PAPER *Special Section on Security, Privacy and Anonymity of Internet of Things*

Privacy-Preserving Logistic Regression with Distributed Data Sources via Homomorphic Encryption*

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

SUMMARY Logistic regression is a powerful machine learning tool to classify data. When dealing with sensitive or private data, cares are necessary. In this paper, we propose a secure system for privacy-protecting both the training and predicting data in logistic regression via homomorphic encryption. Perhaps surprisingly, despite the non-polynomial tasks of training and predicting in logistic regression, we show that only additively homomorphic encryption is needed to build our system. Indeed, we instantiate our system with Paillier, LWE-based, and ring-LWE-based encryption schemes, highlighting the merits and demerits of each instantiation. Besides examining the costs of computation and communication, we carefully test our system over real datasets to demonstrate its utility.

key words: *logistic regression, distributed data sources, homomorphic encryption, Paillier, LWE, ring-LWE, outsourced computation, accuracy, F-score*

1. Introduction

1.1 Background

Logistic regression is a standard method in supervised machine learning to classify data. It is widely applied in various fields of science and engineering.

In several tasks using logistic regression, data contributors employ geographically distributed devices, raising the need of a central server to *receive*, *store*, *share*, and *process* the data as shown in Fig. 1.

However, the concentrated data collection at the central server may easily become a significant target for attacks. Even if the access to the server is properly controlled, the server itself cannot be completely trusted due to the sensitivity of the data.

The above tension of server utility versus data leakage on it has been long realized. Rivest, Adleman, and Dertouzos [25] pointed out that the tension can be relaxed via homomorphic encryption. This work follows the model of [25].

Manuscript received December 14, 2015.
Manuscript revised March 23, 2016.
Manuscript publicized May 31, 2016.

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

*A version of this paper appeared at [7].

a) E-mail: aono@nict.go.jp
b) E-mail: takuya.hayashi@nict.go.jp
c) E-mail: phong@nict.go.jp (Corresponding author)
d) E-mail: wlh@nict.go.jp

DOI: 10.1587/transinf.2015INP0020

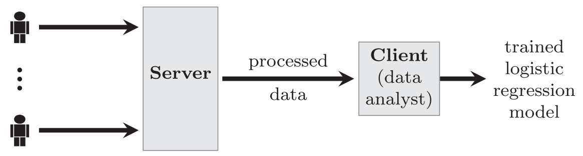


Fig. 1 Data flows.

1.2 Our Contributions

We build the first system for privacy-preserving logistic regression in which the storage and computation at the client side are relatively light. Specifically, our system is constructed via following technical steps and contributions:

(1) We turn the original logistic regression into what we call homomorphism-aware logistic regression via function approximations.
(2) We show how to use additively homomorphic encryption with the homomorphism-aware logistic regression.
(3) We show how to instantiate our system with Paillier, LWE-based, and ring-LWE-based encryption, and highlight their merits and demerits.
(4) We conduct experiments with real datasets from the UCI repository to report the utility of our system in terms of accuracy in prediction.

The effect of our design is that the system can securely handle data from distributed devices as in the Internet of Things. Specifically, our system enjoys the following properties

• (**Privacy-preserving**) Plain data is never leaked to the server, as homomorphic encryption is used.

• (**Outsourcing**) The storage and computation of each party is

$$\begin{cases} O(N_{\text{data}}d^2) & \text{at the server,} \\ O(d^2) & \text{at the client,} \end{cases}$$

where N_{data} is the number of records in a dataset and d is the dimension of each record. Therefore the heavy storage/computation goes to the server, the light work to the client as desired under the model of [25]. For concrete values, see Table 3.

1.3 Directly Related Works

The works [9], [22] consider private prediction in logistic

Table 1 Comparisons with previous works in the same model.

Papers	Data is protected in: training phase?	predicting phase?	Homomorphic operations over ciphertexts
[9], [22]	no	yes	addition, multiplication
Ours	yes	yes	addition (is enough)

regression, given that the regression coefficients (i.e. θ^* in Sect. 2.1) are publicly known. The papers [9], [22] do not consider the training phase of logistic regression. Therefore, our work complements [9] by protecting the training data in producing the regression coefficients. See Table 1. We also show that homomorphic additions over ciphertexts are enough in our construction.

The work [15] considers polynomial learning algorithms on encrypted data, assuming that multiplications over ciphertexts are supported by the underlying homomorphic encryption scheme. The paper [15] does not consider logistic regression; it cites [22] for logistic regression and examines other classifiers such as Linear Means and FLD.

1.4 Other Related Works

Approximate versions of logistic regression have appeared in other contexts such as large-scale text categorization [33] and differential privacy [34]. The work [34] does not consider data secrecy. Therefore, this work complements [34] by showing the secrecy of training data can be gained via encryption.

Bost et al. [10] examines several classifiers in the setting of two-party computation, in which the server has a secret θ (model) and the client has a secret x (data). Both interacts in a protocol with many rounds so that at the end the client learns $\mathsf{Circuit}(\theta, x)$ where $\mathsf{Circuit}$ is one of the classifiers. This two-party setting is different from ours of outsourced computation.

The papers [12], [13], [23] are also in the setting of secure two-party computation in which each party has a secret training set. Both wish to combine those sets in the training phase to obtain the trained model, but do not want to reveal their private dataset to each other. Therefore, the setting is different from ours of outsourced computation.

The paper [31] is also in the setting of two-party computation. It is worth noting that [31] uses an approximation for the sigmoid function, while our approximation is for the logarithm of the sigmoid function. Therefore, strictly speaking, the objects for approximation are also different.

Zhu et al. [35] also consider logistic regression via an interactive protocol in which their customer must be always active in communication in several rounds with the cloud server. The communication cost between the server and the customer in their protocol depends on both dataset size and dimension, and likewise the customer's computation cost. Therefore, [35] does not use the model of [25] and hence is different from ours.

Logistic regression with data privacy protection receives significant attentions from researchers in biomedical informatics [16], [29], [32], considering different settings of integrating and sharing data. These works do not use the model of [25] and hence are different from ours. Our work adds the setting of secure outsourced logistic regression to this line of research, showing the task can be done securely and efficiently.

Khedr et al. [17] examine classifiers using Bayesian filter and decision trees on encrypted data using homomorphic encryption supporting a few multiplications over ciphertexts. These classifiers are different from logistic regression.

The Internet of Things (IoT) can be seen as a cross-product of many fields, among which are distributed computing and big data. This field has attracted significant attention from researchers. Interested readers may refer to [28] (for definition of IoT), [30] (for distributed storage), [19] (for radio access network), [27] (for sensor data), to list just a few.

2. Logistic Regression and Its Approximation

2.1 Original Logistic Regression

Consider a (N_{data}, d)-dataset of N_{data} records and d dimension

$$\left\{x^{(i)}, y^{(i)}\right\}_{1 \le i \le N_{\text{data}}}$$

in which $x^{(i)} = \left(1, x_1^{(i)}, \ldots, x_d^{(i)}\right) \in \mathbb{R}^{d+1}$, $y^{(i)} \in \{0, 1\}$. Define the following cost function $J : \mathbb{R}^{d+1} \to \mathbb{R}$, with variable $\theta = (\theta_0, \theta_1, \ldots, \theta_d) \in \mathbb{R}^{d+1}$,

$$J(\theta) = \frac{\lambda}{2N_{\text{data}}} \sum_{j=1}^{d} \theta_j^2 + \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \Big[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \Big] \quad (1)$$

in which $0 < h_\theta(x) < 1$ is the value of the sigmoid function $h_\theta : \mathbb{R}^{d+1} \to \mathbb{R}$ given by

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)} = \frac{1}{1 + \exp(-\sum_{j=0}^{d} \theta_j x_j)}.$$

The training phase of logistic regression aims at finding the minimizer which optimizes the cost function

$$\theta^* = \text{argmin}_\theta J(\theta).$$

The predicting phase of logistic regression, given a new data $x^{\text{new}} = (1, x_1^{\text{new}}, \ldots, x_d^{\text{new}}) \in \mathbb{R}^{d+1}$, aims at guessing the binary value $y^{\text{new}} \in \{0, 1\}$ by setting

$$y^{\text{new}} = \begin{cases} 1 \text{ if } h_{\theta^*}(x^{\text{new}}) \ge \mathsf{thres} \\ 0 \text{ if } h_{\theta^*}(x^{\text{new}}) < \mathsf{thres} \end{cases} \quad (2)$$

in which $0 < \mathsf{thres} < 1$ is a variable threshold, and typically $\mathsf{thres} = 1/2$.

2.2 Homomorphism-Aware Logistic Regression via Approximation

The cost function $J(\theta)$ in Sect. 2.1 includes logarithm functions, causing obstacles when all the data are in encrypted form. In this section, via approximation of the logarithm functions, we derive what we call homomorphism-aware logistic regression.

Note that,

$$\log(h_\theta(x)) = \log\left(\frac{1}{1+\exp(-\theta^T x)}\right)$$
$$\log(1-h_\theta(x)) = \log\left(\frac{1}{1+\exp(\theta^T x)}\right)$$

so if we approximate the following function of $u \in \mathbb{R}$ by a degree k (e.g., $k=2$) polynomial

$$\log\left(\frac{1}{1+\exp(u)}\right) \approx \sum_{j=0}^{k} a_j u^j \tag{3}$$

we obtain following approximations

$$\log(1-h_\theta(x)) \approx \sum_{j=0}^{k} a_j(\theta^T x)^j \tag{4}$$

$$\log(h_\theta(x)) \approx \sum_{j=0}^{k} (-1)^j a_j(\theta^T x)^j \tag{5}$$

The approximate function of $J(\theta)$ is formed by using the polynomials at (4) and (5) to replace the corresponding logarithm functions in (1). Namely, define the following cost function

$$J_{\text{approx}}(\theta) = \frac{\lambda}{2N_{\text{data}}}\sum_{j=1}^{d}\theta_j^2 + J^*_{\text{approx}}(\theta) \tag{6}$$

in which

$$\begin{aligned} J^*_{\text{approx}}(\theta) &= \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}\Big[-y^{(i)}\sum_{j=0}^{k}(-1)^j a_j(\theta^T x^{(i)})^j \\ &\qquad - (1-y^{(i)})\sum_{j=0}^{k} a_j(\theta^T x^{(i)})^j\Big] \\ &= \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}\sum_{j=0}^{k}\left(y^{(i)} - y^{(i)}(-1)^j - 1\right) a_j(\theta^T x^{(i)})^j \\ &= \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}\sum_{j=1}^{k}\left(y^{(i)} - y^{(i)}(-1)^j - 1\right) a_j(\theta^T x^{(i)})^j - a_0. \end{aligned}$$

For our secure system, we further want to expand $(\theta^T x^{(i)})^j$ for all i. For that purpose, we have with $x = (x_0, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$,

$$\begin{aligned} (\theta^T x)^j &= \left(\sum_{r=0}^{d}\theta_r x_r\right)^j = \sum_{0\le r_1,\ldots,r_j\le d}(\theta_{r_1}x_{r_1})\cdots(\theta_{r_j}x_{r_j}) \\ &= \sum_{r_1,\ldots,r_j=0}^{d}(\theta_{r_1}\cdots\theta_{r_j})(x_{r_1}\cdots x_{r_j}) \end{aligned}$$

and denote

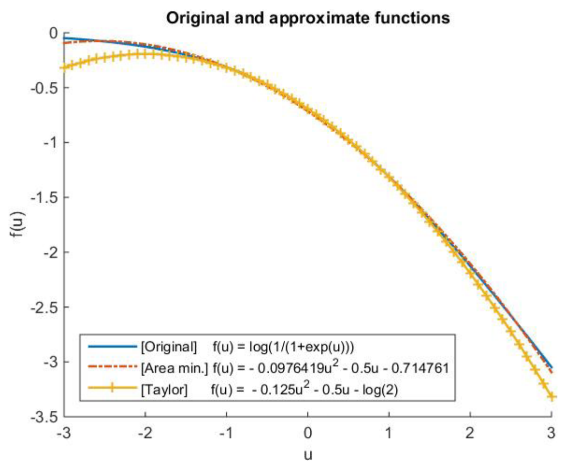


Fig. 2 Approximate functions.

$$A_{j,r_1,\ldots,r_j} = \sum_{i=1}^{N_{\text{data}}}\left(y^{(i)} - y^{(i)}(-1)^j - 1\right)\left(x^{(i)}_{r_1}\cdots x^{(i)}_{r_j}\right) \tag{7}$$

we can finally express

$$J^*_{\text{approx}}(\theta) = \frac{1}{N_{\text{data}}}\sum_{j=1}^{k}\sum_{r_1,\ldots,r_j=0}^{d} a_j(\theta_{r_1}\cdots\theta_{r_j})A_{j,r_1,\ldots,r_j} - a_0 \tag{8}$$

Since degree $k=2$ is mainly used in later sections, we give the explicit formulas of (7) as

$$A_{1,r_1} = \sum_{i=1}^{N_{\text{data}}}\underbrace{\left(2y^{(i)}-1\right)\left(x^{(i)}_{r_1}\right)}_{\text{owned by the } i^{\text{th}} \text{ data source}} \tag{9}$$

$$A_{2,r_1,r_2} = \sum_{i=1}^{N_{\text{data}}}\underbrace{(-1)\left(x^{(i)}_{r_1}x^{(i)}_{r_2}\right)}_{\text{owned by the } i^{\text{th}} \text{ data source}} \tag{10}$$

for later references. Furthermore, the coefficients a_j at (3) can be chosen via Taylor expansion, namely

$$a_0 = -\log 2, a_1 = -0.5, a_2 = -0.125.$$

Another method is via minimizing the area between the original and the quadratic approximation, yielding[†]

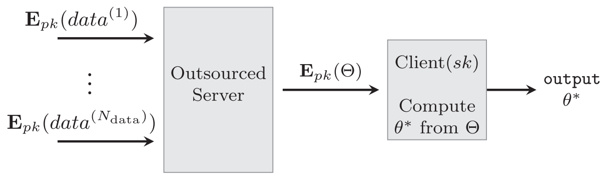
$$a_0 = -0.714761, a_1 = -0.5, a_2 = -0.0976419.$$

Figure 2 depicts the original and the (two) approximate functions, showing that the method of area minimization is apparently better. When d is relatively small such as $d \le 8$, both approximations work equally well in our experiments. When d is bigger, the approximation using the area tends to performs a little better so that we choose it in all later experiments.

3. Security Model (Rivest, Adleman, Dertouzos 1978)

Following [14], [25] we consider the following scenario: a

[†]found by using the function `fit` of `gnuplot`.



$\mathbf{E}_{pk}(data^{(1)})$
$\mathbf{E}_{pk}(data^{(N_{\text{data}})})$
Outsourced Server
$\mathbf{E}_{pk}(\Theta)$
Client(sk)
Compute θ^* from Θ
output θ^*

Fig. 3 Security model.

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. 3. We use $\mathbf{E}_{pk}(data^{(1)}), \dots, \mathbf{E}_{pk}(data^{(N_{\text{data}})})$ to represent the encryption of the data under a public key pk 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[†]; indeed, what requires is that θ^* can be efficiently derived from Θ.

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. 3 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 order 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. Our System for Privacy-Preserving Logistic Regression

This section describes our main system.

[†]For example, in [18], [22], $\Theta = (\theta_1^*, \theta_2^*) \in \mathbb{Z}^2$ and $\theta^* = \theta_1^*/\theta_2^* \in \mathbb{R}$, as current homomorphic encryption schemes do not support division directly.

Table 2 Costs in dataset size N_{data} and dimension d.

	Storage	**Computation**
Server	$O(N_{\text{data}}d^2)$	$O(N_{\text{data}}d^2)$
Client	$O(d^2)$	$O(d^2)$
Data sources	N/A	$O(d^2)$

	Communication
Each data source → server	$O(d^2)$
Server → client	$O(d^2)$

4.1 Securing Training Data via Encryption

We present our secure system for logistic regression under the model outlined in Fig. 3. Overall, in the following, the workload of the parties characterized by dataset size N_{data} and dimension d ($\ll N_{\text{data}}$ presumably) is in Table 2. The server carries the heaviest computation and storage. Remarkably, by our system design, the server computations can be easily parallelized, detailed below.

Encryption at data sources. Each data source i computes real numbers $a_{1,r_1}^{(i)} = \left(2y^{(i)} - 1\right) x_{r_1}^{(i)} \in \mathbb{R}$ and $a_{2,r_1,r_2}^{(i)} = (-1)\left(x_{r_1}^{(i)} x_{r_2}^{(i)}\right) \in \mathbb{R}$ for all $0 \le r_1, r_2 \le d$. Counting distinctly, these are $d+1$ and $(d+1)(d+2)/2$ numbers respectively, so that the data source needs to prepare totally

$$n_d = \frac{(d+1)(d+4)}{2} \tag{11}$$

real numbers using its d-dimensional data. In later sections, for the convenience of indexing, we denote these n_d numbers from data source i as $dat^{(i)} = (dat_1^{(i)}, \dots, dat_{n_d}^{(i)}) \in \mathbb{R}^{n_d}$.

Next, the data source encrypts the above $O(d^2)$ real numbers, using some additively homomorphic encryption scheme, to produce the ciphertext

$$CT^{(i)} = \mathbf{E}_{pk}\left(\{a_{1,r_1}^{(i)}\}_{0\le r_1\le d}, \{a_{2,r_1,r_2}^{(i)}\}_{0\le r_1,r_2\le d}\right)$$

or equivalently $CT^{(i)} = \mathbf{E}_{pk}(dat^{(i)})$, in which concrete instantiations of the encryption $\mathbf{E}_{pk}$ together with real number encodings will be described in later section.

Cloud server computation. The server receives and stores $CT^{(i)}$ for $1 \le i \le N_{\text{data}}$ from all data sources. It then computes the following ciphertext additions

$$CT = \sum_{i=1}^{N_{\text{data}}} CT^{(i)} \tag{12}$$

and sends CT to the client.

Client decryption. The client decrypts CT using its secret key. Due to the additive homomorphism of $\mathbf{E}_{pk}$, what the client obtains is $\sum_{i=1}^{N_{\text{data}}} dat^{(i)} \in \mathbb{R}^{n_d}$ or equivalently,

$$\sum_{i=1}^{N_{\text{data}}} a_{1,r_1}^{(i)} \in \mathbb{R}, \sum_{i=1}^{N_{\text{data}}} a_{2,r_1,r_2}^{(i)} \in \mathbb{R}$$

for all indexes $0 \le r_1, r_2 \le d$. These are exactly the coefficients given in (9) and (10) of the cost function at (8).

Using the coefficients, the client then finds the minimizer $\theta^* = \mathrm{argmin}_\theta J_{\text{approx}}(\theta)$.

Theorem 1: The above system is secure in the model of Sect. 3.

Proof 1: The proof is straightforward since the server only handles encrypted data, so the security is reduced to the semantic security of the underlying encryption scheme (which is either Paillier's or LWE-based in the following section).

4.2 Securing Data in Prediction

The output θ^* in Sect. 4.1 can be used for secure prediction in the sense that

- $\theta^* = (\theta^*_0, \ldots, \theta^*_d)$ can be made public, and
- data used in prediction $x = (x_1, \ldots, x_d)$ is encrypted,

and yet the output

$$h_{\theta^*}(x) = \frac{1}{1 + \exp(-\theta^*_0 - \sum_{j=1}^d \theta^*_j x_j)}$$

can be computed.

To compute $h_{\theta^*}(x)$, it suffices to know $\sum_{j=1}^d \theta^*_j x_j$. If the data for prediction is encrypted, it is necessary to compute

$$\sum_{j=1}^d \theta^*_j \mathbf{E}_{pk}(x_j)$$

which is discussed below:

- If $\mathbf{E}_{pk}(\cdot)$ supports only ciphertext addition, then it is necessary to multiply a greatest common divisor to convert the (fixed precision) scalars $\theta^*_j \in \mathbb{R}$ into integers. Then ciphertext additions are applied to obtain the encrypted result.
- If $\mathbf{E}_{pk}(\cdot)$ supports one multiplication such as in the (below) LWE and ring-LWE cases, it suffices to compute the sum of ciphertext product $\sum_{j=1}^d \mathbf{E}_{pk}(\theta^*_j)\mathbf{E}_{pk}(x_j)$.

In both cases, the data holder with the secret key sk decrypts to obtain $\sum_{j=1}^d \theta^*_j x_j$ and finally computes $h_{\theta^*}(x)$.

5. Instantiations of Our System

In this section we use additively homomorphic encryption schemes to instantiate our system in Sect. 4. We use following schemes, whose semantic securities are recalled in Appendix B, to show instantiations of our system:

- Paillier encryption,
- LWE-based encryption,
- ring-LWE-based encryption.

The Paillier scheme is not quantum-safe, so LWE-based and ring-LWE-based ones are the choices whenever quantum-resistant security is expected. Regarding the quantum-safe schemes, the ring-LWE-based one has smaller public key size while the LWE-based yields surprisingly smaller ciphertexts in communication and storage, showed below. These features enable flexible choices to employ our system in practice.

5.1 Using Paillier Encryption

Paillier encryption. With public key $pk = n$, the encryption of an integer $m \in \{0, \ldots, n-1\}$ is $\mathsf{PaiEnc}_{pk}(m) = r^n(1+n)^m \bmod n^2$ in which r is chosen randomly from $\{0, \ldots, n-1\}$. The encryption is additively homomorphic because the ciphertext product $\mathsf{PaiEnc}_{pk}(m_1)\mathsf{PaiEnc}_{pk}(m_2) \bmod n^2$ becomes an encryption of $m_1 + m_2 \bmod n$.

Packing data in encryption. As the plaintext space has $\log_2 n \geq 2048$ bits, we can pack many integers $dat_1, \ldots, dat_t$ each of prec bits into each Paillier plaintext as follows.

$$\mathsf{PaiEnc}_{pk}\Big(\overbrace{\underbrace{[dat_1 0_{\mathsf{pad}}]}_{\mathsf{prec+pad}\text{ bits}} \cdots \underbrace{[dat_t 0_{\mathsf{pad}}]}_{\mathsf{prec+pad}\text{ bits}}}^{\lfloor \log_2 n \rfloor \text{ bits}}\Big)$$

in which 0_{pad} is the zero padding of pad bits, which helps preventing overflows in ciphertext additions. Typically, $\mathsf{pad} \approx \log_2 N_{\text{data}}$ as we need N_{data} additions of ciphertexts. Moreover, as the number of plaintext bits must be less than $\log_2 n$, it is necessary that $t(\mathsf{prec} + \mathsf{pad}) \leq \log_2 n$. Therefore,

$$t = \left\lfloor \frac{\lfloor \log_2 n \rfloor}{\mathsf{prec} + \mathsf{pad}} \right\rfloor$$

which is the upper-bound of packing prec-bit integers into one Paillier plaintext. As a real number $0 \leq r < 1$ can be represented as an integer of form $\lfloor r \cdot 2^{\mathsf{prec}} \rfloor$, the above packing method can be used to encrypt around $\lfloor \log_2 n \rfloor / (\mathsf{prec} + \mathsf{pad})$ real numbers in the range $[0, 1)$ with precision prec, tolerating around 2^{pad} ciphertext additions.

Communication cost (data source to server). In our system in Sect. 4, each data source needs to encrypt and send $n_d = O(d^2)$ real numbers to the server. Therefore, with the above packing method, the number of Paillier ciphertexts sent from each data source is $\frac{n_d}{t} = \frac{n_d(\mathsf{prec+pad})}{\lfloor \log_2 n \rfloor}$ which is around

$$\mathsf{CommCostPaillier} = 2n_d(\mathsf{prec} + \mathsf{pad}) \text{ (bits)} \qquad (13)$$

where n_d is at (11), since each Paillier ciphertext is of $2\lfloor \log_2 n \rfloor$ bits.

Ciphertext additions (server). The ciphertext additions on the server can be seen as follows:

$$\begin{array}{c}\frac{n_d}{t}\text{ columns of }\mathsf{PaiEnc}_{pk}\\ \overbrace{\hphantom{\mathsf{PaiEnc}_{pk}\Big([dat_1^{(N_{\text{data}})}0_{\mathsf{pad}}]\cdots[dat_t^{(N_{\text{data}})}0_{\mathsf{pad}}]\Big),\ldots\quad}}\end{array}$$

$$\begin{array}{c}N_{\text{data}}\\ \text{rows}\end{array}\left\{\begin{array}{c}
\mathsf{PaiEnc}_{pk}\Big(\overbrace{\underbrace{[dat_1^{(1)}0_{\mathsf{pad}}]}_{\text{prec+pad bits}}\cdots\underbrace{[dat_t^{(1)}0_{\mathsf{pad}}]}_{\text{prec+pad bits}}}^{\lfloor\log_2 n\rfloor\text{ bits}}\Big)\qquad,\ldots\\
\vdots\\
+\\
\vdots\\
\mathsf{PaiEnc}_{pk}\Big(\overbrace{\underbrace{[dat_1^{(N_{\text{data}})}0_{\mathsf{pad}}]}_{\text{prec+pad bits}}\cdots\underbrace{[dat_t^{(N_{\text{data}})}0_{\mathsf{pad}}]}_{\text{prec+pad bits}}}^{\lfloor\log_2 n\rfloor\text{ bits}}\Big),\ldots
\end{array}\right.$$

whose computational cost is around

$$N_{\text{data}}\cdot\frac{n_d}{t}\cdot\mathbf{T}_{\mathsf{PaiAdd}}=N_{\text{data}}\cdot\frac{n_d(\mathsf{prec}+\mathsf{pad})}{\lfloor\log_2 n\rfloor}\cdot\mathbf{T}_{\mathsf{PaiAdd}}\quad(14)$$

where $\mathbf{T}_{\mathsf{PaiAdd}}$ is the time of adding two Paillier ciphertexts. The resulted n_d/t ciphertext sums in columns are sent to the client, which also requires $O(2d^2(\mathsf{prec}+\mathsf{pad}))$ bits.

Decryption (client). The client decrypts the n_d/t sums to obtain the sums $\sum_{i=1}^{N_{\text{data}}} dat_1^{(i)}\in\mathbb{Z},\ldots,\sum_{i=1}^{N_{\text{data}}} dat_t^{(i)}\in\mathbb{Z},\ldots,\sum_{i=1}^{N_{\text{data}}} dat_{n_d}^{(i)}\in\mathbb{Z}$. As $\mathsf{pad}=\lceil\log_2 N_{\text{data}}\rceil$, there will be no overflows in the integer sums as required.

5.2 Using a LWE-Based Encryption

In this section, $\mathbb{Z}_p$ are integers in $(-p/2,p/2]$ and likewise for $\mathbb{Z}_q$.

LWE-based encryption. We use the scheme in [6] due to its flexibility in choosing the plaintext length. Each plaintext is a vector in $\mathbb{Z}_p^l$ where p and l are almost independent with security parameters, so that it is possible to set $p\approx 2N_{\text{data}}$ and $l\approx O(d^2)$.

Concretely, for a plaintext $m\in\mathbb{Z}_p^{1\times l}$, the encryption is

$$\begin{aligned}\mathsf{lweEnc}_{pk}(m)&=e_1[A|P]+p[e_2|e_3]+[0_{n_{\text{lwe}}}|m]\qquad(15)\\ &\in\mathbb{Z}_q^{1\times(n_{\text{lwe}}+l)}\end{aligned}$$

in which $e_1\in\mathbb{Z}^{1\times n_{\text{lwe}}},e_2\in\mathbb{Z}^{1\times n_{\text{lwe}}},e_3\in\mathbb{Z}^{1\times l}$ are Gaussian noise vectors of deviation s; $[A|P]$ is the matrix concatenation of public matrices $A\in\mathbb{Z}_q^{n_{\text{lwe}}\times n_{\text{lwe}}}$ and $P=pR-AS\in\mathbb{Z}_q^{n_{\text{lwe}}\times l}$ given in the public key $pk=(A,P,(p,l),(n_{\text{lwe}},s,q))$ for noises $R,S\in\mathbb{Z}^{n_{\text{lwe}}\times l}$.

The encryption is additively homomorphic because

$$\begin{aligned}&\mathsf{lweEnc}_{pk}(m)+\mathsf{lweEnc}_{pk}(m')\\ &=e_1[A|P]+p[e_2|e_3]+[0_{n_{\text{lwe}}}|m]\\ &\quad+e_1'[A|P]+p[e_2'|e_3']+[0_{n_{\text{lwe}}}|m']\\ &=(e_1+e_1')[A|P]+p[e_2+e_2'|e_3+e_3']+[0_{n_{\text{lwe}}}|m+m']\end{aligned}$$

in $\mathbb{Z}_q^{1\times(n_{\text{lwe}}+l)}$, which is the encryption of $m+m'\in\mathbb{Z}_p^{1\times l}$.

Data encoding and encryption. Real numbers $a\in\mathbb{R}$ can be represented in $\mathsf{prec}=(L+\ell+1)$ signed bits as

$$a=\sum_{k=-L}^{\ell}a_k2^k$$

which is the inner product of following vectors

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

in which all signed bits are in $\{0,1\}$ if $a\geq 0$ and in $\{-1,0\}$ if $a<0$.

Encryption from each data source corresponds to each row in (17). Namely, each data source i having n_d real numbers takes all signed bits of these numbers, concatenating them to form a vector in $\{-1,0,1\}^{n_d\cdot\mathsf{prec}}$ and encrypts that vector using lweEnc. Note that $\{-1,0,1\}\subset\mathbb{Z}_p$ it is sufficient to set the plaintext length $l=n_d\cdot\mathsf{prec}$.

Communication cost (data source to server). Each data source sends to the server a row in (17), whose size is

$$(n_{\text{lwe}}+n_d\cdot\mathsf{prec})\log_2 q\ (\text{bits})\qquad(16)$$

which is $O(d^2\mathsf{prec})$ as n_{lwe} and q are fixed as parameters of the scheme.

Ciphertext additions (server). The N_{data} ciphertext additions on the server can be seen the row additions as follows, where each row comes from each data source.

$$\begin{array}{c}N_{\text{data}}\\ \text{rows}\end{array}\left\{\overbrace{\begin{array}{c}
\mathsf{lweEnc}_{pk}\Big(\underbrace{\mathsf{Bits}(dat_1^{(1)})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\cdots\underbrace{\mathsf{Bits}(dat_{n_d}^{(1)})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\Big)\\
\vdots\\
+\\
\vdots\\
\mathsf{lweEnc}_{pk}\Big(\underbrace{\mathsf{Bits}(dat_1^{(N_{\text{data}})})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\cdots\underbrace{\mathsf{Bits}(dat_{n_d}^{(N_{\text{data}})})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\Big)
\end{array}}^{1\text{ column of }\mathsf{lweEnc}_{pk}\in\mathbb{Z}_q^{1\times(n_{\text{lwe}}+n_d\cdot\mathsf{prec})}}\right.\qquad(17)$$

whose computational cost is around

$$N_{\text{data}}\cdot\mathbf{T}_{\mathsf{lweAdd}}=N_{\text{data}}\cdot(n_{\text{lwe}}+n_d\cdot\mathsf{prec})\cdot\mathbf{t}_{\text{add}\mathbb{Z}_q}\qquad(18)$$

where $\mathbf{T}_{\mathsf{lweAdd}}$ is the adding time of two vectors in the set $\mathbb{Z}_q^{n_{\text{lwe}}+n_d\cdot\mathsf{prec}}$ and $\mathbf{t}_{\text{add}\mathbb{Z}_q}$ is the time for adding two elements in $\mathbb{Z}_q$. The resulted sum is sent to the client and its length is the same as in (16).

Decryption (client). The client decrypts the sum to obtain following sums of vectors (over $\mathbb{Z}$)

$$\begin{array}{c}\mathsf{Bits}(dat_1^{(1)})+\cdots+\mathsf{Bits}(dat_1^{(N_{\text{data}})})\in[-N_{\text{data}},N_{\text{data}}]^{\mathsf{prec}}\\ \vdots\\ \mathsf{Bits}(dat_{n_d}^{(1)})+\cdots+\mathsf{Bits}(dat_{n_d}^{(N_{\text{data}})})\in[-N_{\text{data}},N_{\text{data}}]^{\mathsf{prec}}\end{array}$$

and then takes the inner products with $\mathsf{Pow}(2,L,\ell)$ to get following $n_d=O(d^2)$ real numbers

$$dat_1^{(1)}+\cdots+dat_1^{(N_{\text{data}})}\in\mathbb{R}$$

$$\vdots$$
$$dat_{n_d}^{(1)} + \cdots + dat_{n_d}^{(N_{\text{data}})} \in \mathbb{R}$$

as required. The condition for not overflowing here is that $[-N_{\text{data}}, N_{\text{data}}] \subseteq \mathbb{Z}_p$, meaning the plaintext of the resulted ciphertext after additions must be in $\mathbb{Z}_p^{n_d \cdot \mathsf{prec}}$. Therefore $N_{\text{data}} < p/2$ is necessary.

5.3 Using a Ring-LWE-Based Encryption

Ring-LWE-based encryption. We use the ring-LWE-based scheme described in [15]. Define ring $R = \mathbb{Z}[x]/f(x)$, $f(x) = x^{n_{\text{rlwe}}}+1$, and quotient rings $R_q = R/q, R_p = R/p$. The notion $R_{(0,s)}$ stands for polynomials in R with small Gaussian coefficients of mean 0 and deviation s.

The encryption of $\mathbf{m} \in R_p$ under public key $pk = (\mathbf{a}, \mathbf{p} = -\mathbf{as} - \mathbf{e}) \in R_q^2$ is as follows

$$\mathsf{rlweEnc}_{pk}(\mathbf{m}) = (\mathbf{e}_1\mathbf{a} + \mathbf{e}_2, \mathbf{e}_1\mathbf{p} + \mathbf{e}_3 + \lfloor q/p \rfloor \mathbf{m}) \in R_q^2 \qquad (19)$$

in which $\mathbf{e}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \in R_{(0,s)}$ are noises and secret key $\mathbf{s} \in R_q$.

The addition of ciphertexts can be done naturally as

$$\mathsf{rlweEnc}_{pk}(\mathbf{m}) + \mathsf{rlweEnc}_{pk}(\mathbf{m}') \in R_q^2$$

in which $\mathbf{m}, \mathbf{m}' \in R_p$ can also be seen as vectors of length n_{rlwe} over $\mathbb{Z}_p = (-p/2, p/2] \cap \mathbb{Z}$.

Data encoding and encryption. Data encoding is identical to Sect. 5.2. Encryption of the encoding is different since the plaintext length is fixed with n_{rlwe}. Namely each data source needs to pack their data items into the ring-LWE encryption. Referring to the rows of (21), let t be the number of data items packed into one $\mathsf{rlweEnc}$, we have $t \cdot \mathsf{prec} = n_{\text{rlwe}}$ so that $t = \left\lfloor \frac{n_{\text{rlwe}}}{\mathsf{prec}} \right\rfloor$.

Communication cost (data source to server). Each data source sends to the server a row in (21), whose size is

$$\frac{n_d}{t}(2n_{\text{rlwe}} \log_2 q) \approx 2n_d \cdot \mathsf{prec} \cdot \log_2 q \text{ (bits)} \qquad (20)$$

since each ring-LWE ciphertext is of $2n_{\text{rlwe}} \log_2 q$ (bits). It is worth noting that the cost at (20) in ring-LWE case is bigger than (16) in LWE case if $n_{\text{lwe}} < n_d \cdot \mathsf{prec}$.

Ciphertext additions (server). The N_{data} ciphertext additions on the server can be seen the row additions as follows, where each row comes from each data source.

$$N_{\text{data}} \text{ rows} \left\{ \begin{array}{c} \overbrace{\mathsf{rlweEnc}_{pk}\Big(\underbrace{\mathsf{Bits}(dat_1^{(1)})}_{\in\{-1,0,1\}^{\mathsf{prec}}} \cdots \underbrace{\mathsf{Bits}(dat_t^{(1)})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\Big), \ldots}^{\frac{n_d}{t} \text{ columns of } \mathsf{rlweEnc}_{pk} \in R_q^2} \\ \vdots \\ + \\ \vdots \\ \mathsf{rlweEnc}_{pk}\Big(\underbrace{\mathsf{Bits}(dat_1^{(N_{\text{data}})})}_{\in\{-1,0,1\}^{\mathsf{prec}}} \cdots \underbrace{\mathsf{Bits}(dat_t^{(N_{\text{data}})})}_{\in\{-1,0,1\}^{\mathsf{prec}}}\Big), \ldots \end{array} \right. \qquad (21)$$

whose computational cost is around

$$N_{\text{data}} \cdot \frac{n_d}{t} \cdot \mathbf{T}_{\mathsf{rlweAdd}} \approx N_{\text{data}} \cdot \frac{n_d \cdot \mathsf{prec}}{n_{\text{rlwe}}} \cdot \mathbf{T}_{\mathsf{rlweAdd}} \qquad (22)$$

in which $\mathbf{T}_{\mathsf{rlweAdd}}$ is the time of adding two $\mathsf{rlweEnc}$ ciphertexts (seen as polynomials) in R_q^2.

Decryption (client). This is the same as Sect. 5.2 and the condition $N_{\text{data}} < p/2$ is again necessary to prevent overflows in adding the columns.

5.4 Concrete Estimation of Costs

To have a concrete estimation of costs, fix (number of records, dimension) in the training dataset as

$$(N_{\text{data}}, d) = (10^6, 20)$$

which encompasses many real datasets as in [1] and the later used datasets in Sect. 6.

The estimation result is in Table 3 whose production is detailed below. As $d = 20$, via (11) $n_d = 252$ being used in following cost formula for communication and computation.

Using Paillier's scheme. We take the Paillier modulus n of 3072 bits, which is standard to gain 128-bit security, so that $\log_2 n \approx 3072$ which is also the public key size in bits. Each ciphertext size is computed using (13) with precision $\mathsf{prec} = 64$ and padding number $\mathsf{pad} = \lceil \log_2(N_{\text{data}}) \rceil$. The computation cost is computed via (14).

Using the LWE-based scheme. Each ciphertext size is computed using (16) in which $n_{\text{lwe}} = 2767$, $\mathsf{prec} = 64$, $\log_2 q = 90$. The parameters of (n_{lwe}, q) and Gaussian deviation $s = 8.0$ are set to gain around 128-bit security with respect to current attacks [5], [20], [21]. To handle such N_{data}, we take $p = 2097153$.

The public key consisting $A \in \mathbb{Z}_q^{n_{\text{lwe}} \times n_{\text{lwe}}}$ and $P \in \mathbb{Z}_q^{n_{\text{lwe}} \times n_d \mathsf{prec}}$ of $n_{\text{lwe}}(n_{\text{lwe}} + n_d \mathsf{prec}) \log_2 q$ bits, which becomes around 588.18 Mbyte.

The communication and computation costs are gained via (16) and (18) with above parameters.

Using the Ring-LWE-based scheme. We use the same parameters as in the LWE case for 128-bit security. In particular, $n_{\text{rlwe}} \approx 2767$.

The public key size is $2n_{\text{rlwe}} \log_2 q$ bits, which becomes 62.2575 Kbyte.

The communication cost is computed via (20). The computation cost is computed via (22). It is also worth noting that the computation cost of using the ring-LWE-based scheme can be *larger* than that of the LWE-based one, argued as follows. Looking (18) and (22), it suffices to show that

$$N_{\text{data}} \cdot \mathbf{T}_{\mathsf{lweAdd}} \le N_{\text{data}} \cdot \frac{n_d \cdot \mathsf{prec}}{n_{\text{rlwe}}} \cdot \mathbf{T}_{\mathsf{rlweAdd}}$$

or equivalently

Table 3 Storage and computation when (#records, #dim.) = $(N_{\text{data}}, d) = (10^6, 20)$.

Our system	Server storage of ciphertexts	Server computation	Client received from server	Client computation
using Paillier	5.2877 Gbyte	$6.885 \cdot 10^6 \cdot \mathbf{T}_{\mathsf{PaiAdd}}$	5.2877 Kbyte	$6.885 \cdot \mathbf{T}_{\mathsf{PaiDec}}$
using LWE	212.56 Gbyte	$1.889 \cdot 10^{10} \cdot \mathbf{t}_{\text{add}\mathbb{Z}_q}$	212.56 Kbyte	$1 \cdot \mathbf{T}_{\mathsf{lweDec}}$
using ring-LWE	362.88 Gbyte	$3.225 \cdot 10^{10} \cdot \mathbf{t}_{\text{add}\mathbb{Z}_q}$	362.88 Kbyte	$5.8287 \cdot \mathbf{T}_{\mathsf{rlweDec}}$

$\mathbf{T}_{\mathsf{PaiAdd}}$: time for adding two Paillier ciphertexts each under 3072-bit public key; $\mathbf{t}_{\text{add}\mathbb{Z}_q}$: time for adding two integers in $\mathbb{Z}_q$; $\mathbf{T}_{\mathsf{PaiDec}}$: time for one Paillier decryption; $\mathbf{T}_{\mathsf{lweDec}}$: time for one decryption in the LWE-based scheme; $\mathbf{T}_{\mathsf{rlweDec}}$: time for one decryption in the ring-LWE-based scheme. On our server (2.60GHz x 2 CPU, 128GB RAM), $\mathbf{T}_{\mathsf{PaiAdd}} \approx 10 \cdot 10^{-6}$ seconds, $\mathbf{t}_{\text{add}\mathbb{Z}_q} \approx 1.146 \cdot 10^{-9}$ seconds. On a laptop, decryption times $\mathbf{T}_{\mathsf{PaiDec}}, \mathbf{T}_{\mathsf{lweDec}}, \mathbf{T}_{\mathsf{rlweDec}}$ are in milliseconds, so that the client computation is negligible.

Table 4 Comparing actual/predicted class.

		Actual class $y^{(i)}$	
		1	0
predicted	1	True Positive (TP)	False Positive (FP)
class $y^{(i)}$	0	False Negative (FN)	True Negative (TN)

$$\mathbf{T}_{\mathsf{lweAdd}} \le \frac{n_d \cdot \mathsf{prec}}{n_{\text{rlwe}}} \cdot \mathbf{T}_{\mathsf{rlweAdd}}.$$

Let $\mathbf{t}_{\text{add}\mathbb{Z}_q}$ be the time of adding two elements in $\mathbb{Z}_q$, then we can estimate $\mathbf{T}_{\mathsf{lweAdd}} = (n_{\text{lwe}} + n_d\mathsf{prec})\mathbf{t}_{\text{add}\mathbb{Z}_q}$ and $\mathbf{T}_{\mathsf{rlweAdd}} = 2n_{\text{rlwe}}\mathbf{t}_{\text{add}\mathbb{Z}_q}$. The reason is that $\mathbf{T}_{\mathsf{lweAdd}}$ is the time for adding two elements in $\mathbb{Z}_q^{n_{\text{lwe}}+n_d\mathsf{prec}}$ and $\mathbf{T}_{\mathsf{rlweAdd}}$ is for adding two polynomials of degree n_{rlwe} in $\mathbb{Z}_q$. Therefore, it is sufficient to show

$$(n_{\text{lwe}} + n_d\mathsf{prec})\mathbf{t}_{\text{add}\mathbb{Z}_q} \le \frac{n_d \cdot \mathsf{prec}}{n_{\text{rlwe}}} \cdot 2n_{\text{rlwe}}\mathbf{t}_{\text{add}\mathbb{Z}_q}$$

which holds true as long as $n_{\text{lwe}} \le n_d\mathsf{prec} = (d+1)(d+4)\mathsf{prec}/2$. Table 3 reflects this fact as $n_{\text{lwe}} = 2767$, $d = 20$, $\mathsf{prec} = 64$.

6. Experiments with Real Datasets

This section aims at showing the utility of our secure system regarding the accuracy in prediction. We first remind the confusion matrix in Table 4. We then conducts experiments with a few UCI datasets [1].

Accuracy, Precision, and Recall. Referring to Table 4, define

$$\textbf{Accuracy} = \frac{\#\mathsf{TP} + \#\mathsf{TN}}{\#\mathsf{TP} + \#\mathsf{FP} + \#\mathsf{FN} + \#\mathsf{TN}} \tag{23}$$

$$(\textbf{Precision})\ P = \frac{\#\mathsf{TP}}{\#\mathsf{TP} + \#\mathsf{FP}} \tag{24}$$

$$(\textbf{Recall})\ R = \frac{\#\mathsf{TP}}{\#\mathsf{TP} + \#\mathsf{FN}} \tag{25}$$

in which #TP reads as the number of true positives (TP) and likewise for the others.

Trivial accuracy. This is the percentage of 0 (or 1 if higher) in the actual class of $y^{(i)}$ in the testing set. A trivial "classifier"does nothing than returning 0 (or 1 respectively) will get the trivial accuracy. The accuracy at (23) must be higher than this trivial accuracy.

F-score. The score (aka, F_1-score) is defined as

$$F_{\text{score}} = \frac{2PR}{P+R}$$

where P and R are at (24) and (25). We always have $0 \le F_{\text{score}} \le 1$, and the higher F_{score} is, the better the classifier.

6.1 Using the Pima Diabetes Dataset

The Pima Indians diabetes dataset [1], [26] consists of 768 records each of dimension 8. Following [26], we split the dataset into two parts: a training set of 576 records and a testing set of 192 records. The model gaining by the training set is used in the testing set to compute the accuracy. The datasets in this and following sections are all normalized by their means and deviations.

In our system over encrypted data, after obtaining the coefficients of the approximate cost function, the client runs the gradient descent algorithm with initial model θ_{init} (Table A· 1 in Appendix A) and learning rate 0.1 with 200 steps of iterations. Also we choose $\lambda = 1$ in the cost function to penalize big coefficients in the model. Finally, the client obtains the minimizer θ^* (Table A· 1, Appendix A).

As seen in Table 5, the accuracy of our system is a little better than the known accuracy reported in [4], [26] (over unencrypted data).

6.2 Using the Wisconsin Breast Cancer Dataset

The original Wisconsin breast cancer dataset [1] has 683 records (already excluded those with missing values) each with dimension 9. We split the dataset into two parts: a training set with 400 records and a testing set with 283 records. The benign (resp., malignant) ratios are 57% (resp., 43%) in the training set, and 76.3% (resp., 23.7%) in the testing set. Therefore, a good model for prediction must have an accuracy far higher than the trivial accuracy 76.3%. As reported in Table 5, we reach a better accuracy than the ones known in [3].

6.3 Using the Banknote Authentication Dataset

In this dataset of [1], data is extracted from images of both genuine and forged banknotes. There are 1372 records each of dimension 4. We set $(N_{\text{data}}, d) = (986, 4)$ for the training set, leaving 386 records for the test set. In the test set, the percentages of label 0 and 1 are both 50%, so the trivial accuracy is 50%. Our system reaches a high accuracy of 98.4% over the test set. Previous accuracies are not applied as we take our own partition of the original dataset. Details of parameters, input and output are given in Table A· 1.

Table 5 Experiments with real datasets to compare classifying accuracies.

UCI Dataset Names [1]	**(#records, #dim.)** (N_{data}, d)	**Trivial accuracy**	**Previous accuracy** (data is unencrypted)	**Our system accuracy** (data is encrypted)	**Our system** F_{score}
Pima (diabetes)	$(576, 8)$	63.54%	76% (see [26])	80.7%	0.688525
Breast Cancer	$(400, 9)$	76.3%	95.9% (see [3])	98.2%	0.962406
Banknote Authentication	$(986, 4)$	50%	N/A	98.4%	0.984615
Adult Income	$(30162, 14)$	75.4%	79.65% ~ 85.9% (see [2])	81.97%	0.526921
Skin/NonSkin	$(122528, 3)$	79.24%	N/A	93.89%	0.960130

Table 6 Data dimension vs. accuracies.

UCI Dataset names	Dimension (sorted)	Our system accuracy	Original logistic regression accuracy
Skin/NonSkin	3	93.89%	93.88%
Banknote Authen.	4	98.4%	98.4%
Pima (diabetes)	8	80.7%	80.2%
Breast Cancer	9	98.2%	98.2%
Adult Income	14	81.97%	83.63%

6.4 Using the Adult Income Dataset

In the training set of this dataset of [1], $(N_{\text{data}}, d) = (30162, 14)$ in which 75.1% has label 0 (having income $\leq$ 50K) and 24.9% has label 1 (having income $>$ 50K). The test set has 15060 records of which 75.4% has label 0 and 24.6% has label 1.

After gaining the trained model using encrypted training data, we reach the accuracy of 81.97% on the test set. This is not bad in comparison with known accuracies produced by various classifiers given in [2] over plain data: 79.65% (nearest-neighbor algorithm), 83.88% (Naive-Bayes algorithm), 85.90% (NBTree algorithm).

6.5 Using the Skin/NonSkin Dataset

This UCI dataset [8] contains skin and non-skin face image samples. It has 245057 records, each of dimension 3 (for R, G, B colors). We split the original dataset into two parts: the training set of $(N_{\text{data}}, d) = (122528, 3)$ and the rest is for the testing set. The trivial accuracy over the testing set is 79.24% and our system obtains a higher accuracy of 93.89% as reported in Table 5. Previous accuracies are not applied as we take our own partition of the original dataset. Parameters in the experiment is given in Table A· 1.

6.6 Discussions on the Experimental Results: Dataset Dimension vs. Accuracy

In Table 6, we compare the accuracies of our system and original logistic regression, when using the same parameters as in Table A· 1. From Tables 5 and 6, it is worth noting the pros and cons of our system as follows.

- (Pros) Our system accuracy tends to be not affected by the number of records N_{data}.
- (Cons) Our system (over encrypted data) tends to be less accurate than original logistic regression (over plain data) when the data dimension d goes up.

The illustration for the above points can be as follows. Our approximation in Fig. 2 depends on the value $u = \theta^T x = \sum_{i=0}^{d} \theta_i x_i$. As θ_i, x_i can be small via the standard normalization over the dataset, u depends very much on the data dimension d. When d is small (as in Skin/NonSkin, Banknote, Pima, Breast Cancer), our approximation works very well. However, when d becomes larger (as in Adult Income), our approximation in Fig. 2 becomes less accurate, yielding smaller accuracy than that of the original logistic regression.

Tables 5 and 6 also confirm the fact that logistic regression is very powerful, especially with medical and biomedical data. While the illustration of this fact is out of the security and privacy scope of our paper, interested readers can find such an illustration in, e.g., [11].

7. Conclusion

We show, for the first time, that privacy-preserving logistic regression is efficiently possible with distributed data sources. This is a step towards helping to protect sensitive data while accelerating research which requires integrated data collected via the Internet of Things.

References

[1] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml

[2] Adult dataset. https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names

[3] Breast Cancer. https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/

[4] Pima dataset. https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/

[5] Y. Aono, X. Boyen, L.T. Phong, and L. Wang, "Key-private proxy re-encryption under LWE," G. Paul and S. Vaudenay, editors, Progress in Cryptology – INDOCRYPT 2013, vol.8250 of Lecture Notes in Computer Science, pp.1–18. Springer, 2013.

[6] Y. Aono, T. Hayashi, L.T. Phong, and L. Wang, "Fast and secure linear regression and biometric authentication with security update," Cryptology ePrint Archive, Report 2015/692, 2015. http://eprint.iacr.org/

[7] Y. Aono, T. Hayashi, L.T. Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, pp.142–144, 2016.

[8] R. Bhatt and A. Dhall, "Skin Segmentation Dataset," https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

[9] 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.

[10] R. Bost, R.A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," 22nd Annual Network and Dis-

tributed System Security Symposium, NDSS 2015. The Internet Society, 2015.

[11] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," J. Biomed. Inform., vol.35, no.5-6, pp.352–359, 2002.

[12] W. Du, S. Chen, and Y.S. Han, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," Proceedings of the 4th SIAM International Conference on Data Mining, pp.222–233, 2004.

[13] D.A. duVerle, S. Kawasaki, Y. Yamada, J. Sakuma, and K. Tsuda, "Privacy-preserving statistical analysis by exact logistic regression," 2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, May 21-22, 2015, pp.7–16, IEEE Computer Society, 2015.

[14] C. Gentry, A fully homomorphic encryption scheme, PhD Thesis, Stanford University, 2009. crypto.stanford.edu/craig

[15] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," T. Kwon, M. Lee, and D. Kwon, editors, Information Security and Cryptology – ICISC 2012, vol.7839 of Lecture Notes in Computer Science, pp.1–21, Springer, 2012.

[16] W. Jiang, P. Li, S. Wang, Y. Wu, M. Xue, L. Ohno-Machado, and X. Jiang, "WebGLORE: a web service for grid logistic regression," Bioinformatics, vol.29, no.24, pp.3238–3240, 2013.

[17] A. Khedr, G. Gulak, and V. Vaikuntanathan, "SHIELD: scalable homomorphic implementation of encrypted data-classifiers," IEEE Transactions on Computers, p.1, 2015.

[18] K. Lauter, A. López-Alt, and M. Naehrig, "Private computation on encrypted genomic data," D.F. Aranha and A. Menezes, editors, Progress in Cryptology – LATINCRYPT 2014, vol.8895 of Lecture Notes in Computer Science, pp.3–27, Springer, 2014.

[19] H. Li, M. Dong, and K. Ota, "Radio access network virtualization for the social Internet of Things," IEEE Cloud Computing, vol.2, no.6, pp.42–50, 2015.

[20] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," A. Kiayias, editor, Topics in Cryptology – CT-RSA 2011, vol.6558 of Lecture Notes in Computer Science, pp.319–339, Springer, 2011.

[21] M. Liu and P.Q. Nguyen, "Solving BDD by enumeration: An update," E. Dawson, editor, Topics in Cryptology – CT-RSA 2013, vol.7779 of Lecture Notes in Computer Science, pp.293–309, Springer, 2013.

[22] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," C. Cachin and T. Ristenpart, editors, Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, Oct. 21, 2011, pp.113–124, ACM, 2011.

[23] Y. Nardi, S.E. Fienberg, and R.J. Hall, "Achieving both valid and secure logistic regression analysis on aggregated data from different private sources," Journal of Privacy and Confidentiality, vol.4, no.1, pp.189–220, 2012.

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," J. Stern, ed., Advances in Cryptology – EUROCRYPT '99, vol.1592 of Lecture Notes in Computer Science, pp.223–238, Springer, 1999.

[25] 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.

[26] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," Proceedings of the Annual Symposium on Computer Application in Medical Care, pp.261–265, American Medical Informatics Association, 1988.

[27] S.K. Sowe, T. Kimata, M. Dong, and K. Zettsu, "Managing heterogeneous sensor data on a big data platform: IoT services for data-intensive science," 2014 IEEE 38th International COMPSAC Workshops, pp.295–300, 2014.

[28] J. Voas, "Why is IoT still definition-less?," ACM CODASPY 2016, Keynote talk, http://www.codaspy.org/keynotes

[29] S. Wang, X. Jiang, Y. Wu, L. Cui, S. Cheng, and L. Ohno-Machado, "Expectation propagation logistic regression (EXPLORER): distributed privacy-preserving online model learning," Journal of Biomedical Informatics, vol.46, no.3, pp.480–496, 2013.

[30] J. Wu, M. Dong, K. Ota, L. Liang, and Z. Zhou, "Securing distributed storage for social Internet of Things using regenerating code and Blom key agreement," Peer-to-Peer Networking and Applications, vol.8, no.6, pp.1133–1142, 2015.

[31] S. Wu, T. Teruya, J. Kawamoto, J. Sakuma, and H. Kikuchi, "Privacy-preservation for stochastic gradient descent application to secure logistic regression," 27th Annual Conference of the Japanese Society for Artificial Intelligence, 3L1-OS-06a-3, 2013.

[32] Y. Wu, X. Jiang, S. Wang, W. Jiang, P. Li, and L. Ohno-Machado, "Grid multi-category response logistic models," BMC Med. Inf. & Decision Making, vol.15, p.10, 2015.

[33] J. Zhang, R. Jin, Y. Yang, and A.G. Hauptmann, "Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization," Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), Aug. 2003, Washington, DC, USA, pp.888–895, 2003.

[34] 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.

[35] X.D. Zhu, H. Li, and F.H. Li, "Privacy-preserving logistic regression outsourcing in cloud computing," IJGUC, vol.4, no.2/3, pp.144–150, 2013.

Appendix A: Initial and Trained Models in Our System Using Real UCI Datasets

The following tables contain the initial and trained parameters for Sect. 6. The initial θ_{init} is chosen in a random way. The minimizer $\theta^* = \text{argmin}_\theta J_{\text{approx}}(\theta)$ achieves our system

Table A·1 Gradient descent with the UCI datasets.

Pima (diabetes)		
$\lambda = 1$, learning rate = 0.1, steps of iterations = 200		
θ_{init}	=	[0.334781, −0.633628, 0.225721, −0.648192, 0.406207, 0.044424, −0.426648, 0.877499, −0.426819]
θ^*	=	[−0.618931, 0.272079, 0.687556, −0.164313, 0.023873, −0.078103, 0.426285, 0.215544, 0.085846]
Breast Cancer		
$\lambda = 1$, learning rate = 0.2, steps of iterations = 300		
θ_{init}	=	[−0.817672, 0.452895, −0.122419, 0.060129, −0.082547, 0.598924, −0.137251, 0.938958, 0.904096, −0.031352]
θ^*	=	[−0.738285, 0.660637, 0.042865, 0.245485, 0.101665, 0.660190, 0.170592, 0.975785, 0.949420, 0.044775]
Banknote Authentication		
$\lambda = 1$, learning rate = 0.2, steps of iterations = 200		
θ_{init}	=	[0.050189, −0.231370, −0.855941, 0.591919, 0.470764]
θ^*	=	[−0.83083, −2.79994, −2.15228, −2.02172, 0.19787]
Adult Income		
$\lambda = 1$, learning rate = 0.4, steps of iterations = 200		
θ_{init}	=	[0.569680, −0.126640, −0.414836, −0.409631, 0.050163, 0.518757, −0.729516, −0.795351, 0.992894, 0.535617, −0.353415, 0.060383, −0.884549, 0.217820, −0.032147]
θ^*	=	[−1.890303, 0.501522, 0.190196, 0.079136, 0.153760, 0.880024, −0.694757, −0.102617, −0.313406, −0.046428, −0.207059, 0.470941, 0.310080, 0.306665, 0.010681]
Skin/NonSkin		
$\lambda = 1$, learning rate = 0.4, steps of iterations = 200		
θ_{init}	=	[0.37935, −0.73526, 0.76874, −0.45237]
θ^*	=	[2.201909, 1.842753, −0.057142, −2.681591]

accuracy reported in Table 5.

Appendix B: Security of the Employed Encryption Schemes

Our system relies on the semantic security of the homomorphic encryption schemes, recalled as follows.

Paillier encryption. It is well-known that the Paillier encryption scheme [24] has semantic security under the Decisional Composite Residuosity (DCR) assumption. Therefore, when employing Paillier encryption, our system has security under the DCR assumption.

Ring-LWE-based encryption. The ring-LWE-based scheme described in [15] has semantic security under the ring-LWE assumption (which asserts that $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s}+\mathbf{e}) \in R_q^2$ is computationally random for polynomials $\mathbf{a}, \mathbf{s} \in R_q^2$ and noise polynomial $\mathbf{e} \in R_{(0,s)}$). One can even prove that the ring-LWE ciphertext at (19) is computationally random. This is because: (1) $\mathbf{p}$ is computationally random under the ring-LWE assumption, and then the ciphertext has a one-time-pad of form $\mathbf{e}_1[\mathbf{a}|\mathbf{p}] + [\mathbf{e}_2|\mathbf{e}_3] = \mathbf{e}_1 \cdot (\mathbf{random}) + [\mathbf{e}_2|\mathbf{e}_3] \in R_q^2$ which is also computationally random under the ring-LWE assumption.

LWE-based encryption. The semantic security of the scheme is proved in [6] under the LWE assumption. It is even possible to prove that the ciphertext at (15) is computationally random under the LWE assumption as follows. First, the matrix P is of form (random matrix)×noise + noise, so is random under LWE. Second, the ciphertext at (15) has a one-time-pad $e_1[A|P] + [e_2|e_3]$ which is also of form noise×(random matrix) + noise, so it is computationally random under the LWE assumption.

Yoshinori Aono received B.S. in Engineering from the Musashi Institute of Technology in 2005. Received M.S. and Ph.D. in Mathematical and and Computing Sciences from the Tokyo Institute of Technology in 2007 and 2010, respectively. He is a researcher of 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 a Researcher of 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 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 Cybersecurity Research Institute, National Institute of Information and Communications Technology, Japan. Her research interests include cryptography and information security.