Input and Output Privacy-Preserving Linear Regression

Aono, Yoshinori
Hayashi, Takuya
Phong, Le Trieu
Wang, Lihua

(Citation)
IEICE Transactions on Information and Systems,E100.D(10):2339-2347

(Issue Date)
2017-10

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90004874

PAPER Special Section on Security, Privacy and Anonymity in Computation, Communication and Storage Systems

Input and Output Privacy-Preserving Linear Regression*

Yoshinori AONO[†a)], **Takuya HAYASHI**[†,††b)], ***Members***, **Le Trieu PHONG**[†c)], ***Nonmember***, ***and*** **Lihua WANG**[†d)], ***Member***

SUMMARY We build a privacy-preserving system of linear regression protecting both *input data secrecy* and *output privacy*. Our system achieves those goals simultaneously via a novel combination of homomorphic encryption and differential privacy dedicated to linear regression and its variants (ridge, LASSO). Our system is proved scalable over cloud servers, and its efficiency is extensively checked by careful experiments.

key words: *differential privacy, homomorphic encryption, LWE assumption, linear regression*

1. Introduction

1.1 Background

Imaginatively, the storage and computation on the cloud can be seen as storing data and performing computations on a huge and globally available "machine". Formally, a definition of cloud computing is given by NIST in [18] saying that it is a "*model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"

Among others, machine learning is benefited from the development of cloud computing. Indeed, current commercial platforms such as the Amazon Machine Learning [1] or the Google Cloud Machine Learning Platform [2] allows clients to upload data to cloud servers to do various machine learning tasks.

The benefits of cloud computing come with threats, typically one of which is that plain data stored in the cloud may be accessed unwillingly. Promisingly, homomorphic encryption can balance the situation, as it enables *input data secrecy* and the computations over the data even in encrypted form, on which the discussion can go back far to Rivest, Adleman, and Dertouzos [23] in 1978. Specifically, a client can store encrypted data on the cloud to enjoy the service, but at the same time can ensure that no useful information is leaked to the storage.

On the other hand, *output privacy* refers to the protection of the output of a mining system against inference attacks, namely those guessing useful information on an individual data given the output. As a measure for output privacy, differential privacy [9] ensures that the presence of any single data item will not much affect the output.

1.2 Our Contributions

Results. We build a system for linear regression and its ridge and LASSO** variants protecting *both* input (i.e. training) data secrecy *and* output privacy, simultaneously. We instantiate our system with a modern LWE-based homomorphic public key encryption scheme, which is a variant of [16], [22].

Our system is extremely fast in time and modest in communication. For example, on real datasets, our system finishes in milliseconds with less than 100 kilobytes of communication (see Fig. 3 for details). For large datasets, the efficiency is still reasonable (see Fig. 4).

To position our work into the literature, let us compare with [12], [13], [20], [26]. Previous systems accomplished either input data secrecy as in [12], [13], [20] or output privacy as in [26] independently, but not both as showed in the following table.

Technical outlines. Theorems 2 and 3 are considered the technical results of this paper, and are achieved by a combination dedicated to linear regression of homomorphic encryption and differential privacy.

Along the way of designing our system, we need a few tricks. First, we view linear regression as an optimization problem, not as a matrix inversion problem. Second, each data source slightly pre-processes its data before encryption.

Table 1 Systems for privacy-preserving linear regression.

System	Input data secrecy	Output privacy
[12], [13], [20]	yes	no
[26]	no	yes
Ours	**yes**	**yes**

**Least Absolute Shrinkage and Selection Operator.

Manuscript received January 13, 2017.
Manuscript revised May 23, 2017.
Manuscript publicized July 21, 2017.

†The authors are with National Institute of Information and Communications Technology (NICT), Koganei-shi, 184–8795 Japan.

††The author is with Graduate School of Engineering, Kobe University, Kobe-shi, 657–8501 Japan.

*This work is partially supported by JST CREST #JPMJCR168A.

a) E-mail: aono@nict.go.jp
b) E-mail: t-hayashi@eedept.kobe-u.ac.jp
c) E-mail: phong@nict.go.jp (Corresponding author)
d) E-mail: wlh@nict.go.jp

DOI: 10.1587/transinf.2016INP0019

Third, each pre-processed data item is packed in *one* ciphertext using the LWE-based encryption scheme. Fourth, we arrange the encrypted data items so that the server can handle them correctly. Finally, the client with decryption key does some light post-processing work.

1.3 More Related Works

Linear regression has been used successfully in medical research [24] with geographically distributed data sources (concretely, 21 research groups from 9 countries and 4 continents). Our system for privacy preserving linear regression can help protecting data in this kind of medical research scenarios, particularly when the number of data N_{data} and data dimension d get large. Indeed, getting sufficient data is very important for better results in medical prediction. In the dataset of [24], the number of data records $N_{\text{data}} = 5700$, the data dimension $d = 17$, and yet they wrote "*... we did not have sufficient data ... to include potentially important factors such as smoking status, vitamin K intake, or alcohol consumption, as well as other genetic factors...*".

Linear regression (whose outputs are continuous) is related but different from logistic regression (whose outputs are discrete). Various privacy-preserving systems for logistic regression have been built in [5]–[7], [19].

2. Preliminaries

Let $\mathbb{Z}_{(0,s)}$ be the discrete Gaussian distribution over the integers $\mathbb{Z}$, with mean 0 and deviation $s > 0$. The mark $\xleftarrow{g}$ is for "sampling at random from a discrete Gaussian" set, so that $x \xleftarrow{g} \mathbb{Z}_{(0,s)}$ means x appearing with probability proportional to $\exp(-\pi x^2/s^2)$. Below, $\xleftarrow{\$}$ means "sampling at random uniformly". Also, $\mathbb{Z}_q \subset (-q/2, q/2]$ is the set of integers of centered modulus q.

2.1 Learning with Errors (LWE)

Related to the decision LWE assumption LWE(n, s, q), where n, s, q depend on the security parameter λ, consider matrix $A \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, vectors $r \xleftarrow{\$} \mathbb{Z}_q^{m\times 1}$, $x \xleftarrow{g} \mathbb{Z}_{(0,s)}^{n\times 1}$, $e \xleftarrow{g} \mathbb{Z}_{(0,s)}^{m\times 1}$. Then vector $Ax + e$ is computed over $\mathbb{Z}_q$. Define the following advantage of a poly-time probabilistic algorithm $\mathcal{D}$:

$$\mathbf{Adv}_{\mathcal{D}}^{\text{LWE}(n,s,q)}(\lambda) = \left|\Pr[\mathcal{D}(A, Ax + e) \to 1] - \Pr[\mathcal{D}(A, r) \to 1]\right|.$$

The LWE assumption asserts that $\mathbf{Adv}_{\mathcal{D}}^{\text{LWE}(n,s,q)}(\lambda)$ is negligible as a function of λ.

2.2 Additively Homomorphic Encryption

Definition 1: Public key homomorphic encryption (PHE) schemes consist of the following (possibly probabilistic) poly-time algorithms.

- $\mathsf{ParamGen}(1^\lambda) \to pp$: λ is the security parameter and the public parameter pp is implicitly fed in following algorithms.
- $\mathsf{KeyGen}(1^\lambda) \to (pk, sk)$: pk is the public key, while sk is the secret key.
- $\mathsf{Enc}(pk, m) \to c$: probabilistic encryption algorithm produces c, the ciphertext of message m.
- $\mathsf{Dec}(sk, c) \to m$: decryption algorithm returns message m encrypted in c.
- $\mathsf{Add}(c, c')$: In Add, for ciphertexts c and c', the output is the encryption of plaintext addition c_{add}.
- $\mathsf{DecA}(sk, c_{\text{add}})$: decrypting c_{add} to obtain an addition of plaintexts.

Definition 2: With respect to a PHE scheme as in Definition 1, consider the following game between an adversary $\mathcal{A}$ and a challenger:

- **Setup.** The challenger creates pp and key pairs (pk, sk). Then pp and pk are given to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ chooses two plaintexts m_0, m_1 of the same length, then submits them to the challenger, who in turn takes $b \in \{0, 1\}$ randomly and computes $C^* = \mathsf{Enc}(pk, m_b)$. The challenge ciphertext C^* is returned to $\mathcal{A}$, who produces a bit b'.

A PHE scheme is CPA-secure if the advantage

$$\mathbf{Adv}_{\mathcal{A}}^{\text{cpa}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

is negligible in λ.

3. Building Block: A Homomorphic Encryption Scheme

We will use the following encryption scheme, which is a variant of [22] and relying on the idea of [16] for reducing the public key size.

- $\mathsf{ParamGen}(1^\lambda)$: Fix $q = q(\lambda) \in \mathbb{Z}^+$ and $l \in \mathbb{Z}^+$. Fix $p \in \mathbb{Z}^+$ so that $\gcd(p, q) = 1$. Return $pp = (q, l, p)$.
- $\mathsf{KeyGen}(1^\lambda, pp)$: Take $s = s(\lambda, pp) \in \mathbb{R}^+$ and $n = n(\lambda, pp) \in \mathbb{Z}^+$. Take $R, S \xleftarrow{g} \mathbb{Z}_{(0,s)}^{n\times l}$, $A \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$. Compute $P = pR - AS \in \mathbb{Z}_q^{n\times l}$. Return the public key $pk = (A, P, n, s)$, and the secret key $sk = S$.
- $\mathsf{Enc}(pk, m \in \mathbb{Z}_p^{1\times l})$: Take $e_1, e_2 \xleftarrow{g} \mathbb{Z}_{(0,s)}^{1\times n}$, $e_3 \xleftarrow{g} \mathbb{Z}_{(0,s)}^{1\times l}$. Compute $c_1 = e_1 A + pe_2 \in \mathbb{Z}_q^{1\times n}$, $c_2 = e_1 P + pe_3 + m \in \mathbb{Z}_q^{1\times l}$. Return $c = (c_1, c_2)$.
- $\mathsf{Dec}(S, c = (c_1, c_2))$: Compute $\overline{m} = c_1 S + c_2 \in \mathbb{Z}_q^{1\times l}$. Return $m = \overline{m} \bmod p$.
- $\mathsf{Add}(c, c')$: Compute $c_{\text{add}} = c + c' \in \mathbb{Z}_q^{1\times(n+l)}$. Return c_{add}.
- $\mathsf{DecA}(S, c_{\text{add}})$: Verify $c_{\text{add}} \in \mathbb{Z}_q^{1\times(n+l)}$. Return the decryption $\mathsf{Dec}(S, c_{\text{add}})$

We prove security of the above scheme according to Definition 2.

Theorem 1 (CPA security): The above additively homomorphic encryption scheme is CPA-secure under the LWE

assumption. Specifically, for any poly-time adversary $\mathcal{A}$, there is an algorithm $\mathcal{D}$ of essentially the same running time such that

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{A}}(\lambda) \le (l+1)\cdot \mathbf{Adv}^{\mathrm{LWE}(n,s,q)}_{\mathcal{D}}(\lambda).$$

Proof 1: The proof follows [16] closely. First, we modify the original game in Definition 2 by providing $\mathcal{A}$ with a random $P \xleftarrow{\$} \mathbb{Z}_q^{n\times l}$. Namely $P = pR - AS \in \mathbb{Z}_q^{n\times l}$ is turned to a random matrix. This is indistinguishable to $\mathcal{A}$ thanks to the LWE assumption with secret vectors as l columns of S. More precisely, we need the condition $\gcd(p,q) = 1$ to reduce $P = pR - AS \in \mathbb{Z}_q^{n\times l}$ to the LWE form. Indeed, $p^{-1}P = R + (-p^{-1}A)S \in \mathbb{Z}_q^{n\times l}$. As A is random, $A' = -p^{-1}A \in \mathbb{Z}_q^{n\times n}$ is also random. Therefore, $P' = p^{-1}P \in \mathbb{Z}_q^{n\times l}$ is random under the LWE assumption which in turn means P is random as claimed.

Second, the challenge ciphertext $c^* = e_1[A|P] + p[e_2|e_3] + [0|m_b]$ is turned to random. This relies on the LWE assumption with secret vector e_1, while also basing on the first modification so that $[A|P]$ is a random matrix. Here, the condition $\gcd(p,q) = 1$ is also necessary as above. Thus b is perfectly hidden after this change. The factor $l+1$ is due to l uses of LWE in changing P and 1 use in changing c^*.

The following properties of the variant are worth mentioning:

• **Arbitrarily poly-packing of data.** Due to Theorem 1, there are rooms for choosing parameters

$$(l, p)$$

in the message space $\mathbb{Z}_p^l$ to fit each specific application, as they are not tightly related to parameters deciding security

$$(n, s, q).$$

For example, as l (message length) and n (LWE dimension) can be different, we can set $(l, n) = (16128, 3530)$ as in our system in Sect. 4.

• **Encode real numbers.** Real numbers of precision $\mathsf{prec} = (L+\ell+1)$ (e.g. = 64) bits are encoded via *signed* bit vectors, namely $a, b \in \mathbb{R}$ are expressed as

$$a = \sum_{k=-L}^{\ell} a_k 2^k, \text{ and } b = \sum_{k=-L}^{\ell} b_k 2^k$$

in which all $a_k \in \{0,1\}$ if $a \ge 0$ and $a_k \in \{0,-1\}$ if $a < 0$, and likewise for b_k. For $a \in \mathbb{R}$, define vectors

$$\begin{aligned} \mathsf{Pow}(2, L, \ell) &= [2^{-L} \cdots 2^0 \cdots 2^{\ell}] \in \mathbb{Z}^{1\times \mathsf{prec}}, \\ \mathsf{Bits}(a) &= [a_{-L} \cdots a_0 \cdots a_{\ell}] \in \{-1, 0, 1\}^{1\times \mathsf{prec}}. \end{aligned} \tag{1}$$

• **Encrypt real numbers.** For a real number a precision prec, naturally define its encryption as

$$E_a = \mathsf{Enc}\big(\mathsf{Bits}(a)\big) \in \mathbb{Z}_q^{1\times(n+\mathsf{prec})} \tag{2}$$

where Enc is the encryption algorithm above. The decryption of E_a using the DecA algorithm yields $\mathsf{Bits}(a) \in \mathbb{Z}_p^{1\times\mathsf{prec}}$. If $p \ge 2$, there will be no wrap-around (overflow) and $\mathsf{Bits}(a) \in \mathbb{Z}^{1\times\mathsf{prec}}$ is obtained, yielding the real number

$$a = \langle \mathsf{Pow}(2, L, \ell), \mathsf{Bits}(a)\rangle \in \mathbb{R},$$

where $\langle\cdot,\cdot\rangle$ is the inner product.

When multiple additions are done over ciphertexts, the plaintexts are also added due to the homomorphic property. Even if each plaintext coordinate is in $\{-1, 0, 1\}$, the sum of all N_{data} plaintexts can make each summed coordinate grow into the range $[-N_{\mathrm{data}}, N_{\mathrm{data}}]$. To prevent wrapping-around modulo p, we need $[-N_{\mathrm{data}}, N_{\mathrm{data}}] \subset \mathbb{Z}_p$ so that we need $p/2 > N_{\mathrm{data}}$. As we would like to deal with big N_{data} of order 10^8, we choose $p = 2^{30} + 1$. See also Sect. 4.6.

4. Privacy Preserving Linear, Ridge, and LASSO Regressions

This section contains our main results.

4.1 Basic Model: Input Data Secrecy (Rivest, Adleman, Dertouzos 1978)

Following [11], [23], consider the scenario of outsourced computation using homomorphic encryption: a client outsources its data to a cloud server for computation and storage, but does not want to leak any information to the cloud server. In the following we recap the details.

Model outline. The general picture of the protocol is in Fig. 1. We use $\mathbf{E}_{pk}(data^{(i)})$ $(1 \le i \le N_{\mathrm{data}})$ to represent the encryption of the data under a public key pk. Anyone can contribute data; e.g., the data can be

- from the client itself, or
- possibly from various geographically distributed data contributors.

After receiving the encrypted data, using homomorphic property of $\mathbf{E}_{pk}$, the computing server does necessary computations and sends the output $\mathbf{E}_{pk}(\Theta)$ to the data analyst, from which Θ is recovered by decryption, and the final result θ^* is obtained (from Θ). Note that, naively $\Theta = \theta^*$ but it is not a must. For example, in [15], [19], $\Theta = (\theta_1^*, \theta_2^*) \in \mathbb{Z}^2$ and $\theta^* = \theta_1^*/\theta_2^* \in \mathbb{R}$, as current homomorphic encryption schemes do not support division directly. Indeed, required is that θ^* can be *efficiently derived* from Θ. We will exploit this property in depth in designing our system.

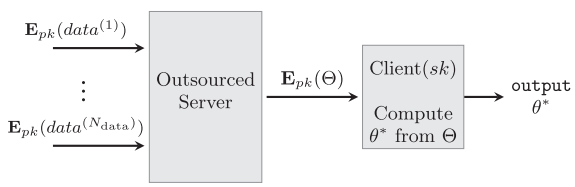

Fig. 1 Basic model of secure outsourced computation (Rivest, Adleman, Dertouzos 1978).

Key generation. The client generates the public and secret key pair (pk, sk) and publicly distributes pk.

Data encryption. Data from the client or many contributors is encrypted and sent to the outsourced server. We assume that these encryption and uploading processes are always correctly executed.

Threat and protection goal. The outsourced server is assumed *honest-but-curious*: it is curious on any information from the data, and yet is honest in instructed computations. This curious nature of the server is considered a threat. The protection goal of our protocol in Fig. 1 is to hide any information of the data from the server.

This honest-but-curious assumption is reasonable to model an economically motivated cloud service provider: it wants to provide excellent service for a successful business, but would be interested in any extra available information. On the other hand, a malicious cloud service provider can mishandle calculations, delete data, refuse to return results, collude with other parties etc. Nevertheless, it is likely to be caught in most of these malicious behaviors, and hence harms its reputation in business. Therefore, we will stick to the assumption of honest-but-curious server.

4.2 Extended Model: Input Data Secrecy + Differential Privacy

Figure 2 extends Fig. 1 by additionally considering differential privacy, namely output privacy in the sense that the output θ^* reveals nothing meaningful from any specific input $data^{(i)}$. For that purpose, we view the server and the client and their interactions as an interactive mechanism $\mathcal{I} = (\text{Server}, \text{Client})$ with inputs $data^{(1)}, \ldots, data^{(N_{\text{data}})}$ and output θ^*. In Fig. 2,

- The mechanism $\mathcal{I}$ encrypts its input data and feeds the resulting ciphertexts to the outsourced server.
- The server and the client behave as in Sect. 4.1.

Let D and D_* be two neighbor databases, namely

$$D = \left\{data^{(1)}, \ldots, data^{(N_{\text{data}}-1)}, data^{(N_{\text{data}})}\right\}$$
$$D_* = \left\{data^{(1)}, \ldots, data^{(N_{\text{data}}-1)}, data_*^{(N_{\text{data}})}\right\}$$

differing only at the final item, i.e., $data^{(N_{\text{data}})} \neq data_*^{(N_{\text{data}})}$.

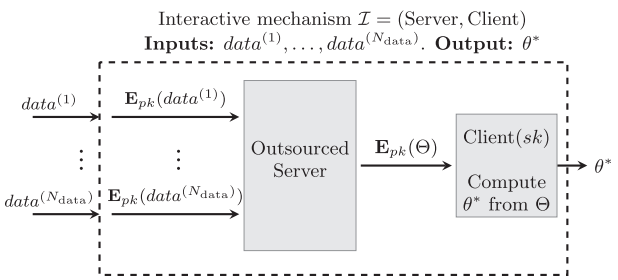

Fig. 2 Extended model of encryption plus differential privacy (for input data secrecy and output privacy).

Denote $\mathcal{I}(D) \Rightarrow \theta^*$ as a shorthand for saying that the interactive mechanism $\mathcal{I}$ in Fig. 2 outputs θ^* with input D, we have the following definition.

Definition 3: (Differential privacy) The interactive mechanism $\mathcal{I} = (\text{Server}, \text{Client})$ in Fig. 2 has ϵ-differential privacy, if and only if

$$\Pr[\mathcal{I}(D) \Rightarrow \theta^*] \leq \exp(\epsilon) \cdot \Pr[\mathcal{I}(D_*) \Rightarrow \theta^*]$$

for any output θ^*.

The above definition naturally extends the definition in the literature [9] in which $\mathcal{I}$ is a randomized algorithm. Intuitively, the definition ensures that the change in any single data item will not much affect the final output.

4.3 Tweaking the Cost Functions of Linear Regression

Cost function of linear regression. Data dimension, namely the number of features, is denoted by d. The number of data items, namely the training set size, is denoted by N_{data}. Each training data item is denoted as $(x^{(i)}, y^{(i)})$ for $1 \leq i \leq N_{\text{data}}$ in which $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}$. As $x^{(i)} \in \mathbb{R}^d$, we can explicitly write it as $x^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$. Linear hypothesis is a linear function $h_\theta : \mathbb{R}^d \to \mathbb{R}$; namely for variable $X = (X_1, \ldots, X_d) \in \mathbb{R}^d$, and $\theta = (\theta_0, \ldots, \theta_d) \in \mathbb{R}^{d+1}$,

$$h_\theta(X) = \sum_{j=1}^{d} \theta_j X_j + \theta_0.$$

The cost function over training data is usually defined as

$$J_{\text{cost}}(\theta) = \frac{1}{2N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2. \tag{3}$$

Tweaking the cost function for our system. For $0 \leq k, j \leq d$, let

$$A_{k,j} = \sum_{i=1}^{N_{\text{data}}} x_k^{(i)} x_j^{(i)} \tag{4}$$

$$B_j = \sum_{i=1}^{N_{\text{data}}} y^{(i)} x_j^{(i)} \tag{5}$$

$$C = \sum_{i=1}^{N_{\text{data}}} \left(y^{(i)}\right)^2 \tag{6}$$

the cost function in (3) can be expressed as a convex quadratic function $J_{\text{cost}} : \mathbb{R}^{d+1} \to \mathbb{R}$,

$$J_{\text{cost}}(\theta) = \frac{1}{2N_{\text{data}}}\Big(\sum_{k,j=0}^{d} (\theta_k \theta_j) A_{k,j} - 2 \sum_{j=0}^{d} \theta_j B_j + C \Big) \tag{7}$$

and the task of linear regression is to produce

$$\theta^* = \text{argmin}_\theta J_{\text{cost}}(\theta),$$

using any method of numerical optimization [21] (e.g., gradient descent, conjugate gradient, BFGS, L-BFGS). As J_{cost}

at (7) has $d+1$ variables, the cost to produce the minimizer θ^* depends on d.

Extensions to ridge and LASSO regression. The above backgrounds and tweak can be straightforwardly applied to ridge regression and LASSO, where the cost functions are defined respectively as $J_{\text{cost}}(\theta) + \mu\|\theta\|_2^2$ (ridge) and $J_{\text{cost}}(\theta) + \mu\|\theta\|_1$ (LASSO) for some parameter $\mu \in \mathbb{R}$. When $\mu = 0$, those regressions become exactly linear regression. When $\mu > 0$, a solution θ with small norm is preferable. As the added term $\mu\|\theta\|_2^2$ or $\mu\|\theta\|_1$ does not vary with data items, the following subsections remain almost unchanged even with those variants. The only change is in the client side, and we will note that in place later.

4.4 Basic System: Achieving Input Data Secrecy

For clarity, we first present our basic system of securely outsourced linear regression under the model outlined in Fig. 1. The full system under the model of Fig. 2 will be described in the next subsection.

The work flow in our system is described in four steps (0)–(3) as follows.

(0) Data format. Each data item i ($1 \le i \le N_{\text{data}}$) is represented as

$$data^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)}, y^{(i)}) \in \mathbb{R}^{d+1}.$$

Via normalization, without loss of generality, we assume that $-1 \le x_1^{(i)}, \ldots, x_d^{(i)}, y^{(i)} \le 1$.

(1) Encryption at each data source (or client). The encryption $\mathbf{E}_{pk}(data^{(i)})$ from any data source (or the client) is done as follows:

• Compute $u_{k,j}^{(i)} = x_k^{(i)} x_j^{(i)}, v_j^{(i)} = y^{(i)} x_j^{(i)}, w^{(i)} = (y^{(i)})^2 \in \mathbb{R}$ for all $1 \le k, j \le d$, which is of $l_d = d(d+1)/2 + d + 1$ real numbers.

• Consider prec bit representations of all $x_1^{(i)}, \ldots, x_d^{(i)}, y^{(i)}$, and above $u_{k,j}^{(i)}$, $v_j^{(i)}$, $w^{(i)}$, concatenate them as one vector in $\{-1, 0, 1\}^{(d+1+l_d)\cdot\mathsf{prec}} \subset \mathbb{Z}_p^l$ for $l = (d+1+l_d)\cdot\mathsf{prec}$, and do the encryption

$$E^{(i)} = \mathsf{Enc}\left(\begin{array}{l}\mathsf{Bits}(x_1^{(i)}), \ldots, \mathsf{Bits}(x_d^{(i)}), \mathsf{Bits}(y^{(i)}), \\ \underbrace{\mathsf{Bits}(u_{k,j}^{(i)}), \ldots, \mathsf{Bits}(v_j^{(i)})}_{1\le k,j\le d}, \mathsf{Bits}(w^{(i)})\end{array}\right)$$

where Enc is the encryption algorithm in Sect. 3. The ciphertext result is a vector in $\mathbb{Z}_q^{1\times(n+(d+1+l_d)\cdot\mathsf{prec})}$, and is sent to the server.

(2) Outsourced server. Having all ciphertexts $E^{(i)}$, the cloud server computes the sum

$$E = \sum_{i=1}^{N_{\text{data}}} E^{(i)} \in \mathbb{Z}_q^{1\times(n+(d+1+l_d)\cdot\mathsf{prec})} \tag{8}$$

and sends the result to the client. The computational cost of (8) is $O(N_{\text{data}}d^2)$ arithmetic operations, yielding scalability when N_{data} or even d is large, e.g. $N_{\text{data}} = 10^8$, $d = 40$. Parallelization of (8) is also extremely easy on the server.

(3) Client (data analyst). The client decrypts E using algorithm DecA in Sect. 3 to obtain the following $O(d^2)$ sums over $\mathbb{Z}_p^{1\times\mathsf{prec}}$

$$\sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(x_1^{(i)}), \ldots, \sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(x_d^{(i)}), \sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(y^{(i)})$$
$$\text{and } \sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(u_{k,j}^{(i)}), \sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(v_j^{(i)}), \sum_{i=1}^{N_{\text{data}}} \mathsf{Bits}(w^{(i)})$$

from which the client, via multiplying with $\mathsf{Pow}(2, L, \ell)$ at (1), obtains the following real numbers

$$\sum_{i=1}^{N_{\text{data}}} x_1^{(i)}, \ldots, \sum_{i=1}^{N_{\text{data}}} y^{(i)} \in \mathbb{R}$$
$$\sum_{i=1}^{N_{\text{data}}} u_{k,j}^{(i)}, \sum_{i=1}^{N_{\text{data}}} v_j^{(i)} \in \mathbb{R}, \sum_{i=1}^{N_{\text{data}}} w^{(i)} \in \mathbb{R}$$

for all $1 \le k, j \le d$. These are exactly $A_{k,j}$, B_j, C for all $0 \le k, j \le d$ in (4), (5), and (6). Having these $O(d^2)$ coefficients of $J_{\text{cost}}(\theta)$ at (7), the client then finds $\theta^* = \text{argmin}_\theta J_{\text{cost}}(\theta)$ via optimization methods in numerical computing [21] such as the BFGS method whose costs depend on d.

For ridge (likewise, LASSO) regression,

$$\theta^* = \text{argmin}_\theta(J_{\text{cost}}(\theta) + \mu\|\theta\|_2^2)$$

at the client side. Other methods (e.g., gradient descent, conjugate gradient, L-BFGS) can also be used at the client to minimize $J_{\text{cost}}(\theta)$.

Theorem 2 (Input data secrecy): The above system leaks nothing on the data to the outsourced server, assuming that the encryption scheme is CPA-secure.

Proof 2: The server receives and computes over encrypted data. The encryption scheme satisfies CPA security (via Theorem 1), so that no information on the plain data can be computationally leaked to the server.

Communication cost between server and client. The cost of sending the computed result is only the size of E at (8), which is in bits

$$\Big(n + (d+1+l_d)\mathsf{prec}\Big)\log_2 q$$
$$= \Big(n + O(d^2)\mathsf{prec}\Big)\log_2 q. \tag{9}$$

Moreover, in the store phase, as $E^{(i)}$ has the same size of E at (8), the cost of each store is also (9).

4.5 Full System: Achieving Input Data Secrecy and Differential Privacy

Differential privacy intuitively ensures that the output leaks almost no information on each single data item.

Difficulty: the Laplace mechanism is not applicable. To

add differential privacy, the usual approach is to use Laplace mechanism [9], namely adding Laplace noise to the output θ^*. That approach will not work for linear regression since it is hard to determine parameters for the Laplace noises; or equivalently it is hard to compute the sensitivity of the algorithm finding the minimizer θ^* of $J_{\text{cost}}(\theta)$. This difficulty applies also to [13], [20] for the same reason.

Our approach. We make use of the functional mechanism [26]. Namely, we add noises of Laplace distribution into the coefficients $A_{k,j}$, B_j, and C of the cost function at (7), while keeping those coefficients encrypted.

Below gives more details. Recall that, a noise $x \in \mathbb{R}$ has $\text{Lap}(\sigma)$ distribution if its probability density function is $\frac{1}{2\sigma}\exp(-|x|/\sigma)$. To obtain ϵ-differential privacy, the only change we need is the computation of the client. Namely, the client itself generates Laplace noises from the $\text{Lap}(n_d/\epsilon)$ distribution, where $n_d = O(d^2)$ (more precisely, $n_d = d^2 + 3d + 4$ as showed below), and does the following computation

$$\begin{aligned}\overline{A}_{k,j} &= A_{k,j} + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\\ \overline{B}_j &= B_j + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\\ \overline{C} &= C + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\end{aligned} \tag{10}$$

for all $0 \le k, j \le d$. The client then uses these perturbed $\overline{A}_{k,j}$, $\overline{B}_j$, $\overline{C}$ instead of $A_{k,j}$, B_j, C as before in Sect. 4.4.

Theorem 3 (Differential privacy): With the change in (10), the system in Sect. 4.4 satisfies ϵ-differential privacy (Definition 3).

Proof 3: It is necessary to show that, for any output θ^*,

$$\Pr[\mathcal{I}(D) \Rightarrow \theta^*] \le \exp(\epsilon) \cdot \Pr[\mathcal{I}(D_*) \Rightarrow \theta^*].$$

The final computation $\theta^* = \text{argmin}_\theta J_{\text{cost}}(\theta)$ at the client is deterministic, and the function $J_{\text{cost}}(\theta)$ is determined by its its coefficients $(A_{k,j}, B_j, C)$ at (4), (5), (6). Therefore, θ^* is determined by the coefficients $(A_{k,j}, B_j, C)$ so that the above is equivalent to proving

$$\frac{\Pr[\mathcal{I}(D) \Rightarrow (A_{k,j}, B_j, C)\forall k, j]}{\Pr[\mathcal{I}(D_*) \Rightarrow (A_{k,j}, B_j, C)\forall k, j]} \le \exp(\epsilon). \tag{11}$$

When a training set D is used, we make D explicit in those coefficients as

$$\begin{aligned}A_{k,j,D} &= \sum_{i=1}^{N_{\text{data}}} x_{k,D}^{(i)} x_{j,D}^{(i)}\\ B_{j,D} &= \sum_{i=1}^{N_{\text{data}}} y_D^{(i)} x_{j,D}^{(i)}\\ C_D &= \sum_{i=1}^{N_{\text{data}}} \left(y_D^{(i)}\right)^2\end{aligned}$$

and likewise for those related to the training set D_*. Moreover,

$$\begin{aligned}\overline{A}_{k,j,D} &= A_{k,j,D} + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\\ \overline{B}_{j,D} &= B_{j,D} + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\\ \overline{C}_D &= C_D + \text{Lap}(n_d/\epsilon) \in \mathbb{R}\end{aligned} \tag{12}$$

are the coefficients perturbed by the mechanism $\mathcal{I}$. Similarly, $\overline{A}_{k,j,D_*}$, $\overline{B}_{j,D_*}$, $\overline{C}_{D_*}$ are defined with respect to D_*. Then (11) becomes

$$\frac{\Pr[(\overline{A}_{k,j,D}, \overline{B}_{j,D}, \overline{C}_D) = (A_{k,j}, B_j, C)\forall k, j]}{\Pr[(\overline{A}_{k,j,D_*}, \overline{B}_{j,D_*}, \overline{C}_{D_*}) = (A_{k,j}, B_j, C)\forall k, j]} \le \exp(\epsilon)$$

whose left hand side is $\delta_1 \cdot \delta_2 \cdot \delta_3$ where

$$\begin{aligned}\delta_1 &= \frac{\prod_{k,j} \exp(-\frac{\epsilon}{n_d}|A_{k,j,D} - A_{k,j}|)}{\prod_{k,j} \exp(-\frac{\epsilon}{n_d}|A_{k,j,D_*} - A_{k,j}|)}\\ \delta_2 &= \frac{\prod_{j} \exp(-\frac{\epsilon}{n_d}|B_{j,D} - B_j|)}{\prod_{j} \exp(-\frac{\epsilon}{n_d}|B_{j,D_*} - B_j|)}\\ \delta_3 &= \frac{\exp(-\frac{\epsilon}{n_d}|C_D - C|)}{\exp(-\frac{\epsilon}{n_d}|C_{D_*} - C|)}\end{aligned}$$

which simplifies to

$$\begin{aligned}\delta_1 &= \prod_{k,j} \exp\left(\frac{\epsilon}{n_d}(|A_{k,j,D_*} - A_{k,j}| - |A_{k,j,D} - A_{k,j}|)\right)\\ \delta_2 &= \prod_{j} \exp\left(\frac{\epsilon}{n_d}(|B_{j,D_*} - B_j| - |B_{j,D} - B_j|)\right)\\ \delta_3 &= \exp\left(\frac{\epsilon}{n_d}(|C_{D_*} - C| - |C_D - C|)\right)\end{aligned}$$

so that

$$\begin{aligned}\delta_1 &\le \prod_{k,j} \exp\left(\frac{\epsilon}{n_d}|A_{k,j,D_*} - A_{k,j,D}|\right)\\ \delta_2 &\le \prod_{j} \exp\left(\frac{\epsilon}{n_d}|B_{j,D_*} - B_{j,D}|\right)\\ \delta_3 &\le \exp\left(\frac{\epsilon}{n_d}|C_{D_*} - C_D|\right)\end{aligned}$$

Since datasets D and D_* differ at only the final N_{data}-th item,

$$\begin{aligned}|A_{k,j,D_*} - A_{k,j,D}| &= |x_{k,D_*}^{(N_{\text{data}})} x_{j,D_*}^{(N_{\text{data}})} - x_{k,D}^{(N_{\text{data}})} x_{j,D}^{(N_{\text{data}})}| \le 2\\ |B_{j,D_*} - B_{j,D}| &= |y_{D_*}^{(N_{\text{data}})} x_{j,D_*}^{(N_{\text{data}})} - y_D^{(N_{\text{data}})} x_{j,D}^{(N_{\text{data}})}| \le 2\\ |C_{D_*} - C_D| &= |(y_{D_*}^{N_{\text{data}}})^2 - (y_D^{N_{\text{data}}})^2| \le 2\end{aligned}$$

in which, via the normalization of the data, we assume that $x_{k,D_*}^{(N_{\text{data}})}$, $x_{k,D}^{(N_{\text{data}})}$, $x_{j,D_*}^{(N_{\text{data}})}$, $x_{j,D}^{(N_{\text{data}})}$, $y_{D_*}^{(N_{\text{data}})}$, $y_D^{(N_{\text{data}})}$ are real numbers in the range $[-1, 1]$. Therefore,

$$\begin{aligned}\delta_1 &\le \prod_{k,j} \exp\left(\frac{2\epsilon}{n_d}\right) = \exp\left(\frac{\epsilon d(d+1)}{n_d}\right)\\ \delta_2 &\le \prod_{j} \exp\left(\frac{2\epsilon}{n_d}\right) = \exp\left(\frac{2\epsilon(d+1)}{n_d}\right)\\ \delta_3 &\le \exp\left(\frac{2\epsilon}{n_d}\right)\end{aligned}$$

Table 2 Differences with previous works.

Ref.	Regression viewed as	Precondition for security	Ingredients (for input data secrecy)	Output privacy?
[13]	matrix inversion	N/A	Paillier enc. + secret sharing	no
[20]	matrix inversion	no GCP*-Server collusion	Paillier enc. + garbled circuit	no
Ours	cost function optimization	Party with decryption key is honest (model of [11], [23])	LWE-based hom. enc.	yes (Sect. 4.5)

*GCP stands for Garbled Circuit Provider.

so that

$$\delta_1 \cdot \delta_2 \cdot \delta_3 \le \exp\left(\frac{\epsilon(d^2+3d+4)}{n_d}\right) = \exp(\epsilon)$$

via setting $n_d = d^2 + 3d + 4$. This ends the proof.

Discussion: Privacy vs. Accuracy. Adding Laplace noises as in (10) gives us differential privacy (Theorem 3), but will decrease the accuracy of the system.

Fix the privacy budget ϵ, say $0.1 \le \epsilon \le 10$, the added noises have distribution $\mathrm{Lap}(n_d/\epsilon) = \mathrm{Lap}((d^2+3d+4)/\epsilon)$, meaning its deviation quadratically depends on data dimension d. Generally, if d is small, the added noises are small, so good accuracy will be obtained; reversely, large d gives bad accuracy. The accuracy is only depends on data dimension d, but not the number of data records N_{data}, so that we can even set large $N_{\text{data}} = 10^8$ as in the next section. It is worth mentioning that the trade-off between privacy and accuracy is well-known, as already examined in [10], [26].

4.6 Our System Costs and Comparisons

Parameter selections. To handle big N_{data}, we choose $q = 2^{114}$, $p = 2^{30} + 1$, $s = 8.0$, and $n = 3530$ (for 128-bit security). As $p = 2^{30} + 1$, the number of data records N_{data} can be as large as $\lfloor p/2 \rfloor$ ($\approx 10^{8.72}$). In other words, no overflow in additions (of real numbers) will occur as long as the number of ciphertexts received by the cloud server is less than $\lfloor p/2 \rfloor$. For smaller N_{data}, the parameters can be smaller. The bit security is estimated via existing attacks on LWE in [4], [8], [16], [17]. The plaintext length

$$l = (d + 1 + l_d)\mathsf{prec} = \frac{(d+1)(d+4)}{2} \cdot \mathsf{prec}$$

is sufficient to handle datasets of dimension d and real numbers of $\mathsf{prec} = 64$ bit precision. Examples of d and l are in the following table:

Dataset dimension d	Plaintext length l in $\mathbb{Z}_p^l$
10	4928
20	16128
30	33728

Since often $l \gg n$ in the ciphertext space $\mathbb{Z}_q^{n+l}$, our system's costs for communication and computation will mainly depend on l (and hence data dimension d) when global modulus q and precision prec are fixed.

Gaussian sampling. To generate discrete Gaussian noises, we employ the Knuth-Yao algorithm [14]. For reference, when $s = 8.0$, we can generate more than $4.1 \cdot 10^4$ Gaussian samples in one millisecond in one thread with 256-bit precision, using only 1.65 megabytes to store a binary tree.

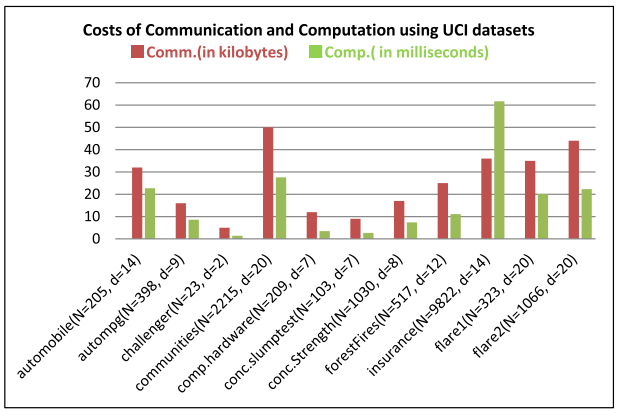

Fig. 3 The costs of our system using the UCI datasets [3].

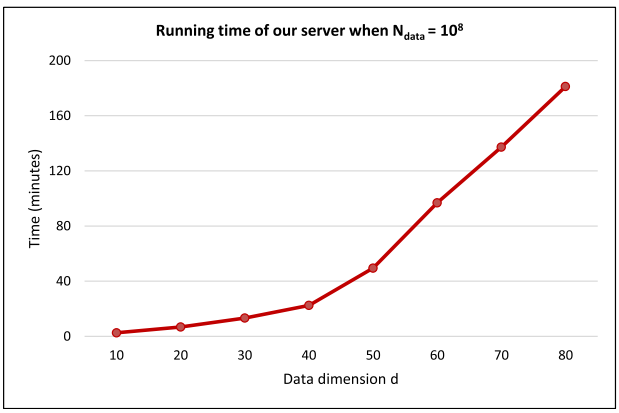

Fig. 4 Running time of our server (2.60GHz × 2 CPU, 128GB RAM) with data size $N_{\text{data}} = 10^8$ and varied dimension $10 \le d \le 80$. We use the parameters for 128-bit security in the experiment.

Experimental results. Experimental results are in Figs. 3 and 4. Specifically, on real UCI datasets [3] with $N_{\text{data}} < 10^4$ and $d \le 20$, our system finishes in milliseconds with less than 100 kilobytes of communication.

On simulated datasets with very large number of data records $N_{\text{data}} = 10^8$, we show that our system on a commodity server can finish in acceptable times depending on data dimension $10 \le d \le 80$.

Comparisons with previous works [13], [20]. The differences are in Table 2. Below give the details.

• (**Views on regression**) We view (linear, ridge) regression as the problem of a quadratic function optimization, while in [13], [20] as matrix inversion.

• (**Crypto model choices**) We use the model of [11], [23], while [20] combines homomorphic encryption with Yao's garbled circuits [25]. The use of garbled circuits (for matrix inversion) causes several rounds of interactions and megabytes of communication in [20].

• (**Server computation in [20]**) In turn, the server in [20] needs to do $O(N_{\text{data}})$ ciphertext additions and solves a garbled linear equation system with d equations and variables. The execution time of the garbled circuit is apparently not scalable well with d. Indeed, [20, Fig. 7] limits $d \leq 14$ giving the execution time in less than 140 seconds, and when $d = 20$, [20] reports hours of running time.

• (**Our server computation**) In contrast, in our system, the server does $O(N_{\text{data}})$ ciphertext additions; the client decrypts *once* to obtain all coefficients of $J_{\text{cost}}(\theta)$ at (7), and then minimize that convex quadratic function of $d+1$ variables, so that our approach scales better with d in both computation and communication. For example, even with dimension $d = 40$, the communication cost between the server and the client is only about 861 Kbytes computed via (9) using parameters for 80-bit security. Furthermore, if using the BFGS method to find the minimizer of $J_{\text{cost}}(\theta)$, the computational cost is $O(d^2)$ arithmetic operations with superlinear convergence rate [21]. Concretely, using the `fminunc` function (which implements the BFGS algorithm) in Octave 3.8.1 on a laptop, the minimizer θ^* is found in less than 1 minute when $d \leq 99$.

• (**Preconditions on the systems**) It is worth noting that, in [20], if the outsourced server and the garbled circuit provider (called crypto service provider or CSP in [20]) are colluded, then both can decrypt and learn all the data, so that no-collusion is assumed to ensure the security of their system. In ours, we follow [11], [23] assuming that the client is honest and can decrypt the encrypted data, and pay all security attention to the semi-honest outsourced server. These different preconditions on the systems, besides computation and communication costs, should be fully realized before any deployment in practice.

5. Conclusion

We build a privacy-preserving system for linear (ridge, LASSO) regressions which is efficiently scalable, while protecting both input (in terms of secrecy) and output (in terms of differential privacy). This is the first system of its kind for privacy-preserving linear regression.

References

[1] Amazon Machine Learning, https://aws.amazon.com/machine-learning/
[2] Google Cloud Machine Learning Platform, https://cloud.google.com/products/machine-learning/
[3] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml
[4] Y. Aono, X. Boyen, L.T. Phong, and L. Wang, "Key-private proxy re-encryption under LWE," INDOCRYPT, G. Paul and S. Vaudenay, eds., Lecture Notes in Computer Science, vol.8250, pp.1–18, Springer, 2013.
[5] Y. Aono, T. Hayashi, L.T. Phong, and L. Wang, "Privacy-preserving logistic regression with distributed data sources via homomorphic encryption," IEICE Trans. Inf. & Syst., vol.E99-D, no.8, pp.2079–2089, Aug. 2016.
[6] Y. Aono, T. Hayashi, L.T. Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," Proc. Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, E. Bertino, R. Sandhu, and A. Pretschner, eds., pp.142–144, ACM, 2016.
[7] J.W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," Journal of Biomedical Informatics, vol.50, pp.234–243, 2014.
[8] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," Advances in Cryptology, ASIACRYPT 2016, 22nd International Conference on the Theory and Application of Cryptology and Information Security, Part I, pp.3–33, 2016.
[9] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," Foundations and Trends in Theoretical Computer Science, vol.9, no.3-4, pp.211–407, 2014.
[10] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," Proc. 23rd USENIX Security Symposium, K. Fu and J. Jung, eds., pp.17–32, USENIX Association, 2014.
[11] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. thesis, Stanford University, 2009. https://crypto.stanford.edu/craig/
[12] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," Information Security and Cryptology, ICISC 2012, T. Kwon, M. Lee, and D. Kwon, eds., Lecture Notes in Computer Science, vol.7839, pp.1–21, Springer, 2012.
[13] R. Hall, S.E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," Journal of Official Statistics, vol.27, no.4, p.669, 2011.
[14] D.E. Knuth and A.C. Yao, "The complexity of non-uniform random number generation," Algorithms and Complexity, pp.357–428, Academic Press, New York, 1976.
[15] K. Lauter, A. López-Alt, and M. Naehrig, "Private computation on encrypted genomic data," Progress in Cryptology, LATINCRYPT 2014, D.F. Aranha and A. Menezes, eds., Lecture Notes in Computer Science, vol.8895, pp.3–27, Springer, 2014.
[16] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," CT-RSA, A. Kiayias, ed., Lecture Notes in Computer Science, vol.6558, pp.319–339, Springer, 2011.
[17] M. Liu and P.Q. Nguyen, "Solving BDD by enumeration: An update," CT-RSA, E. Dawson, ed., Lecture Notes in Computer Science, vol.7779, pp.293–309, Springer, 2013.
[18] P. Mell and T. Grance, "The NIST definition of cloud computing," http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[19] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," Proc. 3rd ACM Cloud Computing Security Workshop, CCSW 2011, C. Cachin and T. Ristenpart, eds., pp.113–124, Chicago, IL, USA, ACM, 2011.
[20] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," 2013 IEEE Symposium on Security and Privacy, SP 2013, pp.334–348, Berkeley, CA, USA, IEEE Computer Society, May 2013.
[21] J. Nocedal and S.J. Wright, Numerical Optimization, 2nd ed., Springer, New York, 2006.
[22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," STOC, H.N. Gabow and R. Fagin, eds., pp.84–93, ACM, 2005.

[23] R.L. Rivest, L. Adleman, and M.L. Dertouzos, "On data banks and privacy homomorphisms," Foundations of Secure Computation, vol.4, no.11, pp.169–180, 1978.
[24] The International Warfarin Pharmacogenetics Consortium, "Estimation of the warfarin dose with clinical and pharmacogenetic data," New England Journal of Medicine, vol.360, pp.753–764, 2009.
[25] A.C.-C. Yao, "How to generate and exchange secrets," Proc. 27th Annual Symposium on Foundations of Computer Science, SFCS '86, pp.162–167, Washington, DC, USA, IEEE Computer Society, 1986.
[26] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," PVLDB, vol.5, no.11, pp.1364–1375, 2012.

Yoshinori Aono received B.S. in Engineering from the Musashi Institute of Technology in 2005. Received M.S. and Ph.D. in Mathematical and Computing Sciences from the Tokyo Institute of Technology in 2007 and 2010, respectively. He is a researcher of the National Institute of Information and Communications Technology, Japan, conducting researches on security analysis of cryptography. He received Encouragement award in GPU Challenge 2009 from IPSJ, Young researcher's award 2011 from IEICE, and SCIS 2013 paper award from IEICE.

Takuya Hayashi received the Bachelor of Media Architecture and the Master of Systems Information Science degrees from Future University-Hakodate in 2008 and 2010, respectively, and the Doctor of Functional Mathematics degree from Kyushu University in 2013. He is currently an assistant professor of the Kobe university, and an invited advisor of the National Institute of Information and Communications Technology. His current research interests are in cryptanalysis and information security. He was awarded SCIS paper prize in 2010, and DOCOMO Mobile Science Award in 2013.

Le Trieu Phong received his B.S. from the University of Natural Sciences – Ho Chi Minh City, Viet Nam, in 2002, and his M.Sc. and Ph.D. from Tokyo Institute of Technology in 2006 and 2009 respectively. He is a senior researcher at the National Institute of Information and Communications Technology, Japan.

Lihua Wang received the B.S. degree in mathematics from Northeast Normal University, P.R. China, in 1988, the M.S. degree in mathematics from Harbin Institute of Technology, P.R. China, in 1994, and the Ph.D. degree in engineering from University of Tsukuba, Japan, in 2006, respectively. She is currently a senior researcher at the National Institute of Information and Communications Technology, Japan. Her research interests include cryptography and information security.