Highly Efficient Linear Regression Outsourcing to a Cloud

Fei Chen, Tao Xiang, Xinyu Lei, and Jianyong Chen

Abstract—With cloud computing and mobile computing becoming more and more popular, there are a lot potential applications for computation outsourcing to the cloud. This paper investigates the linear regression outsourcing problem, which is a quite common engineering task and employed in various applications, as a case study to find out the possible problems that need to be solved. We propose two protocols which can enable secure and efficient outsourcing of linear regression problems to the cloud. The protocols can protect the client's data privacy well and at the same time have good efficiency. We show all subtleties and the techniques in designing such protocols. The main idea to protect the privacy is employing some transformations to the original linear regression problem to get a new problem which is sent to the cloud; and then transforming the answer returned back from the cloud to get the true solution to the original problem. Experimental results validate the practical usability of our protocols.

Index Terms—Linear regression, computation outsourcing, cloud computing, data transformation

1 Introduction

With the advent of the age of cloud computing and mobile computing, computation outsourcing is becoming a popular computing paradigm. Clients with low computation power or resources can off-load their computation tasks to clouds, which are equipped with more computing resources. The clients could be mobile phones, companies, and senor nodes in a wireless sensor network, etc. Since there are a lot of such applications, this paper investigates the computation outsourcing problem with a case study of the linear regression (LR) task.

Computation outsourcing. As a general computing paradigm, computation outsourcing is widely employed in various practical applications and in distributed computing applications. In the financial industry [1], companies can outsource their computations on the anticipation of the economic developments, such as tax rate, inflation rate, stock price, etc. In the energy industry [1], companies can also outsource their huge computation to a third party. The computation may try to find where there is some oil or some new energy material in a certain area. The data is so huge that the computation is quite time-consuming and requires special software. In distributed computing projects such as SETI@HOME [2], FOLD@HOME [3], the large-scale computation is also divided into a lot of parts and outsourced to some distributed parties. In sensor networks, a sensor node may also outsource its computation to a more powerful node, such as the sink [4].

The reasons why computation outsourcing is needed include, from the client's perspective, low computation power, lacking special software, and high cost to carry out the computation locally. Suppose a financial company running a business has a computation intensive task, e.g. computation on some economic models. If the company chooses to carry out the computation on its own, it may take a lot of time due to limited economical computing resources, such as CPU, memory, network, etc. This will incur slow response to the market, which means the company possibly cannot make money and survive. Moreover, it requires a lot of human resources and computing resources to maintain its own data center and computing facilities, which implies a high maintenance cost. Therefore, outsourcing the computation to a third party, e.g. a cloud, is a better choice. These reasons also serve as the motivation for cloud computing.

However, there are a lot of challenges in this computing paradigm. The most important one is the privacy issue, including the client's input/output data privacy. The data of the client is often so valuable that the client doesn't want a third party to get a meaningful sense of the data. This is because the data may be the proprietary asset of the company, and leaking out the data may decrease the advantage of the company in the competitive market. Thus, a protocol that can fulfill computation outsourcing and protect the privacy of the data simultaneously is highly expected. The second challenge is the verification of the result returned by the third party. Sometimes, a third party may be lazy in the computation and just return a random false answer to the computation outsourced. This may be due to accidental and intentional reasons. In some cases, the third party may have some hardware and software faults when carrying out the computation. These failures will result in a wrong computation. In other cases, the third party may intentionally returns

- *F. Chen is with the Department of Computer Science and Engineering, Shenzhen University, Shenzhen 518060, China and the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China. E-mail: feichenn@gmail.com.*
- *T. Xiang and X. Lei are with the College of Computer Science, Chongqing University, Chongqing 400044, China. E-mail: txiang@cqu.edu.cn, xy-lei@qq.com.*
- *J. Chen is with the Department of Computer Science and Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: jychen@szu.edu.cn.*

Manuscript received 11 Jan. 2014; revised 26 Oct. 2014; accepted 27 Nov. 2014. Date of publication 7 Dec. 2014; date of current version 30 Jan. 2015. Recommended for acceptance by D.S.L. Wei, S. Pearson, K. Matsuura, P.P.C. Lee, and K. Naik.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2014.2378757

2168-7161 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

back a false answer. Although the third party has more computation resources than the client, it also has limited resources and has the financial incentive to focus all its resources to those clients who pay more. It may also want to get some useful knowledge of the client's data by returning an incorrect answer. This implies that the protocol handling the computation outsourcing needs to have a way to detect whether the returned computation result is correct or not. Another challenge is efficiency. In order to protect the privacy, the original computation problem needs to be transformed in some way and this requires time. The time is expected to be smaller than the time needed if the client chooses to carry out the computation locally. Therefore, a computation outsourcing protocol should satisfy four conditions: it is correct, secure, verifiable and efficient.

Linear regression. To study the problem of secure and efficient computation outsourcing, we consider the linear regression task outsourcing to a cloud as an example. Linear regression is a common engineering computation used everywhere in practice. In machine learning [5], linear regression is used as a model to do training and prediction on a data set. For example, it can be used to predict the sales of a new book according to past experiences, e.g. the author's publication record, the reader's reading habit, the publication press, etc. In sensor networks, this model can also help a sensor node to locate itself in a two-dimensional or three-dimensional space according to the information it collects from its neighbors. In financial applications, the model can help predict the price of a stock according to the past and recent trends of similar stocks.

Linear regression is also a computation-intensive task if the data set is large. It involves a lot of matrix operations such as inversion, multiplication, etc. When the matrix is large, memory is often a bottleneck for fast computation. Thus, a client can outsource such computation to a cloud. A secure and efficient protocol that enables linear regression outsourcing is helpful in practice, especially in the age of cloud computing.

Our work and contribution. We motivate and formulate the problem of secure and efficient linear regression outsourcing to the cloud. Based on a model of linear regression, we design two protocols that can outsource this kind of computation to the cloud in a correct, secure, verifiable and efficient way. The key idea of the protocols is to employ some efficient linear transformations on the input data. The transformations not only protect the privacy of the input data, but also help mask the output data. The transformations serve as the secret key of the client. To recover the computation result, the client performs some other transformations on the returned answer using the secret key and checks whether the cloud cheats. Theoretical analysis and experimental results show that the protocols can be used immediately in practice. To summarize, our contributions include the followings:

- We identify a common engineering task, i.e. linear regression, motivate and formulate the problem of linear regression outsourcing for the first time.
- We design two protocols that can enable outsourcing linear regression computation to a cloud. The two protocols are designed for different applications, which feature different security and efficiency requirements. From technical perspective, we discuss several potential design approaches and finally target one approach which we believe the best. These potential techniques can serve as the foundation for other secure protocol designs.
- We analyze our protocols both theoretically and experimentally, which validates the practicibility of the protocols. All the experimental results can be reproduced.

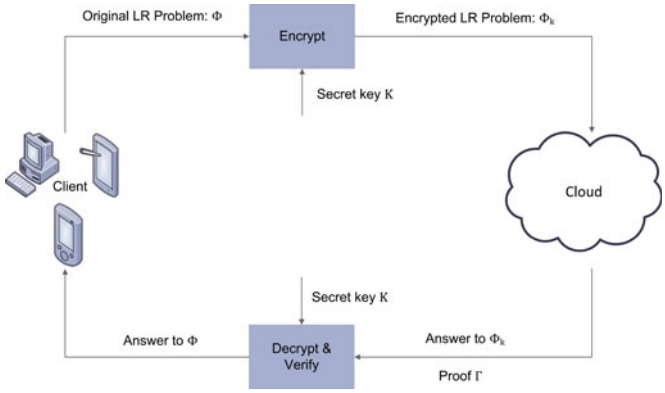


Fig. 1. LR outsouricng system model.

The following of the paper proceeds as follows. Section 2 formulates the linear regression outsourcing problem and presents a framework to solve the problem. A formal model of linear regression is also presented. Section 3 designs two protocols for secure and efficient linear regression computation outsourcing according to the framework in Section 2. A lot of design details and subtleties are also presented. Security analysis of our protocols follows immediately and is shown in Section 4, together with some further discussions on our protocols. Section 5 shows experimental results of the most efficient protocol. Section 6 discusses related work on computation outsourcing. Finally, the paper is concluded in Section 7.

2 Problem Formulation

In this section, we define the linear regression outsourcing problem that we are going to investigate. In order to solve this problem, a framework is employed, which divides the problem into several parts. Each part can be individually solved. The framework is inherited from [6], [7], [8]. To handle the linear regression outsourcing problem, we need to focus on a detailed linear regression model and such a model is also presented. The model helps in our protocol design.

2.1 System Model, Design Goals, and Framework

Setup. We consider the cloud computing application scenario and the system model is shown as in Fig. 1. A client with low computation power has an engineering problem, where a linear regression model is employed. So the client needs to solve the LR problem. Since the limited computation resource the client possesses, the client wants to outsource the computation to a cloud service provider, who has a more powerful computation power and special

software. However, at the same time, the client doesn't want the cloud to get meaningful knowledge about the data. Thus, the client chooses to encrypt the original LR problem Φ using a secret key K to get an encrypted problem Φ_K. Later, the encrypted problem is given to the cloud for a solution. Once the cloud receives the problem, the computation is carried out with some software; then the cloud sends back the answer of Φ_K to the client. The cloud also sends back a proof Γ that tries to show the returned answer is indeed a correct answer and the cloud doesn't cheat. On receiving the returned answer, the client decrypts the answer using the secret key K to get an answer to the original problem Φ. The client checks whether this answer is correct; if yes, accept the answer; if no, just reject it.

Assumptions. We model the cloud as malicious. On one hand, the cloud may try to learn some useful information about the client's data. This is because the client's data may contain valuable commercial information. On the other hand, the cloud may also cheat in the computation. The cloud may just return a random answer to the client, instead of carrying out the computation to get a true answer. This saves the cloud's computing resources. The reason is that the cloud may expect the client not to check the correctness of the returned answer. Even the cloud is not malicious, the returned answer could also be false because of the software and hardware failures in the cloud. Thus, result verification is a must. We also assume the cloud is computationally bounded, i.e. it can only work on polynomial time computations. For those **NP** problems, the cloud cannot get a solution in a reasonable time.

Design goals. Once the problem is clear, we need to design a protocol to solve the problem. An efficient and secure linear regression outsourcing protocol is expected to have the following properties:

- *Correct*. The protocol can fulfill the task of LR computation outsourcing. In other words, if the client and the cloud both follow the protocol honestly, the LR problem can indeed be solved by the cloud and the client gets a correct answer to the LR problem.
- *Secure*. The protocol can protect the privacy of the client's data. On one hand, the cloud cannot get meaningful knowledge of the client's input data, which is called input privacy. On the other hand, the correct answer of the original problem is also hidden from the cloud, which is called output privacy.
- *Efficient*. The computation, memory and communication cost of the client should be as small as possible. The sum of the time of transforming the original problem to an encrypted one and the time of verifying whether a returned answer is correct should be strictly smaller than the time that is needed to solve the LR problem on the client's own. Otherwise, there is no motivation for outsourcing the computation problem.
- *Verifiable*. The client should have a way to figure out the correctness of an answer. The cloud cannot cheat.
- *Suitable for immediate practical uses*. The protocol should be able to be deployed immediately in practice.

Framework. Syntactically, a secure and efficient linear regression outsourcing protocol should contain four algorithms (KeyGen, ProbTransform, ProbSolve, ResultVerify) as follows:

- KeyGen(1^λ). On input a security parameter λ, the client uses this algorithm to generate an encryption key K. This key will be used to transform the original problem and to decrypt the returned answer.
- ProbTransform$(\Phi; K)$. On input the original problem Φ and the secret key K, the client uses this algorithm to transform the original problem to an encrypted problem Φ_K to protect the client's data.
- ProbSolve(Φ_K). On input the encrypted problem Φ_K, the cloud uses this algorithm to solve Φ_K. Then, the cloud returns the answer β back, together with a proof Γ showing that the returned answer is correct.
- ResultVerify$(\beta, \Gamma; K)$. On input the returned answer β, Γ and the secret key K, the client decrypts to get an answer for the original LR problem. The client also checks whether the answer is correct. If the answer is true, then accept the answer; otherwise, reject the answer.

Everytime the client has an LR problem, the client will run the above protocol. The key point is that the KeyGen will be run every time when there is an LR problem to be outsourced, which means the protocol is similar to the one-time pad encryption scheme. Indeed, a one-time encryption scheme is the most secure one. Once we have the framework, we just need to work out the details of these four algorithms, which will be shown in Section 3.

2.2 Linear Regression

Model. Linear regression model is employed a lot in various applications [9], e.g. machine learning, finance, and statistics. A typical linear regression model is as follows:

$$\mathbf{y} = \mathbf{X}\beta, \tag{1}$$

where $\mathbf{y}$ is an $m \times 1$ vector in $\mathbb{R}^m$, $\mathbf{X}$ is an $m \times n$ matrix in $\mathbb{R}^{m \times n}$, β is an $n \times 1$ vector in $\mathbb{R}^n$ and most importantly $m > n$. In practical applications, $\mathbf{y}$ is called the output variable, which is a linear function of the input variable $\mathbf{X}$. The coefficient of the linear function is β. Normally, we can observe a lot of input-output pairs in practice; however, we don't know the coefficient β. A natural choice to find β is to employ a least squared error method to find a good approximation of β according to the many samples in practice. This method yields an solution for β as follows [9]:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2}$$

In practice, we just need to solve this equation to get a good value for β. Now the task of LR computation outsourcing is to find this β without letting the cloud know the input data $\mathbf{X}, \mathbf{y}$ and the true value of β. The basic idea is to have some encryption on the client's data. Details will be shown in the next section.

Relation with linear equation solving. The linear regression model is quite similar to a system of linear equations solving problem. However, we want to point out some significant differences. The main difference lies in the input

data $\mathbf{X}$ which is not a square matrix. There is a lot of redundancy in the number of linear equations. A correct solution minimizes the squared error of the whole system. However, for a system of linear equations with invertible $\mathbf{X}$, a solution is much easier to find. Another difference lies in the algorithm to solve the underlying problem. To find a solution for linear regression requires more effort than that is needed to find a solution for a system of linear equations. More importantly, researchers have proposed protocols for linear equation outsourcing [10], [11] while linear regression outsourcing remains open, to the best knowledge of the authors'.

3 Protocols for Secure Linear Regression Outsourcing

In this section, we detail the design of our protocols for secure and efficient linear regression outsourcing. First, we show our basic idea on handling this problem, i.e. transforming the original LR problem linearly to get a new LR problem. Then, we design special techniques for the problem transformation, which helps protect the privacy of the client's data. Several possible techniques are discussed and then the best is chosen. Following this, the verification algorithm and key generation algorithm are presented. Normally, more security can be obtained if we add more complexity in the protocol design. However, we aim at designing highly efficient protocols with sufficient security. Once all the basic ingredients are well prepared, we finally present our protocols.

3.1 Basic Idea

Our protocols aim at immediate practical uses for the linear regression outsourcing problem. Although there may be a few possible choices for secure outsourcing, we adopt the linear transformation approach because of the efficiency issue. For other approaches, please refer to Section 6. The basic idea is as follows: Given a linear regression problem as in Eq. (1), denote this problem as $\Phi = (\mathbf{X}, \mathbf{y})$ and its solution as β. We carefully design some linear transformations on the client's data to get a new LR problem using a secret key K, denoted as $\Phi_K = (\mathbf{X}', \mathbf{y}')$. The new LR problem is in the exact same form as in Eq. (1). Thus, from the viewpoint of the cloud, the original problem and the new problem are the same. In this way, the privacy of the input data is protected from the cloud. However, we need to solve the original LR problem, which requires that the solution of the new LR problem has some relation with the original solution. Using the secrete key, the client can transform the answer of the new LR problem β' returned from the cloud to get an answer β to the original LR problem. In this way, the privacy of the output data is also protected from the cloud since the cloud only has an answer to the encrypted LR problem. For verification, it seems easy because of the special structure of the LR problem. We just need to check whether the residual error $\|\mathbf{X}\beta - \mathbf{y}\|$ is small.

Note that we need to implement these ideas in detailed protocols. A protocol should contain four algorithms (KeyGen, ProbTransform, ProbSolve, ResultVerify). When the above ideas are applied, the secret key of the client is generated in the KeyGenalgorithm. The client transforms the original LR problem $\Phi = (\mathbf{X}, \mathbf{y})$ to get a new LR problem $\Phi_K = (\mathbf{X}', \mathbf{y}')$ using the secret key in the ProbTransform algorithm; then the client sends Φ_K to the cloud. The cloud solves Φ_K to get an answer β' and sends it back to the client in the ProbSolve algorithm. The client recovers the original answer β and checks the correctness in the ResultVerify algorithm.

3.2 Protecting Input Privacy and Output Privacy

Here we show how to design these linear transformations to protect the privacy of the client's data. For an LR problem $\Phi = (\mathbf{X}, \mathbf{y})$, its solution is $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. The privacy of $(\mathbf{X}, \mathbf{y})$ is called input privacy; the privacy of β is called output privacy. To hide the original problem Φ, a natural choice is to do some efficient and secure linear transformations on the data $(\mathbf{X}, \mathbf{y})$ to get a new LR problem $\Phi_K = (\mathbf{X}', \mathbf{y}')$.

Attempt 1. Notice that $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. This reminds us to transform $\mathbf{X}$ into a matrix $\mathbf{X}' = \mathbf{AX}$. Now we have $\beta' = (\mathbf{X}'^T\mathbf{X}')^{-1}\mathbf{X}'^T\mathbf{y} = (\mathbf{X}^T\mathbf{A}^T\mathbf{AX})^{-1}\mathbf{X}^T\mathbf{A}^T\mathbf{y}$. However, the original solution is $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. In a try to relate β' and β, this gives a hit that $\mathbf{A}$ should be an orthogonal matrix, i.e. $\mathbf{A}^T\mathbf{A} = \mathbf{I}$. In this case, we have $\beta' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^T\mathbf{y}$, which is much similar to β. Now it seems still hard to relate the two values β' and β. Besides, we also need to protect the privacy of $\mathbf{y}$. This suggests to transform $\mathbf{y}$ in some way. There are several choices:

1) A first think indicates we should transform $\mathbf{y}$ into $\mathbf{y}' = \mathbf{Dy}$ using a matrix $\mathbf{D}$. This yields $\beta' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^T\mathbf{Dy}$. The annoying term $\mathbf{A}^T\mathbf{D}$ prevents us from linking β' and β.
2) To deal with the annoying term, we may try $\mathbf{y}' = \mathbf{ADy}$. Now we have $\beta' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^T\mathbf{ADy} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Dy}$. This form is much closer to $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Although having tried for a while, we cannot find a way to establish a useful relation between β' and β.
3) To eliminate the matrix $\mathbf{D}$, we can just transform $\mathbf{y}$ into $\mathbf{y}' = \mathbf{Ay}$. This results in $\beta' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^T\mathbf{Ay} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, which means $\beta' = \beta$. We have successfully linked the two solutions of the original LR problem and the transformed new LR problem. However, a new trouble emerges since $\beta' = \beta$ leaks the output to the cloud! We may need to turn to some other directions.

Attempt 2. Although we got stuck, attempt 1 indeed gives some inspiration on how to handle the privacy problem. A second thought shows we may split $\mathbf{y}$ into a series of $\mathbf{y}_i$'s such that $\mathbf{y} = \sum \mathbf{y}_i$. In this case, we can send several transformed LR problems to the cloud $\Phi_{K,i} = (\mathbf{AX}, \mathbf{Ay}_i)$. Suppose the answer to $\Phi_{K,i}$ is β_i'. Then, we have $\beta = \sum \beta_i'$. This seems a possible solution if the detailed protocol is hidden from the cloud, i.e. the cloud does not know the original problem is separated into a series of sub-problems. But this is not a plausible assumption. Besides, this solution incurs much more computation and communication cost for the client and the cloud. We need to find other solutions.

ATTEMPT 3. After thinking about attempts 1 and 2 for a while, we find that we could transform the original problem in this way: $\mathbf{X}' = \mathbf{AXD}$ and $\mathbf{y}' = \mathbf{Ay}$. This yields that

$$\begin{aligned}\beta' &= (\mathbf{X}'^T\mathbf{X}')^{-1}\mathbf{X}'^T\mathbf{y}' = (\mathbf{D}^T\mathbf{X}^T\mathbf{A}^T\mathbf{AXD})^{-1}\mathbf{D}^T\mathbf{X}^T\mathbf{A}^T\mathbf{Ay} \\ &= \mathbf{D}^{-1}(\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{D}^T)^{-1}\mathbf{D}^T\mathbf{X}^T\mathbf{y} = \mathbf{D}^{-1}\beta,\end{aligned} \tag{3}$$

where $\mathbf{A}$ is an orthogonal matrix in $\mathbb{R}^{m\times m}$ and $\mathbf{D}$ is an invertible matrix in $\mathbb{R}^{n\times n}$. There is also a relation between β' and β. Then, the matrix $\mathbf{A}$ and $\mathbf{D}$ will serve as the secret key of the client.

ATTEMPT 4. A natural question is whether we can do better. Notice that if we restrict $\mathbf{A}$ and $\mathbf{D}$ to some special forms, e.g. diagonal, we can have some computational gains. Thus, for the matrix $\mathbf{D}$, we can require it be diagonal. For the orthogonal matrix $\mathbf{A}$, we can restrict it be diagonal. But this will imply all the entries of $\mathbf{A}$ be 1 or -1. Thus, we relax a little bit on the orthogonality of $\mathbf{A}$. Let k be a real number and the diagonal matrix $\mathbf{A}$ be $\mathbf{A}^T\mathbf{A} = k^2\mathbf{I}$. In other words, all entries of $\mathbf{A}$ are k or $-k$. Now the transformed problem is $\Phi_K = (\mathbf{X}', \mathbf{y}') = (\mathbf{AXD}, \mathbf{Ay})$ and the relation of β' and β is

$$\begin{aligned}\beta' &= (\mathbf{X}'^T\mathbf{X}')^{-1}\mathbf{X}'^T\mathbf{y}' = (\mathbf{D}^T\mathbf{X}^T\mathbf{A}^T\mathbf{AXD})^{-1}\mathbf{D}^T\mathbf{X}^T\mathbf{A}^T\mathbf{Ay} \\ &= \mathbf{D}^{-1}(k^2\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T k^2\mathbf{y} = \mathbf{D}^{-1}\beta,\end{aligned} \tag{4}$$

i.e. remaining unchanged. The special $\mathbf{A}$ and $\mathbf{D}$ improve the efficiency; however, there is a trade-off between the security and the efficiency.

Remark 1. We note that the random matrix $\mathbf{A}$ as in [12], [13] cannot work well for the linear regression outsourcing problem here since we need $\mathbf{A}^T\mathbf{A} = \mathbf{I}$. For the diagonal matrix $\mathbf{D}$, the monomial matrix can also be applied here. For streaming data, a diagonal matrix requires less computation than a monomial matrix; however, a monomial matrix provides more security. Thus for applications demanding more strict security, a monomial matrix is a good choice. We also note that the inference attacks in [13] for linear programming do not apply for linear regression because of the redundant equality constraints. Further, the input $\mathbf{X}$ over real numbers cannot be recovered from $\mathbf{X}'$ easily using greatest common divisor attacks, which can work over integers. It is then recommended that the input $\mathbf{X}$ should be disturbed with noises before outsourcing the computation.

We will next construct two protocols based on attempts 3 and 4. One is more secure; the other is more efficient. We will detail the security issue in Section 4.

3.3 Key Generation and Correctness Verification

Note that for the original LR problem $\Phi = (\mathbf{X}, \mathbf{y})$, the transformation is $\mathbf{X}' = \mathbf{AXD}$ and $\mathbf{y}' = \mathbf{Ay}$. The matrices $\mathbf{A}$ and $\mathbf{D}$ are the client's secret key. In order to generate such matrices, we need a truly random number string, and then pack the truly random numbers to get $\mathbf{A}$ and $\mathbf{D}$. However, it is normally quite hard to generate such truly random strings in practice [14]. Instead, we will use cryptographic pseudo-random number generators [14], [15], [16] to generate such a string of numbers, which looks like the same as a truly random one. The advantage is that the key will be much smaller, normally 128 or 256 bits; while the security remains almost unchanged.

Generally, verification is not an easy problem. However, we can employ the special structure of the linear regression problem to verify whether an answer is correct. Suppose $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is a solution to an LR problem $\Phi = (\mathbf{X}, \mathbf{y})$. For the ith element in the vector $\mathbf{y} - \mathbf{X}\beta$, define its absolute value as its residual error, and define all possible residual errors as its error range. Then, the residual error of each element in $\mathbf{y} - \mathbf{X}\beta$ should be small. A client is possible to have a prior experience on the error range. After the client receives the returned answer from the cloud, the client can decrypt the answer using the secret key; then the client can compute the residual error for each element of the vector $\mathbf{y} - \mathbf{X}\beta$. If the error is in the reasonable range, accept the answer; if it is out of the range, reject the answer. This approach is effective for many practical problems. On one hand, the linear regression model assumes theoretically that the error in each constraint of the linear regression problem satisfies some normal distribution. Thus, the length of the error vector is small and can be obtained using background information and previous experimental data. On the other hand, many practical applications also have a small error vector. Currently, we only figure out this approach for result verification. There could also be some other better verification methods; we leave it for future research.

3.4 Detailed Protocols

Now we present two protocols that can securely and efficiently outsource linear regression computation to the cloud. Protocol 1 is more secure while protocol 2 is more efficient. They can be chosen according to the security and efficiency requirements of different applications. The only difference of protocols 1 and 2 lies in the key generation algorithm.

PROTOCOL 1. Our protocol for secure LR outsourcing contains four algorithms (KeyGen, ProbTransform, ProbSolve, ResultVerify) as follows:

- KeyGen(1^λ). On input a security parameter λ, the client generates an orthogonal matrix $\mathbf{A}$ and a diagonal matrix $\mathbf{D}$. The matrices are packed from a cryptographic pseudo-random number generator. Denote $K = (\mathbf{A}, \mathbf{D})$.
- ProbTransform$(\Phi; K)$. On input the original LR problem $\Phi = (\mathbf{X}, \mathbf{y})$ and the secret key K, it generates a new LR problem $\Phi_K = (\mathbf{AXD}, \mathbf{Ay})$, and sends Φ_K to the cloud.
- ProbSolve(Φ_K). On input the encrypted problem Φ_K, the cloud solves Φ_K using an LR solver to get an answer β'. Then, the cloud returns β' and an empty proof Γ to the client.
- ResultVerify$(\beta', \Gamma; K)$. On input the returned answer β', Γ and the secret key K, the client decrypts β' to get $\beta = \mathbf{D}\beta'$ for the original LR problem. The client also checks whether the residual error of each element in $\mathbf{y} - \mathbf{X}\beta$ is small. If yes, then accept the answer; otherwise, reject the answer.

TABLE 1
Theoretical Performance of Our Protocols for Secure Linear Regression Outsoucing

Client				Cloud
KeyGen	ProbTransform	ResultVerify	Communication	ProbSolve
$\mathbf{A}, \mathbf{D}$	$m^2n + mn + m$ or $2mn + m$	$mn + m$	$(\mathbf{X}', \mathbf{y}')$	$n^2m + n^3 + mn + n^2$

Protocol 2. This protocol is almost the same as protocol 1 except that $\mathbf{A}$ is not orthogonal. Now $\mathbf{A}$ is a diagonal matrix and $\mathbf{A}^T\mathbf{A} = k^2\mathbf{I}$. The entries of $\mathbf{A}$ is randomly set to be k or $-k$ according to the cryptographic pseudo-random number string. As normal, $\mathbf{A}$ and $\mathbf{D}$ are also packed from the pseudo-random number string. This protocol is more efficient than protocol 1 since the problem transformation is much simpler. In protocol 1, the problem transformation involves a matrix multiplication; while protocol 2 simplifies this computation because of the special form of $\mathbf{A}$. We show the analysis of the two protocols in the next section.

4 Analysis and Further Discussion

In this section, we analyze the security properties and performance of the proposed protocols. Furthermore, we also want to ask and partially solve the question: Can we do better?

4.1 Analysis of the Proposed Protocols

Correctness. If the client and the cloud both follow the protocol honestly, the task of linear regression outsourcing can indeed be done correctly. Eqs. (3) and (4) show indeed the client can recover the correct answer by $\beta = \mathbf{D}\beta'$, where β' is the retuned answer from the cloud and β is the true answer of the original LR problem.

Privacy. We follow previous work, e.g. [7], [8] , [10] etc., in this area for the security analysis. First, we want to remind that our protocols are similar to the one-time pad encryption scheme. A new secret key is used every time the protocol is run. Thus, there is no know-plaintext attack or chosen-plaintext attack. Now we discuss the privacy issue of the two protocols one by one. Suppose the original LR problem is $\Phi = (\mathbf{X}, \mathbf{y})$ and the encrypted problem is $\Phi_K = (\mathbf{X}', \mathbf{y}')$ where $\mathbf{X}' = \mathbf{AXD}$ and $\mathbf{y}' = \mathbf{Ay}$. Let β and β' be solutions to Φ and Φ_K, respectively.

For protocol 1, $\mathbf{A}$ is an orthogonal matrix and $\mathbf{D}$ is a diagonal matrix. The goal of input privacy is to protect $(\mathbf{X}, \mathbf{y})$ from the cloud. In the viewpoint of the cloud, the only information is $(\mathbf{X}' = \mathbf{AXD}, \mathbf{y}' = \mathbf{Ay})$. Given $\mathbf{X}'$, for any $\mathbf{X}$, there are some matrices $\mathbf{A}$ and $\mathbf{D}$ such that $\mathbf{X}' = \mathbf{AXD}$. This implies that $\mathbf{X}'$ gives no information about the input $\mathbf{X}$ since $\mathbf{A}$ and $\mathbf{D}$ is the secret key of the client and the cloud has no idea of what they are. Given $\mathbf{y}'$, for any $\mathbf{y}$, there is also some matrix $\mathbf{A}$ such that $\mathbf{y}' = \mathbf{Ay}$. Thus, $\mathbf{y}'$ gives no information about the input $\mathbf{y}$. However, the pair $(\mathbf{X}' = \mathbf{AXD}, \mathbf{y}' = \mathbf{Ay})$ together does leak some information about $(\mathbf{X}, \mathbf{y})$. But we argue that the cloud still has a great uncertainty about the client's data as follows.

Given the pair $(\mathbf{X}', \mathbf{y}')$, there are a lot of choices for the secret key $\mathbf{A} \in \mathbb{R}^{m\times m}$ and $\mathbf{D} \in \mathbb{R}^{n\times n}$. Suppose each entry of the matrix is an l-bit number. Since $\mathbf{A}$ is an orthogonal matrix, there are roughly $\frac{1}{2}m^2l$ bit information for $\mathbf{A}$. In another way, there are $2^{\frac{1}{2}m^2l}$ possibilities for the choice of $\mathbf{A}$. Similarly, there are 2^{nl} possible choices for $\mathbf{D}$. Thus, given $(\mathbf{X}', \mathbf{y}')$, there are $2^{\frac{1}{2}m^2l+nl}$ possibilities for the client's data. But notice that the protocol generates a new secret key whenever the protocol is run. Therefore, the cloud cannot find meaningful knowledge about the client's data even for a computationally unbounded cloud.

For output privacy, $\beta = \mathbf{D}\beta'$. Given β', for any β, there is a diagonal matrix $\mathbf{D}$ such that β can be obtained by $\mathbf{D}\beta'$. In other words, any β is possible. Thus, the output of the client's data is also well protected.

For protocol 2, similar analysis also applies. The difference now is about the secret matrix $\mathbf{A}$. Note that now $\mathbf{A}$ is diagonal and the entries of $\mathbf{A}$ is randomly set to be k or $-k$, where k is a random l-bit number. There are 2^{m+l} possible choices for $\mathbf{A}$. In this case, the cloud knows that the client's data is among the 2^{m+l+nl} possibilities, which is an exponential number on m and n. Therefore, the cloud still cannot gain significant knowledge about the client's data.

We remark that both our security analysis here and analysis in previous work are heuristic ones. The spirit is similar to the symmetric encryption algorithm designs, e.g. DES, AES, etc. The algorithm is very efficient, but a formal provable security argument still remains open although researchers have found methods to protect the algorithms from differential and linear cryptoanalysis. Note that there are provable security encryption algorithms based on various hard number-theoretic problem, e.g. RSA-OAEP [17]. However, symmetric encryption algorithms are employed in practice to encrypt large volume data to achieve high efficiency. We also believe that it is an interesting work to investigate whether the protocols in this paper and previous work can be proven secure based on some hard problems. We leave it as a future work.

Theoretical performance analysis. The theoretical analysis is shown in Table 1. If we don't outsource the LR computation, we need to calculate $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. To compute $\mathbf{X}^T\mathbf{X}$ takes time n^2m; then inversion takes time n^3; later $\mathbf{X}^Ty$ takes time nm; finally multiplying $(\mathbf{X}^T\mathbf{X})^{-1}$ and $\mathbf{X}^Ty$ takes time n^2. The total time is $n^2m + n^3 + mn + n^2$, which is also the same for the cloud. For the client who chooses to outsource the LR problem, the cost can be divided into computation cost and communication cost. In the key generation process, protocol 2 requires little time since $\mathbf{A}$ and $\mathbf{D}$ are both diagonal. However, a naive implementation of protocol 1 takes much more time since we need to generate an orthogonal matrix using a Gram-Schmit algorithm. Note the Gram-Schmit algorithm takes a substantial time. To solve this, we can pre-compute a group of orthogonal vectors and pile them up in a pool. Whenever a new orthogonal matrix

TABLE 2
Performance of Secure Linear Regression Outsoucing

#	Benchmark Dimension	Original Problem	Transformed Problem		Client Speedup	Cloud Efficiency
		t_{original}	t_{cloud}	t_{customer}	t_{original} / t_{customer}	t_{original} / t_{cloud}
1	50×30	0.00093801	0.00010415	3.2875×10^{-5}	28.533	9.006
2	100×60	0.0004385	0.00043542	6.6164×10^{-5}	6.6275	1.0071
3	200×120	0.0027623	0.0027577	0.00018262	15.126	1.0017
4	400×240	0.011791	0.011756	0.00088963	13.253	1.003
5	768×10	0.014934	0.009961	0.00055599	26.86	1.4993
6	800×480	0.076318	0.075454	0.0090261	8.4553	1.0114
7	$1{,}600 \times 1{,}000$	0.7169	0.71541	0.041492	17.278	1.0021
8	$4{,}000 \times 3{,}000$	12.554	12.528	0.38685	32.451	1.002

is needed, we can pick some orthogonal vectors in the pool and then generate constant (e.g. 2 or 3) new random orthogonal vectors. Thus, key generation takes little time. For the problem transformation algorithm, the client needs to compute $\mathbf{X}' = \mathbf{AXD}$ and $\mathbf{y}' = \mathbf{Ay}$. For protocol 2, this takes time $2mn + m$ since $\mathbf{A}$ and $\mathbf{D}$ are both diagonal. However, for protocol 1, to compute $\mathbf{AX}$ takes m^2n time, which is quite large. For the result verification algorithm, we just need to compute $\|\mathbf{y} - \mathbf{X}\beta\|$, which takes time $mn + m$. The communication cost is to send $\mathbf{X}', \mathbf{y}'$ to the cloud. Compared with the cost of not outsourcing, the total cost for protocol 2 is $3mn + 2m$, which is strictly smaller than $n^2m + n^3 + mn + n^2$. However, the total cost for protocol 1 is $m^2n + 2mn + 2m$, which is only smaller than $n^2m + n^3 + mn + n^2$ if m is not too larger than n, e.g. $n < m \leq 1.5n$. Note that if $m \geq 2n$, the cost of the client is larger and then there is no computational saving. The point is that for one hand, if m is not too large, the computation cost for protocol 1 is indeed smaller. If m is indeed large, the more efficient protocol 2 is more recommended. For the other hand, protocol 1 possesses more security while protocol 2 possesses more efficiency and sufficient security. The applications can choose protocol 1 or protocol 2 according to the specific requirements.

4.2 Further Improvements

LOWER BOUND. For the efficient protocol 2, the computation cost is $3mn + 2m$. A natural question is whether we can find out a better protocol than this one. A second thought shows that it is possible but the improvement is not that much. To protect the client's data, we need to do some kind of transformation on each entry of $(\mathbf{X}, \mathbf{y})$. Since there are $mn + m$ entries in $(\mathbf{X}, \mathbf{y})$, a computation cost $mn + m$ is a lower bound for any protocol that enables secure and efficient outsourcing of linear regression problems. Therefore, the room for improvement is small; and we can only improve the constants before mn and m.

RANDOMIZED VERIFICATION. To further improve the performance, we can improve the result verification algorithm by employing a randomized one. Instead of checking all entries of $\|\mathbf{y} - \mathbf{X}\beta\|$, we can just check c entries randomly, where c is some constant. If β is a false answer, then it is quite possible that each entry of $\|\mathbf{y} - \mathbf{X}\beta\|$ is larger than expected as a normal error. This can improve the cost to be $2mn + m + cn + c$, smaller than the original cost $3mn + 2m$.

5 PERFORMANCE EVALUATION

In this section, we report the experimental performance evaluation of protocol 2, which is more efficient. Theoretical analysis of this protocol has been given in Table 1, which shows that outsourcing indeed benefits the client. We implement it using Matlab 2010 on a PC simulating a client and a cloud. We want to evaluate whether a client can benefit from outsourcing a linear programming problem. For clients with weak computation resources (which is also among the targets of our protocol), the LR computation cannot be conducted due to limited CPU and RAM. Even secondary memory (e.g. hard drive) can be leveraged to enable the computation, moving data into ant out of the memory costs significant time. Thus, to evaluate the performance of our protocol more objectively, we mainly focus on the transformation time on the client side and the computation time on the cloud side. We do not take into account of the communication time cost by sending the data to the cloud for three reasons: 1) for weak clients with low RAM and secondary storage media, outsourcing the data is the only choice; 2) moving data into and out of the memory costs significant time; and 3) communication cannot be avoided when cloud storage is the only choice for weak clients due to limited local storage.

We randomly generate linear regression problem instances; first, we solve it locally; then we transform the problem to get a new problem and ask the cloud to solve the problem. In all processes, we record the computation time. Finally, we compute the performance gain and average it to get a more stable performance indicator. More details can be found in our source code for performance evaluation. The code can be downloaded from https://sites.google.com/site/chenfeiorange/linear-regression. Thus, all the experimental results can be reproduced.

Our goal is to find the performance gain for the client by outsourcing. Thus, the main performance indicator is a ratio of the time that is needed if the computation is done locally over the time that is needed by the client's computation if outsourcing is chosen. We let t_{original} denote the time for the client to compute the LR problem locally; let t_{cloud} denote the time of the cloud to compute the outsourced problem; let t_{customer} denote the time of the client to transform the original LR problem to a new LR problem. Then, the performance gain of the client can be shown by $\frac{t_{\text{original}}}{t_{\text{customer}}}$; we call this client speedup. This value theoretically should be a

considerable positive number that is greater than 1, which means there is considerable performance gain. We also consider another metric, i.e. the cloud's efficiency, using $\frac{t_{\text{original}}}{t_{\text{cloud}}}$. Ideally, the problem transformation should not increase the time to solve the LR problem. That is, the cloud's efficiency should be around 1. The main performance result is shown in Table 2. We randomly generate 20 LR problems and then measure and average their performance. We also employ a real world data set [18], [19] as in benchmark 5 in Table 2 to validate the proposed LR outsourcing protocol. From Table 2, we can find that there are more than 10 times speedup for LR problems with large dimension, which is expected. For the cloud, its efficiency is also around 1, except the dimension 50×30. This only occurs in low dimensional problems. This exception may also be due to some non-stable random LR instances.

We want to remark that the performance really depends on the implementation of matrix multiplications, problem size and the algorithm solving an LR problem. In practice, there may be more performance gains. For example, the cloud has more computational resources, e.g. memory. If the scale of the LR problem becomes large, there will be a lot of input/output operations for matrix operations. For the client with small memory, a lot of additional cost is needed to move data in and out of the memory. However, the cloud can handle this problem more easily with a large memory.

6 Related Work

The problem of computation outsourcing has attracted a lot of interests both in the security community and the cryptography community. The former community often tries to give some solutions that can be used in practice immediately. They often employ one-time pad style protocols and focus on specific problems. The latter community often makes use of some basic cryptographic tools to form a complex protocol. The secret key can be re-used for different problem instances. The solution is normally quite elegant and suitable for any computational problems, with a condition that these problems can be transformed into a Boolean circuit over $\{0, 1\}$. Indeed, any feasible computation on a Turing machine can be transformed into such a circuit. However, the solution is often far from practice. Another challenging problem is that cryptography often focuses computation on bits; there is a long way to go from bit computations to practical computational problems, where a lot of real numbers are involved. We review important and recent work here.

Security community. Atallah et al. [1] considered the problem of outsourcing of scientific computations. They identified some useful techniques to disguise the problem and proposed practical protocols for these computations. However, the efficiency was not considerably focused and the verification of the returned result was not discussed. Recently, Wang et al. [7], [8] proposed two protocols for secure linear programming and linear equation outsourcing. Their protocols are reasonably efficient and can be deployed in practice immediately. Their protocol enables the client to get a remarkable speedup when using the outsourcing protocol to solve linear programming problems. Our protocol here handles linear regression problems, which are different from theirs. Thus, the speedup is also different; their speedup is better than that of this paper, but both are remarkable. This paper also adopts their framework to work on computation outsourcing. The transformation approach in this paper has also been employed to conquer other outsourcing problems [12], [13], [20], [21], [22], [23]. However, we emphasize that our linear regression problem here is quite different from these work in several aspects. First, the linear regression outsourcing problem has never been studied before while this problem is very important. For example, linear regression is a commonly used tool in the machine learning area. Second, the linear transformation approach is also employed in previous work; however, the choice of linear transformations is tricky and not straightforward for the linear regression outsourcing problem. The reason lies in the fact that there are more constrains than variables in a linear regression problem, which can also be found in our design in this paper. We tried several approaches in order to find a good solution. Third, we aim to construct an *efficient* protocol with sufficient security. It is reasonable that adding more complexity in a protocol makes the protocol more secure. As mobile computation is becoming more and more popular in recent years, there are also a lot of applications for mobile computation outsourcing. Ramakrishnan et al. [24] identified some interesting computational problems on mobile phones, for which outsourcing can bring sufficient performance gains. These problems are not well solved and thus can be explored in the future.

Cryptography community. There are two general but highly non-trivial protocols for outsourcing any reasonable computations, which can be done by a Turing machine in polynomial time, proposed in the cryptography community [6], [25]. The first solution [6] expresses a computation using a Boolean circuit; then mask this circuit using Yao's gabled circuit technique [26], [27] and encrypt the garbled circuit using a fully homomorphic encryption algorithm [28]; later, the encrypted circuit is sent to the cloud for solving the outsourced problem homomorphically and the final result is returned back; finally, the client recovers the result and checks its correctness using the secret key. The other solution [25] employs the idea of interactive proof systems [29]. To outsource a problem, the client prepares a few such problems with a half being the same problem to be solved; then the client encrypts these problems using a fully homomorphic encryption algorithm and sends them to the cloud for solutions; the cloud solves these problems homomorphically and returns the answers back; finally, the client decrypts to get an answer and checks its correctness. The two solutions are quite elegant; however, they are far from practice because of the efficiency issue. Expressing a computational problem into a Boolean circuit is not easy and the fully homomorphic encryption is quite complicated and inefficient, which also requires to express a computation into a circuit. Barbosa and Farshim [30] extended two-party computation outsourcing and studied delegable homomorphic encryption to support secure computation outsourcing. Parno et al. [31] also built a practical and special system to support verifiable computation outsourcing based on a quadratic programme encoding computations.

7 Concluding Remarks

In this paper, we proposed two protocols for the secure and efficient linear regression outsourcing problem. One is more secure while the other is more efficient; thus, the applications can choose either one based on their requirements. The two protocols can be deployed in practice immediately. It is interesting to find some other new solutions to this problem. But as discussed in this paper, there is not much room for improvement. However, new ideas in the new solutions are more important to the whole community. We believe the more important future research direction should be to identify practical problems for which outsourcing can bring a lot of benefits in the mobile and cloud computing age; and then propose protocols and software to solve them. Sometimes, for specific problems, we can employ their special structures to ease our design, which is a huge difference from the general but extremely inefficient solutions in the cryptography community.

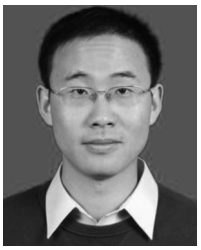
Acknowledgments

This research was supported by the Natural Science Foundation Project of CQ CSTC (No. cstc2013jcyjA40001), the Fundamental Research Funds for the Central Universities (No. CDJZR13185501), the National Nature Science Foundation of China under Grant 61170283, the National High-Technology Research and Development Program ("863" Program) of China under Grant 2013AA01A212, and the Ministry of Education in the New Century Excellent Talents Support Program under Grant NCET-12-0649.

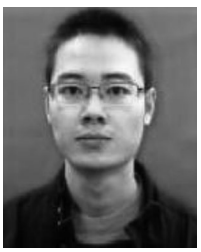
References

[1] M. Atallah, K. Pantazopoulos, J. Rice, and E. Spafford, "Secure outsourcing of scientific computations," *Adv. Comput.*, vol. 54, pp. 215–272, 2002.
[2] Wikipedia. (2012). Seti@home—Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/w/index.php?title=SETI@home
[3] Wikipedia. (2012). Folding@home—Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/w/index.php?title=Folding@home
[4] Wikipedia. (2014). Wireless sensor network [Online]. Available: http://en.wikipedia.org/wiki/Sensor_network
[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[6] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, 2010, pp. 465–482.
[7] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. IEEE 30th Conf. Comput. Commun.*, 2011, pp. 820–828.
[8] C. Wang, K. Ren, J. Wang, and K. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 549–558.
[9] G. A. F. Seber and A. J. Lee, *Linear Regression Analysis*, 2nd ed. Hoboken, NJ, USA: Wiley, 2003.
[10] C. Wang, K. Ren, J. Wang, and Q. 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.
[11] F. Chen, T. Xiang, and Y. Yang. (2014, Mar.). Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud. *J. Parallel Distrib. Comput.* [Online]. *74(3)*, pp. 2141–2151 Available: http://dx.doi.org/10.1016/j.jpdc.2013.11.007
[12] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," *Optim. Lett.*, vol. 6, pp. 1–6, 2012.
[13] Y. Hong and J. Vaidya, "An inference-proof approach to privacy-preserving horizontally partitioned linear programs," *Optim. Lett.*, vol. 8, no. 1, pp. 267–277, 2014.
[14] Wikipedia. (2012). Cryptographically secure pseudorandom number generator—Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/w/index.php?title=Cryptographically_secure_pseu dorandom_number_generator
[15] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM J. Comput.*, vol. 13, no. 4, pp. 850–864, 1984.
[16] R. Impagliazzo, L. Levin, and M. Luby, "Pseudo-random generation from one-way functions," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, 1989, pp. 12–24.
[17] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1995, pp. 92–111.
[18] UC Irvine Machine Learning Repository. (2012) Energy efficiency data set [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
[19] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy Buildings*, vol. 49, pp. 560–567, 2012.
[20] W. Du, "A study of several specific secure two-party computation problems," Ph.D. dissertation, Dept. Comput. Sci., Purdue Univ. West Lafayette, IN, USA, 2001.
[21] J. Vaidya, "Privacy-preserving linear programming," in *Proc. ACM Symp. Appl. Comput.*, 2009, pp. 2002–2007.
[22] A. Bednarz, N. Bean, and M. Roughan, "Hiccups on the road to privacy-preserving linear programming," in *Proc. 8th ACM Workshop Privacy Electron. Soc.*, 2009, pp. 117–120.
[23] Y. Hong, J. Vaidya, and H. Lu, "Secure and efficient distributed linear programming," *J. Comput. Security*, vol. 20, no. 5, pp. 583–634, 2012.
[24] S. Ramakrishnan, R. Reutiman, H. Iverson, S. Lian, A. Chandra, and J. Weissman, "Early experience with mobile computation outsourcing," *Dept. Comput. Sci. Eng.*, Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. 10-020, 2010.
[25] K. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, 2010, pp. 483–501.
[26] A. Yao, "How to generate and exchange secrets," in *Proc. IEEE 27th Annu. Symp. Found. Comput. Sci.*, 1986, pp. 162–167.
[27] A. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.
[28] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.
[29] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. 17th Annu. ACM Symp. Theory Comput.*, 1985, pp. 291–304.
[30] M. Barbosa and P. Farshim, "Delegatable homomorphic encryption with applications to secure outsourcing of computation," in *Proc. 12th Conf. Topics Cryptol.*, 2012, pp. 296–312.
[31] B. Parno, C. Gentry, J. Howell, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 238–252.

Fei Chen received the PhD degree in computer science and engineering from the Chinese University of Hong Kong in 2014. His research interests include network and information security, and data protection and privacy.



Tao Xiang received the BEng, MS, and PhD degrees in computer science from Chongqing University, China, in 2003, 2005, and 2008, respectively. He is currently an associate professor at Chongqing University. His research interests include cloud security, wireless security, multimedia security, and cryptography. He has published more than 40 papers on international journals and conferences. He also served as a referee for numerous international journals.



Xinyu Lei received the BEng degree in computer science from Chongqing University, China, in 2010. He is currently working towards the MS degree in computer science at Chongqing University. He was once a visiting scholar in Texas AM University at Qatar. His research interests include information security and algorithms.

Jianyong Chen received the PhD degree from the City University of Hong Kong, Hong Kong, China, in 2003. He is a professor in the Department of Computer Science and Technology, Shenzhen University. His research interest include information security and artificial intelligence. He was with the ZTE Corporation as senior engineer of network technology from 2003 to 2006. After that, he was with the Shenzhen University. Since 2004, he has been a vice-chairman of International Telecommunication Union-Telecommunication (ITU-T) SG17 and editor of three recommendations developed in ITU-T SG17. He has published more than 40 papers and applied more than 30 patents in the field of artificial intelligence and information security.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

The author has requested enhancement of the downloaded file. All in-text references underlined in blue are linked to publications on ResearchGat