the client will check locally if adding another grid point can lead to a better solution for original problem. Note that to verify the correctness of the returned result from the cloud, the client only needs to check if the KKT conditions hold at point $\bar{\lambda}_{jv}$'s.

The details of the proposed privacy-preserving outsourcing scheme for large-scale CSP problems are summarized in Algorithm 2.

Algorithm 2 Privacy-preserving Scheme for CSP Problem

```
Input: P3, initial grid point $\hat{x}_{j0}$ and $k_j = 1$
1: Cloud solves P3 and sends the result to the client
2: Client solves $\Lambda_{jk_j} = \bar{\Lambda}_{jk_j} - \mathbf{r_j}, j = 1, \cdots, n$
3: Client solves subproblem (18) and obtain $\psi_j(\hat{x}_j)$'s and $\hat{\lambda}_{jv}$'s ($v \in [1, k_j]$)
4: For ($\psi_j(\hat{x}_j) < 0$)
     Client adds a new grid point $\hat{x}_{j(k_j+1)} = \sum_{v=1}^{k_j} \hat{x}_{jv}\hat{\lambda}_{jv}$ with $\hat{\lambda}_{j(k_j+1)} = 0$, and set $k_j = k_j + 1$
     Client updates P2 with the new grid points
     Client transforms P2 into P3 and send it to the cloud
     Cloud solves P3 and sends the result to the client
     Client solves subproblem (18) and update $\psi_j(\hat{x}_j)$ for all $j \in [1, n]$
   end
Output: $\hat{x}_j = \sum_{v=1}^{k} \hat{x}_{jv}\hat{\lambda}_{jv}$ and $\sum_{j=1}^{n} f_j(\hat{x}_j)$
```

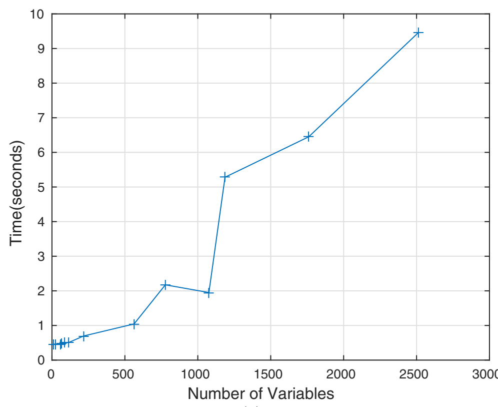
V. Evaluation Results

In this section, we present the performance of the proposed privacy-preserving outsourcing algorithm for CSPs. To evaluate our algorithm in a practical scenario, we implement the client-side computations of the proposed algorithm on a laptop with a dual-core 2.6 GHz CPU, 8 GB RAM, and a 150 GB solid state drive, and the cloud-side computations on an AWS EC2 instance. We implement both the client-side and the cloud-side computations of the proposed algorithm on Matlab 2015a. We evaluate the performance of our algorithm by generating random large-scale CSPs.

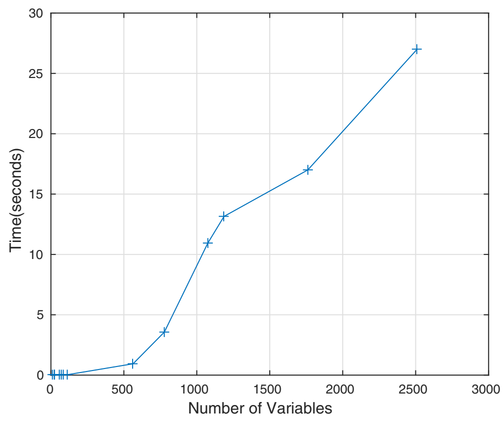
First, we measure the computing time of our proposed algorithm at both the client and at the cloud, and show the results in Fig. 3. In particular, we measure the computation time of the client, that is, the time it takes to find the transformed problems, i.e., **P2** and **P3**, plus the time it takes to find the grid points. Fig. 2(a) shows the client's computing time for CSPs with an increasing number of variables. We observe that even when the CSP problem has a large number of variables, the client can still finish its computations in a very short time. For example, the computing time of the client the CSP with 2.5×10^3 variables is only 9.5s.

In Fig. 2(b), we show the total computing time at the cloud for solving the transformed CSP with different numbers of variables. We observe a low computing time for the cloud even when the number of variables is vary large large separable programs. For instance, the cloud takes 26.5s to solve a SP problem with 2.5×10^3 variables, which is very efficient in real-world scenarios.

Next, we explore the computational savings offered by our proposed algorithm. Specifically, we compare the time it takes for the client to solve the CSPs by itself with that when the client and the cloud collaborate to solve the CSPs using our proposed privacy-preserving outsourcing algorithm. We first show the time that the client takes to solve CSPs with an increasing number of variables by on its own in Fig. 3(a). We can see that it increases very fast. Fig. **??** shows the computing time when the client outsources the problem to the cloud and collaborates with the cloud to find the solution. It is obvious that the proposed algorithm offers significant computing time savings to the client. For example, we observe that CSP with 2.5×10^3 variables can be solved in only 34s by using our algorithm, while it takes the client almost 40s to solve the same problem by itself. Moreover, we can observe from Fig. 3(a) and Fig. **??** that when the CSPs do not have many variables, computing by the client itself is more efficient. This is reasonable since our proposed algorithm includes computations for problem transformation. However, when the scale of the problem increases beyond 2×10^3 variables, the proposed algorithm is much more efficient. As the number of variable increases,

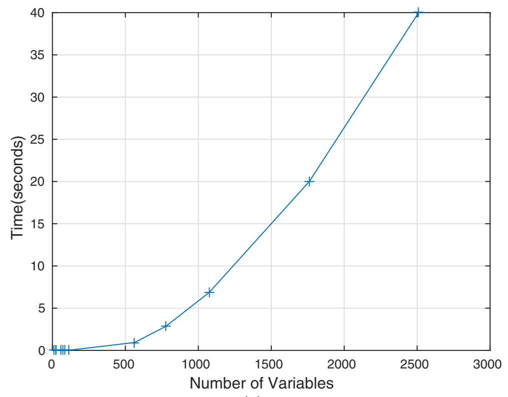


(a)

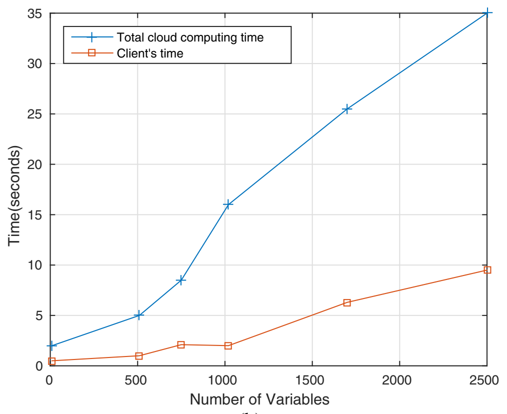


(b)

Figure 2. Computing time of the proposed algorithm at the client and cloud for different size of CSP problems. (a) Computing time at the client. (b) Computing time at the cloud.



(a)



(b)

Figure 3. Comparison for the proposed algorithm between the client and collaboration for different size of CSP problems. (a) Computing time for the client. (b) Computing time for the collaboration.

the proposed scheme will achieve more and more time savings.

VI. Conclusions

In this paper, we have investigated the problem of privacy-preserving outsourcing of large-scale CSPs. To the best of our knowledge, this is the first work to solve CSPs in a privacy-preserving manner in cloud computing. To protect the client's private data, we have developed an efficient vector transformation scheme that is only based on linear algebra and shown that the transformed data is computationally indistinguishable from a random vector. The proposed private linear approximation algorithm can enable the cloud server to efficiently find the solution while protecting the client's privacy. In addition, the correctness of the returned results from the cloud can be efficiently verified by the client to prevent any malicious behavior of the cloud. Experimental results on Amazon Elastic Compute Cloud (EC2) have shown that the proposed algorithm can achieve noticeable time savings.

Acknowledgment

This work was partially supported by the U.S. National Science Foundation under grants CNS-1602172 and CNS-1566479.

References

[1] R. G. Hollands, "Will the real smart city please stand up? intelligent, progressive or entrepreneurial?" *City*, vol. 12, no. 3, pp. 303–320, 2008.

[2] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 1, pp. 131–143, 2013.

[3] V. Chang, D. Bacigalupo, G. Wills, and D. De Roure, "A categorisation of cloud computing business models," in *Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing*, 2010, pp. 509–512.

[4] P. Kim, C. K. Ng, and G. Lim, "When cloud computing meets with semantic web: A new design for e-portfolio systems in the social media era," *British Journal of Educational Technology*, vol. 41, no. 6, pp. 1018–1028, 2010.

[5] S. Salinas, C. Luo, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale quadratic programs," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 281–292.

[6] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of Şbig dataŤ on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[7] W. Venters and E. A. Whitley, "A critical review of cloud computing: researching desires and realities," *Journal of Information Technology*, vol. 27, no. 3, pp. 179–197, 2012.

[8] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing– the business perspective," *Decision support systems*, vol. 51, no. 1, pp. 176–189, 2011.

[9] C. Wang, K. Ren, J. Wang, and K. M. R. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," in *31st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2011, pp. 549–558.

[10] S. Salinas, C. Luo, X. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1035–1043.

[11] S. Salinas, X. Chen, J. Ji, and P. Li, "A tutorial on secure outsourcing of large-scale computations for big data," *IEEE Access*, vol. 4, pp. 1406–1416, 2016.

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

[13] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2386–2396, 2014.

[14] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 497–506.

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

[16] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Transactions on cloud computing*, vol. 1, no. 1, pp. 1–1, 2013.

[17] X. Lei, X. Liao, T. Huang, and H. Li, "Cloud computing service: The caseof large matrix determinant computation," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 688–700, 2015.

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

[19] S. M. Stefanov, *Separable programming: theory and methods*. Springer Science & Business Media, 2013, vol. 53.

[20] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[21] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[22] M. Kojima, S. Mizuno, and A. Yoshise, "A primal-dual interior point algorithm for linear programming," in *Progress in mathematical programming*. Springer, 1989, pp. 29–47.