Applied Intelligence (2023) 53:10114–10129
https://doi.org/10.1007/s10489-022-04015-z

Efficient homomorphic encryption framework for privacy-preserving regression

Junyoung Byun[1] · **Saerom Park**[2] · **Yujin Choi**[1] · **Jaewook Lee**[1]

Accepted: 16 July 2022 / Published online: 15 August 2022

Abstract

Homomorphic encryption (HE) has recently attracted considerable attention as a key solution for privacy-preserving machine learning because HE can apply to various areas that require to delegate outsourcing computations of user's data. Nevertheless, its computational inefficiency still hinders its wider application. In this study, we propose an alternative to bridge the gap between the privacy and efficiency of HE by encrypting only a small amount of private information. We first derive an exact solution to HE-friendly ridge regression with multiple private variables, while linearly reducing the computational complexity of this algorithm over the number of variables. The proposed method has the advantage that it can be implemented using any HE scheme. Moreover, we propose an adversarial perturbation method that can prevent potential attacks on private variables, which have rarely been explored in HE-based machine learning studies. An extensive experiment on real-world benchmarking datasets supports the effectiveness of our method.

Keywords Homomorphic encryption · Ridge regression · Adversarial perturbation

1 Introduction

During the past few years, machine learning has shown remarkable achievements in almost every sector, including autonomous driving, marketing and finance [7, 17, 24]. The success of machine learning is due to the explosion in the number of published data, as well as advances in computational and storage capabilities represented by GPUs. However, individual data contain private information, which has raised concerns regarding the infringement of data privacy. Even if the individuals have agreed to provide such information, an unwanted level of information breach may occur if the database is attacked. Representatively, owing to the Facebook-Cambridge Analytica Data Breach in 2018, a number of people became aware of the seriousness of information leaks. Before and after, legislation such as the European General Data Protection Regulation (GDPR) [31] was enacted to protect personal privacy and information, which commonly mandates reliable protection in the use of data by third parties.

Therefore, applying security techniques in data publishing is no longer optional, but essential. Among privacy-preserving techniques, homomorphic encryption (HE) ensures data security without damaging the information contained in the data. HE enables computation on encrypted data, and it is possible to translate arbitrary machine learning techniques into a combination of homomorphic operations. HE guarantees the highest level of security as long as the scheme has not cracked, and recently proposed LWE-based HE schemes are known to be resistant against quantum attacks. The biggest factors limiting the utilization of HE in machine learning are time and storage efficiency. Currently, a state-of-the-art

✉ Jaewook Lee
jaewook@snu.ac.kr

Junyoung Byun
quswns95@snu.ac.kr

Saerom Park
psr6275@sungshin.ac.kr

Yujin Choi
uznhigh@snu.ac.kr

[1] Industrial Engineering, Seoul National University, Seoul, 08826, Republic of Korea

[2] Department of Convergence Security Engineering, Sungshin Women's University, Seoul, 02844, Republic of Korea

neural-network-training model requires 8 days for training with the encrypted MNIST dataset, which can be achieved in a few minutes with plaintext [23]. As a result, only the evaluation phase has been implemented in many studies; however, given that the main purpose of data collection is for training, model training should be further studied.

Ridge regression is an essential building block for various machine learning techniques, which models the linear relationship between input variables and a dependent variable. The most expensive part of a ridge regression learning process is a matrix inversion operation that solves a linear system, which becomes far more difficult for encrypted data. Aono et al. [2] and Morshed et al. [25] have proposed methods which obtains an approximate solution through a gradient descent instead of directly solving the linear system. Their methods are computationally costly, and setting an appropriate learning rate remains an open problem.

It is widely accepted that security for the private variables is more important than security for other variables. Cai et al. [5] and Guo et al. [14] proposed data transformation methods to prevent inference attack for private attributes, and [22] proposed a the method for dealing with multiple numerical private attributes. [11, 30] selectively encrypted the sensitive variables to provide a trade-off among privacy, time, and storage efficiency. Another example that has evident importance for sensitive variables is fair machine learning (FML), which has recently attracted significant interest [9, 16, 21]. In FML, information that can result in discrimination, such as gender or race, is regarded as a private attribute.

In this study, we propose a system to securely perform ridge regression in which only private variables are encrypted using HE. First, we present a novel efficient algorithm of ridge regression for two private variables. Then, we propose a defense method which can protect private variables from a machine learning service provider (MLSP) who conducts outsourced computations for training ML models.

The main contributions of this paper are as follows.

- Our method can be implemented with any HE scheme. Although unexpected challenges might occur when using special HE schemes in practice, because our algorithm only uses the general framework of HE that is shared by most HE schemes, our method operate effectively regardless of the scheme used and can be easily updated by an update version of HE scheme.
- We suggest a new perspective to an efficiency-security trade-off, which is currently the most notable issue in the application of HE [19]. When each ciphertext contains a different column, the complexity of our efficient ridge regression method is $O(d \log n)$, whereas the complexity of a method encrypting the entire data is $O(d^2 \log n)$. The experimental results using 11 real-world datasets support the claim.
- Our proposed defense system makes our method secure against attribute inference attacks on private variables. The method can be applied to continuous variables as well as categorical variables, and the experimental results show that our defense method effectively defend against the attacks while little affecting the performance of ridge regression.

The remainder of this paper is organized as follows. In Section 2, the basic properties of HE are provided. In Section 3, we present the entire system of our model, including the properties of the participants, their interactions, and the security goal. In Sections 4 and 5, we formalize our method and describe the proposed HE-friendly algorithm. We then present the experimental results in Section 6. Finally, some concluding remarks and areas of future study are discussed in Section 7.

2 Homorphic encryption

A general HE scheme has the following structure:

- $\mathsf{KeyGen}(1^\lambda, 1^\tau) \rightarrow (\mathsf{sKey}, \mathsf{pKey})$. Input the security parameter λ and functionality parameter τ, and output a secret key sKey and a public key pKey.
- $\mathsf{Encrypt}(\mathsf{pKey}, m) \rightarrow c$. Input the public key and a plaintext m, and output the corresponding ciphertext c.
- $\mathsf{Decrypt}(\mathsf{sKey}, c) \rightarrow m$. Input the secret key and a ciphertext c, and output the corresponding plaintext m.
- $\mathsf{Evaluate}(\mathsf{pKey}, f, \mathbf{c}) \rightarrow \mathbf{c}_f$. For a vector of ciphertexts $\mathbf{c}$ for input of a circuit f, output a vector of ciphertexts $\mathbf{c}$ for output of f.

The correctness of the scheme holds when for an arbitrary plaintext message m and its encryption c, $\mathsf{Evaluate}(\mathsf{pKey}, f, c)$ is equal to $\mathsf{Encrypt}(\mathsf{pKey}, f(m))$ where pKey is the public key generated from KeyGen [15]. Depending on f, a homomorphic encryption has various applications from simple queries, such as information retrieval, to complex machine learning algorithms [23, 26]. Studies have also been conducted to achieve more efficient LHE schemes. In [3, 13], among other studies, SIMD operations were made possible by packing several plaintexts into a single ciphertext.

In most HE schemes, noise is added to the ciphertext during the encryption process. The noise increases as it goes through a homomorphic operation. When the noise exceeds a certain threshold, the resulting ciphertext cannot be decrypted correctly. Gentry [12] presented the first FHE scheme with bootstrapping, which was based on an ideal

lattice. Bootstrapping refers to evaluating Decrypt for a ciphertext, resulting in a new ciphertext with reduced noise. However, because the decryption process is complex and nonlinear, bootstrapping is extremely costly.

When using HE, nonpolynomial functions should be approximated as polynomial functions. For instance, division operations can be approximated using Goldschmidt's algorithm. In this case, multiple iterations are required for high precision, which are the main factor of increasing the multiplicative depth.

3 Proposed framework

Our system resembles a three-party model, which is widely used in homomorphic encryption machine learning approaches. It consists of data owners, a machine learning service provider (MLSP), and a crypto-service provider (CSP). The CSP first generates a public key and a secret key for the scheme and publishes the public key to the MLSP and data owners. Data owners encrypt their private attributes with the public key and send the encrypted private attributes and other (unencrypted) variables to the MLSP. The MSLP then conducts privacy-preserving ridge regression using the data and sends the encrypted trained model parameters to the CSP. Finally, the CSP decrypts the result using the secret key and sends the result back to the data owners.

We assume that the MLSP and the CSP are honest-but-curious and do not collude with each other, as assumed in [25, 29]. Therefore, they follow the procedure correctly, although they can try to obtain hidden information from consumers by using the information given to them. The security goal under this assumption is defined as in [27].

Definition 1 [27] The proposed protocol is defined to be *secure* if the following holds.

- (Privacy of sensitive user information) Neither the CSP nor the MLSP gets any information for the private variables except for their respective outputs.
- (Model secrecy) The output of the training is not disclosed to the CSP or data owners.
- (Minimal disclosure of non-private attributes) None of the user data, including non-private variables, are exposed to the CSP.

With an appropriate security parameter, the privacy of the proposed framework can be guaranteed directly following the proof of [27]. However, in this study we investigate another possible privacy breach, in which the MLSP infers the values of private variables using unencrypted variables with the help of external knowledge. Such attacks have been referred to as linking attacks or attribute inference attacks [10, 20]. Jia and Gong [18] proposed a defense method based on the evasion attack, which is a kind of an adversarial attack. However, their method can deal with only discrete private variables, and it requires a separate optimization for each data point. Therefore, in this study, we propose a novel defense algorithm against attribute inference attack, which compensates for the shortcomings of [18]. Because the defense must be performed before data owners transfer the data to the MLSP, the existence of a reliable defender is additionally required if data owners themselves cannot perform the defense.

As another security concern, the CSP can directly decrypt the final result using the secret key. However, this type of attack can be easily prevented with a slightly higher communication cost. Data owners first determine the values for a random vector with the same length as the final result, which is called a mask. They then encrypt the mask and send it to the MLSP with the data. After the computation of the MLSP is applied, the MLSP adds the mask to the final encrypted result. Because the CSP does not know the distribution of the values for the mask, no information can be obtained from the decrypted result. The overall protocol of our system is shown in Fig. 1.

To summarize, the contribution of this study is to propose a unified framework that can efficiently perform ridge regression on multiple encrypted private variables and at the same time prevent information leakage of private variables through non-private variables. By encrypting only private variables, the proposed method offers an efficient computation compared to existing ridge regression studies using HE, including [2] and [25]. Extending the study of [4] which encrypts only sensitive information, we provide a novel method to find the ridge estimate with multiple private variables, and further enhance the privacy of the proposed method by dealing with attribute inference attacks.

To operate between ciphertexts, the number of plaintext slots for each ciphertext must be the same. Because the number of the data may vary for each owner, data owners

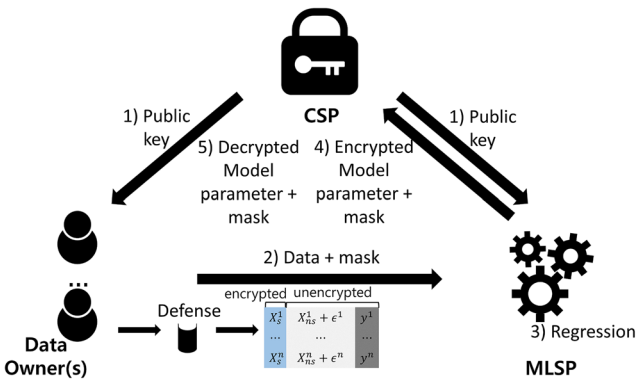


Fig. 1 Description of overall system

should determine the size of the plaintext slot in advance. That is, if the number of data points that a data owner has is r and the size of the plaintext slot is N, the data owner should split each of the columns into $\lceil r/N \rceil$ pieces, and 0-pad the last piece such that its length is N. We note that the setup is no different than having a single data owner owning multiple pieces of data. Therefore, in the next section, our method is described under the assumption that one data owner owns all of the data.

4 Methods

4.1 Efficient ridge regression with encrypted private variables

Let $(x_{i1}, \ldots, x_{id}, x_{is_1}, ..., x_{is_t}, y_i)$ be the i-th observations of d- independent *non-private* variables $\boldsymbol{X}_{ns} = (X_1, X_2, \ldots, X_d)$, and *private* variables $\boldsymbol{X}_s = (X_{s_1},..., X_{s_t})$, and one dependent variable Y. Let $h_{ij} = h_j(x_{is_j})$ be the fully homomorphic encryption of the private variable $x_{is_1}, ..., x_{is_t}$ where $(x_{i1}, ..., x_{id}, h_1(x_{is_1}), \cdots, h_t(x_{is_t}), y_i)$ for $i = 1, ...n$. We express operations with ciphertexts as operations with plaintexts.

By using Sherman-Woodbury inversion formula and singular value decomposition (SVD), the ridge estimate can be obtained efficiently as in [4]:

$$\begin{aligned}\hat{\boldsymbol{r}} &= \sum_{i=1}^{d} \frac{\sigma_i^2}{\sigma_i^2 + \lambda} \boldsymbol{u}_i (\boldsymbol{u}_i^T \boldsymbol{y}) \\ &\quad + \lambda \left(\sum_{j=1}^{d+t} \frac{\boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{H}_s}{\sigma_j^2 + \lambda} \right) (\boldsymbol{I}_t + \boldsymbol{\xi})^{-1} \boldsymbol{\eta}, \end{aligned} \tag{1}$$

where $\boldsymbol{u}_j$, $j = 1, \ldots, d$ are d orthogonal vectors obtained from SVD of $\boldsymbol{X}_{ns}$ whose corresponding singular values are not 0. Also, for $k = 1, ..., t$,

$$\begin{aligned}\hat{\boldsymbol{u}}_{d+k} &= \boldsymbol{h}_k - \sum_{i=1}^{d+k-1} \boldsymbol{u}_i (\boldsymbol{u}_i^T \boldsymbol{h}_k) \\ \boldsymbol{u}_{d+k} &= \hat{\boldsymbol{u}}_{d+k} / \|\hat{\boldsymbol{u}}_{d+k}\|, \end{aligned} \tag{2}$$

and

$$\begin{aligned}\boldsymbol{\eta} &= \sum_{j=1}^{d+t} \frac{1}{\sigma_j^2 + \lambda} \boldsymbol{H}_s^T \boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{y} \\ \boldsymbol{\xi} &= \sum_{j=1}^{d+t} \frac{1}{\sigma_j^2 + \lambda} \boldsymbol{H}_s^T \boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{H}_s. \end{aligned} \tag{3}$$

When there is a single private variable, (1) can be easily computed [4] because

$$\boldsymbol{u}_{d+1} \boldsymbol{u}_{d+1}^T \boldsymbol{h}_1 = \boldsymbol{h}_1 - \sum_{i=1}^{p} \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{h}_1. \tag{4}$$

However, their method cannot be naturally extended to the case with more private variables. For example, when $t = 2$, (4) does not hold for $\boldsymbol{u}_{d+2}$. It is not trivial to evaluate the (2) because a square root and a division are required, which makes the algorithm not efficient in the encrypted domain. To overcome this limitation, we propose an efficient algorithm for two private variables.

Let the SVD of $\boldsymbol{X}_{ns} = U \Sigma V^T$ where $U \in \Re^{n \times n}$ and $V \in \Re^{d \times d}$ are orthogonal matrices and $\Sigma \in \Re^{n \times d}$ are diagonal matrix with diagonal entries $\sigma_1 \geq .... \geq \sigma_d \geq \sigma_{d+1} = \cdots = \sigma_n = 0$. Then $U = [U_1, U_2]$ where $U_1 = [\boldsymbol{u}_1, ..., \boldsymbol{u}_d] \in \Re^{n \times d}$ and $U_2 = [\boldsymbol{u}_{d+1}, ..., \boldsymbol{u}_n] \in \Re^{n \times (n-d)}$ and $U_1 \perp U_2$. Because the corresponding singular values are 0, there is freedom to choose U_2. We can obtain the following equations through the orthogonalization process that is sequentially applied to $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$.

$$\begin{aligned}&\hat{\boldsymbol{u}}_{d+1} = \boldsymbol{h}_1 - \sum_{i=1}^{d} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_1 \right) \\ &\boldsymbol{u}_{d+1} = \hat{\boldsymbol{u}}_{d+1} / \|\hat{\boldsymbol{u}}_{d+1}\| \\ &\boldsymbol{u}_{d+1} \left(\boldsymbol{u}_{d+1}^T \boldsymbol{h}_1 \right) = \boldsymbol{h}_1 - \sum_{i=1}^{d} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_1 \right) \\ &\hat{\boldsymbol{u}}_{d+2} = \boldsymbol{h}_2 - \sum_{i=1}^{d+1} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_2 \right), \quad \boldsymbol{u}_{d+2} = \hat{\boldsymbol{u}}_{d+2} / \|\hat{\boldsymbol{u}}_{d+2}\|. \end{aligned} \tag{5}$$

In addition, we found that without loss of generality, the orthogonalization process can be sequentially applied to $\boldsymbol{h}_2$ and $\boldsymbol{h}_1$. That is,

$$\begin{aligned}&\hat{\boldsymbol{u}}'_{d+1} = \boldsymbol{h}_2 - \sum_{i=1}^{d} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_2 \right) \\ &\boldsymbol{u}'_{d+1} = \hat{\boldsymbol{u}}'_{d+1} / \|\hat{\boldsymbol{u}}'_{d+1}\| \\ &\boldsymbol{u}'_{d+1} \left(\boldsymbol{u}'^T_{d+1} \boldsymbol{h}_2 \right) = \boldsymbol{h}_2 - \sum_{i=1}^{d} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_2 \right) \\ &\hat{\boldsymbol{u}}'_{d+2} = \boldsymbol{h}_1 - \sum_{i=1}^{d} \boldsymbol{u}_i \left(\boldsymbol{u}_i^T \boldsymbol{h}_1 \right) - \boldsymbol{u}'_{d+1} \left(\boldsymbol{u}'^T_{p+1} \boldsymbol{h}_1 \right) \\ &\boldsymbol{u}'_{d+2} = \hat{\boldsymbol{u}}'_{d+2} / \|\hat{\boldsymbol{u}}'_{d+2}\|. \end{aligned} \tag{6}$$

Therefore, even if $\boldsymbol{u}_{d+2} \boldsymbol{u}_{d+2}^T \boldsymbol{h}_2$ cannot be computed efficiently in (5), $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ can be further reduced by using

both (5) and (6) as

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\lambda}\left(\boldsymbol{y}^T\boldsymbol{h}_1 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{h}_1^T\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{y}\right) \\ \frac{1}{\lambda}\left(\boldsymbol{y}^T\boldsymbol{h}_2 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{h}_2^T\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{y}\right) \end{bmatrix} \quad (7)$$

$$\boldsymbol{\xi} = \begin{bmatrix} \xi_{11} & \xi_{12} \\ \xi_{21} & \xi_{22} \end{bmatrix}$$

$$\xi_{11} = \frac{1}{\lambda}\left(\boldsymbol{h}_1^T\boldsymbol{h}_1 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{h}_1^T\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{h}_1\right)$$

$$\xi_{12} = \xi_{21} = \frac{1}{\lambda}\left(\boldsymbol{h}_1^T\boldsymbol{h}_2 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{h}_1^T\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{h}_2\right)$$

$$\xi_{22} = \frac{1}{\lambda}\left(\boldsymbol{h}_2^T\boldsymbol{h}_2 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{h}_2^T\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{h}_2\right) \quad (8)$$

Finally, the ridge estimation is

$$\begin{aligned}\hat{\boldsymbol{r}} &= \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j\left(\boldsymbol{u}_j^T\boldsymbol{y}\right) \\ &+ \frac{(\eta_1+\xi_{22}\eta_1-\xi_{12}\eta_2)}{(1+\xi_{11})(1+\xi_{22})-\xi_{12}^2}\left(\boldsymbol{h}_1 - \sum_{j=1}^{d}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j\left(\boldsymbol{u}_j^T\boldsymbol{h}_1\right)\right) \\ &+ \frac{(\eta_2+\xi_{11}\eta_2-\xi_{12}\eta_1)}{(1+\xi_{11})(1+\xi_{22})-\xi_{12}^2}\left(\boldsymbol{h}_2 - \sum_{j=1}^{p}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j\left(\boldsymbol{u}_j^T\boldsymbol{h}_2\right)\right). \quad (9)\end{aligned}$$

4.2 Adversarial perturbation against attribute inference attack

Let a classification network $\mathcal{A}$ denote the attacker's classifier. For one private variable $\boldsymbol{x}_s$ and non-private variables $\boldsymbol{X}_{ns}$, let public and private datasets $D^{pub} = (\boldsymbol{X}_{ns}^{pub}, \boldsymbol{x}_s^{pub})$ and $D^{priv} = (\boldsymbol{X}_{ns}^{priv}, \boldsymbol{x}_s^{priv})$. $\mathcal{A}$ is trained to minimize a loss function $\mathcal{L}_{att}(\mathcal{A}(\boldsymbol{X}_{ns}^{pub}), \boldsymbol{x}_s^{pub})$. $\mathcal{L}_{att}$ can be determined depending on what kind of variable $\boldsymbol{x}_s$ is. For example, the attacker can use cross entropy loss if $\boldsymbol{x}_s$ is a discrete variable, and the MSE loss can be used if it is a continuous variable. For t private variables, the attacker can train t models separately.

On the other hand, the goal of the defender is to maximize $\mathcal{L}_{att}(\mathcal{A}(\boldsymbol{X}_{ns}^{priv} + \boldsymbol{\epsilon}), \boldsymbol{x}_s^{priv})$ by adding a little noise $\boldsymbol{\epsilon}$ to $\boldsymbol{X}_{ns}^{priv}$. Because the defender does not have information of the model the attacker used, the defender trains a model $\mathcal{D}$ with D^{pub} and use it as an alternative of $\mathcal{A}$. As explained in [18], using a surrogate model is plausible because of the transferability between the two models. Based on the evasion attack which is a kind of an adversarial attack, [18] considered $\boldsymbol{\epsilon}$ to be the minimum noise that changes the classification result of $\boldsymbol{X}_{ns}^{priv}$. However, their method has a fundamental problem that it cannot be applied for a continuous $\boldsymbol{x}_s$. Also, it has a problem of scalability because an optimization should be solved for each row of $\boldsymbol{X}_{ns}^{priv}$.

To address both of the above problems, we propose a noise generating network $\mathcal{G}_\theta$ as

$$\mathcal{G}_\theta(\boldsymbol{X}_{ns}) = \mathbb{E}[\boldsymbol{\epsilon}|\boldsymbol{X}_{ns}]. \quad (10)$$

Because the dimensionality of the noise is same as the dimensionality of the input, we use an autoencoder structure for $\mathcal{G}_\theta$. The generator $\mathcal{G}_\theta$ is trained to maximize the loss of the defender $\mathcal{D}$, i.e.,

$$\theta = \arg\max_{\theta'}\mathcal{L}_{def}(\mathcal{D}(\boldsymbol{X}_{ns}^{priv} + \mathcal{G}_\theta(\boldsymbol{X}_{ns}^{priv})), \boldsymbol{x}_s^{priv}). \quad (11)$$

We note that it is also possible to make the output of $\mathcal{G}_\theta$ become (input+noise). However, in this case it is difficult to directly limit the magnitude of the noise. On the other hand, if the output of $\mathcal{G}_\theta$ is used as noise as in (11), the noise level can be directly adjusted by clipping the output to the desired magnitude. The overall flow of our defense method is depicted in Fig. 2.

Our defense method can be seen as adopting a generative adversarial perturbation (GAP) [28], a kind of adversarial attack which generates an adversarial attack with a neural network, to the task of defending against attribute inference attacks. However, our method can support both classification model for discrete private variables and regression model for continuous private variables, whereas GAP concentrates on attacking a classification model.

5 Algorithms

5.1 CKKS scheme

We first summarize the basic algorithms of the CKKS scheme used for the experiments to form the building blocks that compose our main algorithm. CKKS is one of the most widely used HE schemes, which supports computation of approximate numbers by considering the noise for the hardness assumption as part of error that occurs during arithmetic. As a result, CKKS solves floating-point operations efficiently by introducing a bounded loss of precision. Considering that there is always a numerical error in the computation of a machine, this trade-off is beneficial for the purpose of machine learning. Also, CKKS supports SIMD operations by encoding a complex vector (with size at most $N/2$) into a ring element. Algorithm 1 demonstrates the procedures used for key generation, encryption and decryption. As an important feature of CKKS, it has a unique method of encoding a plaintext vector into a ring element prior to encryption. Through Encode, CKKS supports encryption for complex vectors,

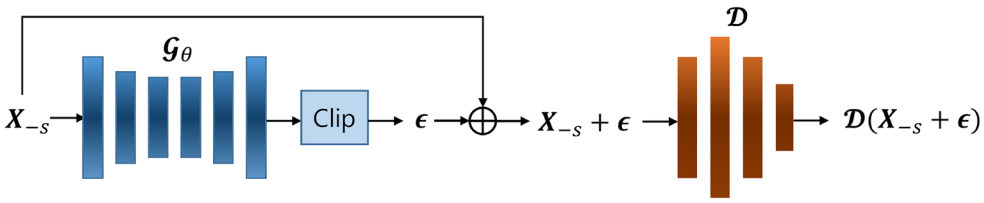


Fig. 2 Architecture for adversarial perturbation against attribute inference attack

and naturally enables a SIMD operation as well. Decode is almost the same as the inverse of Encode and is used after Decrypt.

1: KeyGen$(1^\lambda) \rightarrow$ (sKey, pKey, evKey) ▷ Key generation
2: Encode$(\vec{z}) \rightarrow m \in \mathcal{R}$ ▷ Complex vector to polynomial ring
3: Decode$(m) \rightarrow \vec{z} \in \mathbb{C}^{N/2}$ ▷ Polynomial ring to complex vector
4: Encrypt(pKey, m) $\rightarrow c$ ▷ Encryption
5: Decrypt(sKey, c) $\rightarrow m$ ▷ Decryption

Algorithm 1 Basic Algorithms.

1: ConstAdd$(c, v \in \mathcal{R}) \rightarrow c_{\text{add}}$ ▷ Addition between ciphertext and plaintext
2: Add$(c, c') \rightarrow c_{\text{add}}$ ▷ Addition between ciphertexts
3: Sub$(c, c') \rightarrow c_{\text{sub}}$ ▷ Subtraction between ciphertexts
4: ConstMult$(c, v \in \mathcal{R}) \rightarrow c_{\text{mult}}$ ▷ Multiplication between ciphertext and plaintext
5: Mult(evKey, c, c') $\rightarrow c_{\text{Mult}}$ ▷ Multiplication between ciphertexts
6: Rotate(c, j) $\rightarrow c_{\text{Rotate}}$ ▷ Rotation of a ciphertext for j slots

Algorithm 2 Operation Algorithms.

Algorithm 2 presents the procedures for addition and multiplication operations. By combining these procedures, the evaluation of an arbitrary function is performed. Whereas Add and Mult refer to the addition and multiplication between ciphertexts, ConstAdd and ConstMult refer to operations between a ciphertext and a constant polynomial. In any method that supports parallel operation as well as CKKS, since these operations are performed in units of slots, as many operations as the number of plaintext slots can be performed at a time. The scheme additionally includes Rotate and Rescale procedures. Rotate is slot-wise rotation, and is are particularly useful when adding values in the same ciphertext, and enable efficient parallel operations. Rescale is appended after every ConstMult or Mult to manage the magnitude of the error. Rescale reduces the ciphertext modulus, which limits the number of operations performed without bootstrapping. For a more detailed description of the scheme and the algorithms, refer to [6]. It should emphasized that most operations are included in the general scheme described in Section 2.

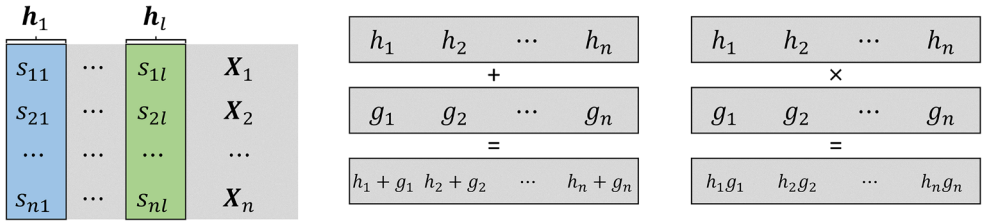
5.2 Algorithm for ridge regression

Because (9) requires as many summations as the number of variables, we put each private variable in a different ciphertext, and all values for each column are grouped into a single ciphertext. Some details about our ciphertext packing is presented in the appendix. The maximum number of plaintext values that can be packed into one ciphertext is determined based on the security parameter. Here, we assume that the length of the columns does not exceed this parameter. Otherwise, each column should be divided into multiple ciphertexts, but this does not significantly affect the complexity of the algorithm. When a ciphertext contains a scalar value such as η_1 or ξ_{11}, it can be thought of as a vector that repeats the same value as many times as the number of slots. Our packing strategy, and addition and multiplication between ciphertexts are depicted in Fig. 3.

Algorithm 3 demonstrates the use of our method for a ridge regression with two encrypted private variables. The basic operations used in the algorithm are presented in the appendix. For convenience, Rescale after any multiplication is omitted. Using Rotate, the sum of n elements in a ciphertext can be calculated within $\log n$ times. Therefore, the complexity of Algorithm 3 is $O(d \log n)$. Inv is an evaluation of Goldschmidt's division algorithm with τ times of iterations.

5.3 Algorithm for adversarial perturbation

Algorithm 4 presents the procedures for defending against attribute inference attacks. The first phase of the algorithm is to train an attack model which aims to predict the value of private variables. With the trained attack model, in the second phase the noise generator network is trained to degrade the performance of the attack model for private data. We note that cl in the second phase is the clipping function which bounds the norm of the generated noise to



(a) Packing (b) Addition (c) Multiplication

Fig. 3 Description of the ciphertext packing and operations between ciphertexts. The values in the thick box are packed into the same ciphertext

C. Any norm such as L_1-norm and L_2-norm can be used for clipping, and the trade-off between the degree of defense and the regression utility can be controlled by adjusting the value of C.

6 Experiments

In this section, we evaluate the efficiency of our method on 11 real-world datasets. We then empirically show the

procedure RidgeEstimate($\mathbf{h}_1, \mathbf{h}_2, \{\mathbf{u}_i\}, \{\sigma_i\}, \mathbf{y}, \lambda, \tau$) ▷ calculate the ridge estimate
  for s in $1, 2$ **do**
    for i in $1, \ldots, d$ **do**
      $tmp \leftarrow \mathsf{ConstMult}(\mathbf{h}_s, \mathbf{u}_i)$
      for j in $1, \ldots, \log_2 n$ **do**
        $tmp\_rot \leftarrow \mathsf{Rotate}(tmp, -2^j)$
        $tmp \leftarrow \mathsf{Add}(tmp, tmp\_rot)$
      end for
      $tmp \leftarrow \mathsf{ConstMult}(tmp, \frac{\sigma_i^2}{\sigma_i^2+\lambda}\mathbf{u}_i)$
      if i=0 **then**
        $\mathbf{c}_s \leftarrow tmp$
      else
        $\mathbf{c}_s \leftarrow \mathsf{Add}(\mathbf{c}_s, tmp)$
      end if
    end for
    $\mathbf{c}_s \leftarrow \mathsf{Sub}(\mathbf{h}_s, \mathbf{c}_s)$
  end for
  $c_3 \leftarrow \mathsf{ConstMult}(\mathbf{c}_1, \frac{1}{\lambda}\mathbf{y})$; $c_4 \leftarrow \mathsf{ConstMult}(\mathbf{c}_2, \frac{1}{\lambda}\mathbf{y})$
  $c_5 \leftarrow \mathsf{Mult}(\mathbf{c}_1, \mathsf{ConstMult}(\mathbf{h}_1, \frac{1}{\lambda}))$; $c_6 \leftarrow \mathsf{Mult}(\mathbf{c}_2, \mathsf{ConstMult}(\mathbf{h}_2, \frac{1}{\lambda}))$;
  $c_7 \leftarrow \mathsf{Mult}(\mathbf{c}_1, \mathsf{ConstMult}(\mathbf{h}_2, \frac{1}{\lambda}))$
  for i in $3, \ldots, 7$ **do**
    for j in $1, \ldots, log_2 n$ **do**
      $c_i\_rot \leftarrow \mathsf{Rotate}(c_i, -2^j)$
      $c_i \leftarrow \mathsf{Add}(c_i, c_i\_rot)$
    end for
  end for
  $c_8 \leftarrow \mathsf{Sub}(\mathsf{Add}(c_3, \mathsf{Mult}(c_3, c_7)), \mathsf{Mult}(c_4, c_6))$
  $c_9 \leftarrow \mathsf{Sub}(\mathsf{Add}(c_4, \mathsf{Mult}(c_4, c_5)), \mathsf{Mult}(c_3, c_6))$
  $c_{10} \leftarrow \mathsf{Inv}(\mathsf{Sub}(\mathsf{Mult}(\mathsf{ConstAdd}(\mathbf{c}_5, 1), \mathsf{ConstAdd}(c_7, 1)), \mathsf{Mult}(c_6, c_6)), \tau)$
  $\hat{\boldsymbol{r}} \leftarrow \mathsf{Add}(\mathsf{Mult}(c_{10}, \mathsf{Mult}(c_8, c_1)), \mathsf{Mult}(c_{10}, \mathsf{Mult}(c_9, c_2)))$
  $\hat{\boldsymbol{r}} \leftarrow \mathsf{ConstAdd}(\hat{\boldsymbol{r}}, \sum_{i=1}^{d} \frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{y})$
  return $\hat{\boldsymbol{r}}$
end procedure

Algorithm 3 Ridge Regression for two Sensitive Variables.

procedure TrainAttack(public non-private data $\{X_{ns}^{pub}\}_{k=1}^{n_{pub}}$, public private variable $\{x_s^{pub}\}_{k=1}^{n_{pub}}$, number of epochs e_1, batch size b_1, learning rate ℓ_1)
  Initialize parameters ϕ of $\mathcal{D}_\phi$
  for $1, \ldots, e_1$ **do**
    for $1, \ldots, \lceil n_{pub}/b_1 \rceil$ **do**
      Sample mini-batches of b_1 samples B
      $\mathcal{L} = \sum_{(X_{ns}^{pub}, x_s^{pub}) \in B} \mathcal{L}_{def}(\mathcal{D}_\phi(X_{ns}^{pub}), x_s^{pub})$
      $\phi \leftarrow \phi - \ell_1 \nabla_\phi \mathcal{L}$
    end for
  end for
end procedure
procedure TrainNoise(private non-private data $\{X_{ns}^{priv}\}_{k=1}^{n_{priv}}$, private private variable $\{x_s^{priv}\}_{k=1}^{n_{priv}}$, number of epochs e_2, batch size b_2, learning rate ℓ_2, clipping parameter C)
  Initialize parameters θ of $\mathcal{G}_\theta$
  for $1, \ldots, e_2$ **do**
    for $1, \ldots, \lceil n_{priv}/b_2 \rceil$ **do**
      Sample mini-batches of b_2 samples B
      $\mathcal{L} = \sum_{(X_{ns}^{priv}, x_s^{priv}) \in B} \mathcal{L}_{def}(\mathcal{D}_\phi(X_{ns}^{priv} + \mathsf{cl}(\mathcal{G}_\theta(X_{ns}^{priv}), C)), x_s^{priv})$
      $\theta \leftarrow \theta + \ell_2 \nabla_\theta \mathcal{L}$
    end for
  end for
end procedure

Algorithm 4 Defense against Attribute Inference Attack.

security of our defense method against attribute inference attacks by measuring the prediction performance for the private attributes. In addition, we present the prediction performance for the target variable of ridge regression with and without the defense method.

6.1 Settings

6.1.1 Datasets

We used ten real-world datasets from the UCI machine learning repository [8], along with another dataset from [1]. Among the 11 datasets, 7 were originally generated for classification tasks and have binary target variables. Before encryption, all input variables, including private variables, were scaled to have values within [0, 1]. In addition, we used only numerical and binary categorical variables. We considered this to be more advantageous to the compared method, because the storage cost of our approach does not significantly decrease even if the number of variables decreases, due to the fact that our method only encrypts a fixed number of private variables. A detailed description of the datasets is shown in Table 1.

Because the criterion for determining a private variable differs from study to study, we measured the feature importance through a ridge regression on plaintext with Python and treated the variable with higher feature importance as a private variable. In real situations, the data owner can determine the private variables according to their level of importance.

6.1.2 Compared methods

We compared our method with three methods as follows:

- **Baseline**: A ridge regression method without any encrypted variable. It serves as a benchmark for regression performance.
- **1Sens**: A method that encrypts one private variable, proposed in [4].
- **FullColumn**: A method that encrypts all input variables implemented with CKKS. Following [2, 25], it was trained using a gradient descent.
- **Ours**: Our proposed ridge regression method with two encrypted private variables.

6.1.3 Experimental details

All experiments were carried on a machine using 40 threads of an Intel Xeon E-2660 v3 @2.60_GHz CPU processors.

Table 1 Description of the datasets

Dataset	Columns	Total Data points	Binary Target variable	Learning rate (FullColumn)
Australian Credit (Credit)	10	690	yes	0.01
Bupa Liver Disorders (Disorders)	5	345	no	0.1
Heart Disease (Heart)	8	303	yes	0.01
Pima Indians Diabetes (Indians)	8	768	yes	0.01
Wisconsin Breast Cancer (Cancer)	9	698	yes	0.001
Bank Marketing (Bank)	10	4521	yes	0.001
Student Performance (Student)	24	395	no	0.01
Graduate Admission (Admission)	7	400	no	0.01
Diabetes (Diabetes)	10	442	no	0.01
Adult (Adult)	11	30162	yes	0.0001
Census Income (Census)	18	199523	yes	0.00001

For the CKKS parameters, we set $\log q_L = 1200$, $\log p = 40$ and $N = 2^{16}$. We tested our method by changing the value of λ in $\{0.1, 0.5, 1, 5, 10\}$. The learning rates for the gradient descent method were predetermined within the range of $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ before the experiment with plaintext to show the best performance, and the number of iterations was fixed to 8 to avoid bootstrapping. The selected learning rates for each dataset are shown in the appendix.

Because the efficiency of homomorphic encryption is highly dependent on the packing strategy for SIMD operations, for a fair comparison we packed each column (attribute) into one ciphertext for all datasets except **Census**. Because the number of training data is larger than the maximum number of plaintext values that can be packed into one ciphertext ($= N/2$), for **Census** dataset we split each column into three ciphertexts. We empirically compared the computational time and R^2 score of our method with those of **1Sens** and **FullColumn**. All experiments were repeated five times, and the average of each measurement was recorded. We split each dataset into three sets. First, 50% of each dataset was randomly sampled as the public set. The public set was used for training classifiers of the attacker and the defender, and the defense method. Then, we used 80% of the remaining data as the training set for the ridge regression, and the remaining 20% as the test set. The models of the attacker and the defender were set as feed-forward neural networks with one hidden layer, and the defense method was set as an autoencoder with one hidden layer. The size of the hidden layer of the attacker's network was set to be 3 times the number of the variables, and the size of the hidden layer of the other networks were set to be 5 times the number of the variables.

6.2 Results

6.2.1 Computation time

Table 2 shows the computation time for the 11 datasets with a regularization parameter $\lambda = 1$. It is shown that owing to encrypting only private variables, our method is 5-20 times faster than **FullColumn** depending no the number of variables. In addition to the performance of each method, we plotted in Fig. 4 the performance ratio of **FullColumn** and **Ours**, as well as the performance ratio of **Ours** and **1Sens**.

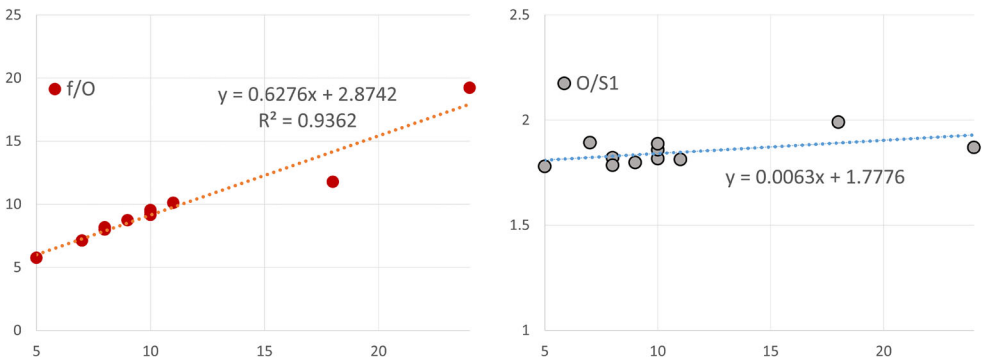
Because the computation cost of each method is roughly proportional to the number of columns, and because we packed each column as a ciphertext, we predicted that the performance ratio would be proportional to the number of columns in the dataset. Therefore, if the ratio value is higher than the number of variables divided by 2, **Ours** is more efficient, and if the value is lower, the opposite can be considered. Figure 4(a) demonstrates that **Ours** was always more efficient than **FullColumn**, especially with fewer variables. In addition, our method does not require searching from the learning rate because it does not use a gradient descent. Although is difficult to quantitatively express the time taken for parameter search, we emphasize that this fact makes our method far more efficient.

Figure 4(b) shows that **Ours** required about 1.7-1.9 times of computation time compared to **1Sens**. The reason that ratio is always smaller than 2 is that **Ours** exploits one less non-private variable by assuming two private variables. Because Algorithm 3 requires as many iterations as the number of non-private variables, the computation time of **Ours** is slightly reduced. It can be seen from the figure that the ratio tends to approach 2 as the number of variables

Table 2 Results for computation time and regression performance with $\lambda = 1$

Dataset	Time (sec)			R^2 score			
	FullColumn	1Sens	Ours	FullColumn	1Sens	Ours	Baseline
Credit	744.466	42.951	78.029	0.5162	**0.5178**	**0.5178**	0.5177
Disorders	245.907	24.039	42.789	0.1416	**0.1573**	**0.1573**	0.1573
Heart	452.905	31.141	56.707	**0.1974**	0.1739	0.1737	0.1740
Indians	533.078	36.556	65.258	0.2056	**0.2421**	**0.2421**	0.2421
Cancer	630.344	40.146	72.184	**0.7729**	0.7644	0.7492	0.7644
Bank	851.390	50.069	93.076	-0.0283	**0.1757**	**0.1757**	0.1757
Student	2959.940	82.322	153.976	**0.0810**	0.0676	0.0660	0.0677
Admission	396.904	29.428	55.701	0.7635	0.8118	**0.8136**	0.8137
Diabetes	684.247	38.670	73.004	0.3972	**0.4308**	**0.4308**	0.4308
Adult	1166.396	63.678	115.406	0.2359	**0.3230**	**0.3230**	0.3230
Census	7983.130	343.381	680.125	0.1036	**0.1948**	0.1947	0.1948

The best results are shown in bold



(a) Comparison with **Fullcolumn** (b) Comparison with **1Sens**

Fig. 4 Visualization of the results for the computation time. The x-axis indicates the number of variables, and the y-axis indicates the ratio of the computation times

Table 3 Results for computation time and regression performance with $\lambda = 0.1$

Dataset	Time (sec)			R^2 score			
	FullColumn	1Sens	Ours	FullColumn	1Sens	Ours	Baseline
Credit	757.938	42.958	78.024	0.5159	**0.5170**	0.5169	0.5146
Disorders	242.810	23.502	41.849	0.1616	0.2170	**0.2181**	0.2192
Heart	454.040	31.159	57.386	**0.1998**	0.1506	0.1494	0.1506
Indians	529.409	36.753	63.641	0.2093	**0.2330**	0.2251	0.2330
Cancer	629.056	40.116	72.194	**0.7733**	0.7609	0.7609	0.7609
Bank	847.323	50.953	93.129	-0.0284	**0.1761**	**0.1761**	0.1761
Student	2929.000	83.062	154.588	**0.0792**	0.0449	0.0462	0.0462
Admission	395.290	29.510	56.615	0.7658	**0.8203**	**0.8203**	0.8203
Diabetes	684.568	38.049	72.691	0.4025	**0.4217**	0.4216	0.4217
Adult	1165.470	61.642	114.676	0.2359	**0.3233**	0.3232	0.3232
Census	7913.180	343.083	649.067	0.1036	**0.1948**	**0.1948**	0.1948

The best results are shown in bold

Table 4 Results for computation time and regression performance with $\lambda = 0.5$

Dataset	Time (sec)			R^2 score			
	FullColumn	1Sens	Ours	FullColumn	1Sens	Ours	Baseline
Credit	743.300	43.743	79.265	0.5161	0.5099	**0.5192**	0.5163
Disorders	249.023	23.462	43.747	0.1522	**0.1847**	0.1844	0.1847
Heart	452.676	31.259	56.785	**0.1988**	0.1632	0.1614	0.1632
Indians	534.419	37.004	66.870	0.2077	**0.2332**	0.2318	0.2389
Cancer	634.045	40.317	73.069	**0.7732**	0.7626	0.7626	0.7626
Bank	862.840	50.174	94.006	-0.0283	**0.1759**	**0.1759**	0.1759
Student	2961.280	82.137	155.392	**0.0800**	0.0575	0.0575	0.0575
Admission	393.240	29.337	55.781	0.7648	0.8179	**0.8180**	0.8180
Diabetes	678.350	39.419	72.903	0.4001	0.4202	**0.4273**	0.4273
Adult	1164.760	62.536	113.801	0.2359	0.3232	**0.3246**	0.3231
Census	7936.260	336.299	685.656	0.1036	**0.1948**	0.1943	0.1948

The best results are shown in bold

Table 5 Results for computation time and regression performance with $\lambda = 5$

Dataset	Time (sec)			R^2 score			
	FullColumn	1Sens	Ours	FullColumn	1Sens	Ours	Baseline
Credit	740.849	42.767	78.723	0.5161	**0.5191**	**0.5191**	0.5191
Disorders	246.376	23.943	42.801	**0.0918**	**0.0918**	0.0891	0.0918
Heart	451.372	30.740	56.695	0.1842	**0.1904**	**0.1904**	0.1904
Indians	542.671	35.884	67.632	0.1897	0.2220	**0.2221**	0.2221
Cancer	633.365	40.231	71.659	0.7681	**0.7716**	0.7692	0.7716
Bank	844.614	49.746	91.668	-0.0278	**0.1739**	0.1738	0.1739
Student	2965.350	81.926	152.499	0.0860	0.0977	**0.0981**	0.0981
Admission	401.839	29.751	53.882	0.7504	**0.7782**	**0.7782**	0.7782
Diabetes	698.430	39.075	74.380	0.3721	0.3849	**0.4070**	0.4096
Adult	1164.140	65.187	114.903	0.2358	0.3224	**0.3234**	0.3223
Census	8035.450	342.837	680.125	0.1037	**0.1946**	**0.1946**	0.1946

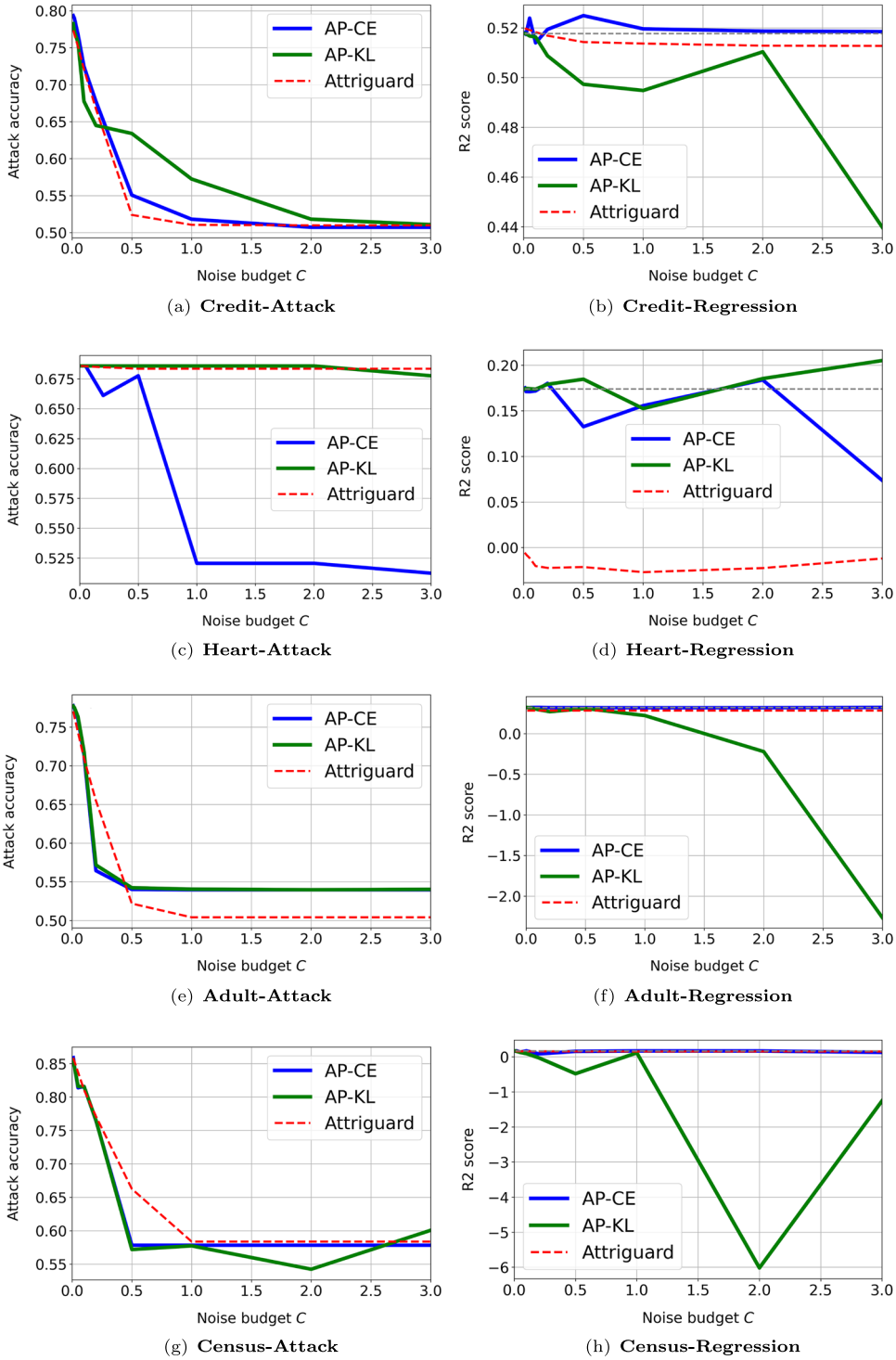
The best results are shown in bold

Table 6 Results for computation time and regression performance with $\lambda = 10$

Dataset	Time (sec)			R^2 score			
	FullColumn	1Sens	Ours	FullColumn	1Sens	Ours	Baseline
Credit	740.320	42.745	78.529	0.5127	**0.5146**	**0.5146**	0.5146
Disorders	245.705	25.111	42.637	0.0643	**0.0655**	**0.0655**	0.0655
Heart	450.651	31.255	56.102	0.1641	**0.1713**	0.1711	0.1713
Indians	528.869	36.966	63.694	0.1713	0.1884	**0.1891**	0.1891
Cancer	632.784	40.219	73.061	0.7532	**0.7730**	0.7723	0.7731
Bank	859.006	49.864	93.128	-0.0272	**0.1690**	**0.1690**	0.1690
Student	2941.460	81.514	153.395	0.0874	0.0951	**0.0989**	0.0989
Admission	397.849	29.515	56.652	0.7295	**0.7433**	0.7430	0.7433
Diabetes	681.683	38.734	71.069	0.3398	**0.3635**	0.3633	0.3635
Adult	1171.130	64.754	119.145	0.2358	**0.3214**	**0.3214**	0.3214
Census	7849.710	336.474	688.455	0.1037	**0.1938**	**0.1938**	0.1938

The best results are shown in bold

Springer



(a) **Credit-Attack** (b) **Credit-Regression**

(c) **Heart-Attack** (d) **Heart-Regression**

(e) **Adult-Attack** (f) **Adult-Regression**

(g) **Census-Attack** (h) **Census-Regression**

Fig. 5 Defense results against attribute inference attack (first column) and ridge regression results with defense methods (second column) for datasets with discrete private variable

increases. The results for different values of λ are provided in Tables 3, 4, 5 and 6.

Meanwhile, the results for **Census** dataset in which each column is divided into multiple ciphertexts deviate slightly from the overall trend (second point from the right in Fig. 4). If the number of ciphertexts for each column increases, the operation time of all methods increases approximately in proportion to it. However, not every part of the algorithm becomes slower. Taking Algorithm 3 as an example, while the number of operations to calculate c_1 to c_7 (line 2 to line 26) increases by almost exactly the number of ciphertexts for each column, the next part is no longer affected by the number of ciphertexts. Because the part that is affected by the number of ciphertexts is relatively larger than the

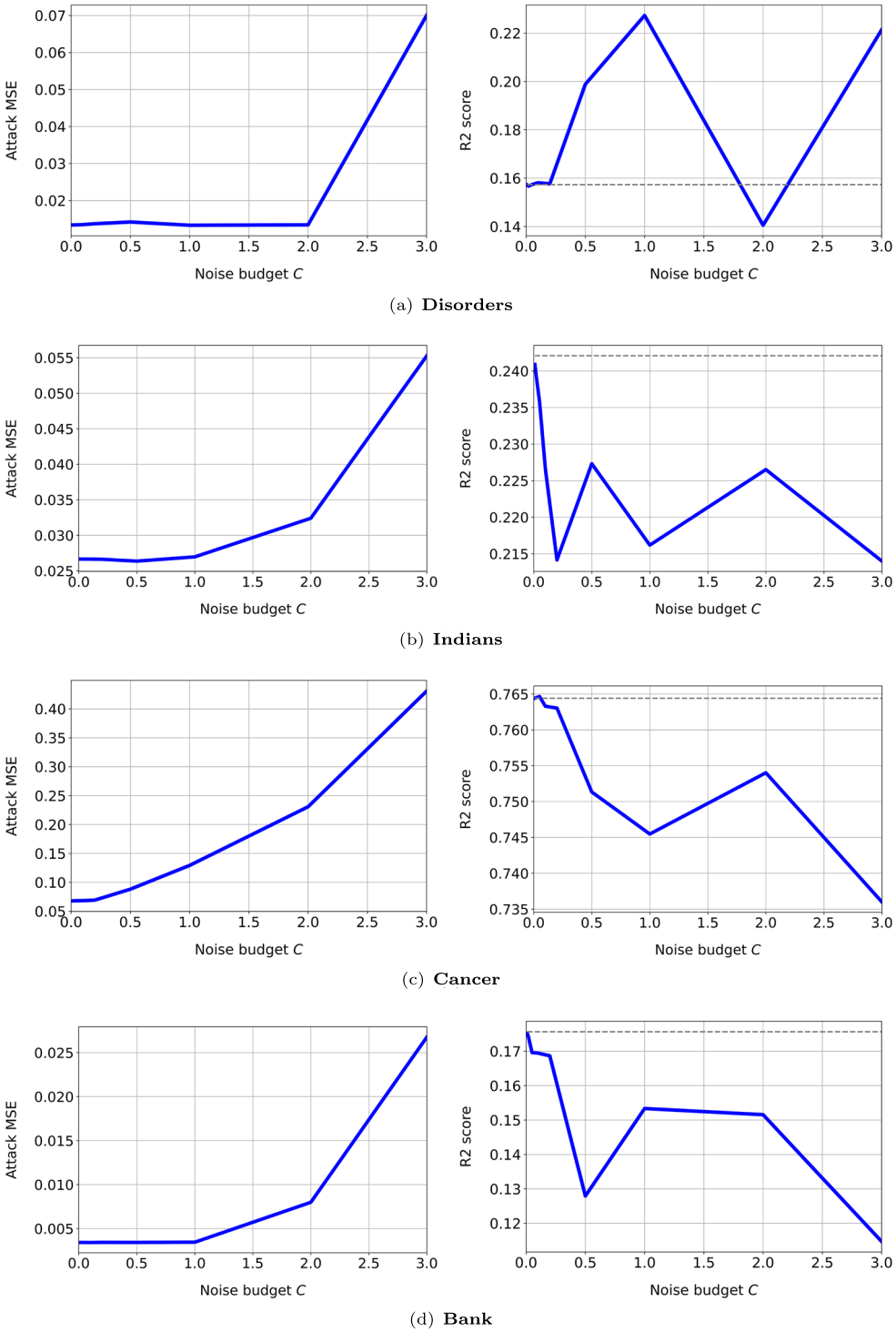


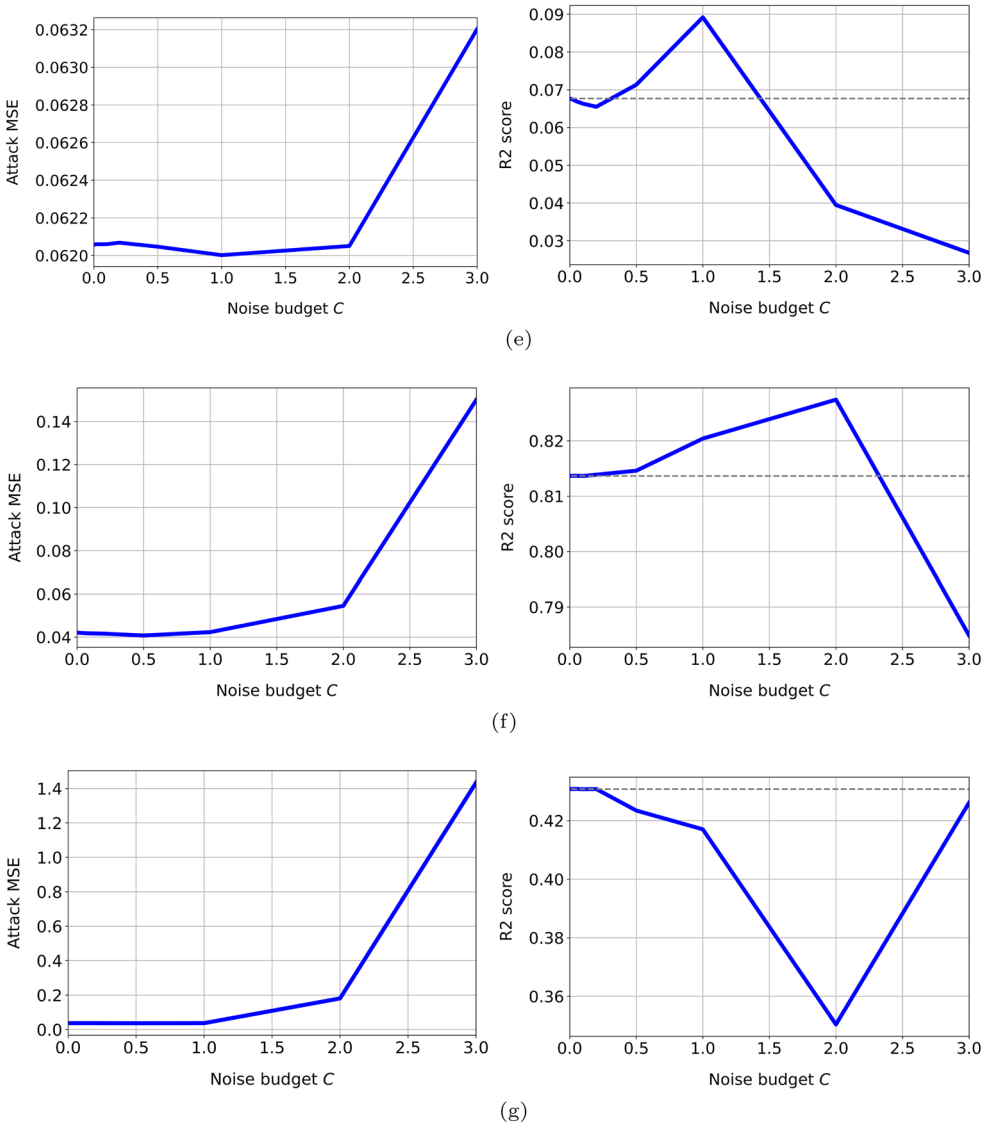
Fig. 6 Defense results against attribute inference attack (first column) and ridge regression results with defense methods (second column) for datasets with continuous private variable

compared methods, **Ours** becomes less efficient when the number of data is very large, as shown in Fig. 4.

6.2.2 Regression result

The R^2 score is a metric widely used to measure the performance of regression models. The higher the value is, the higher the prediction performance is and it takes a value of 1 when the model achieves a perfect prediction. The R^2 score of each model ($\lambda = 1$) is also shown in Table 2. The table also contains the regression results for **Baseline**. Except for three datasets, **Ours** consistently performed better than **FullColumn**. Even when **FullColumn** had a higher R^2 score, the R^2 score of **Ours** was more similar

Fig. 6 (continued)



to that of **Baseline**. The results show that using a gradient descent, it is difficult to converge to the exact solution using only a small number of iterations. Although in some cases, by chance, the approximate solution performs better than the actual solution, in most cases it does not. The difference between the R^2 score of **Baseline** and **Ours** is mainly due to the small number of iterations in Inv, and partly due to the approximation of CKKS scheme. The results for different values of λ are provided in Tables 3 to 6.

6.2.3 Defense against attribute inference attack

In this section we evaluate our adversarial perturbation method (**AP**) with one private variable. We compared **AP** with **Attriguard** proposed in [18]. Because **Attriguard** can only be applied to discrete private variables, the comparison was made only for four datasets, **Credit**, **Heart**, **Adult**, and **Census**. We considered the cross entropy with the target variable, which is mainly used for classification task, as the loss function $\mathcal{L}_{def}$ (**AP-CE**), but following **Attriguard**, the cross entropy with the prior distribution of the private variable was also considered as a loss function (**AP-KL**).

The first column of Fig. 5 plots the attacker's inference accuracy for the noise budget C. For **Credit** dataset, all three methods lowered the attacker's accuracy by similar values. For very low C values, **Attriguard** showed the best performance while **AP-KL** showed bad performance. For **Heart** dataset, only **AP-CE** defended well against the attack. For **Adult** dataset, the attacker's accuracy converged for all methods, but **Attriguard** recorded the lowest convergence value. For **Census** dataset, **AP-CE** was always better than **Attriguard**, and **AP-KL** showed the best defense at $C = 2.0$.

Although the utility loss due to the noise can be measured as the magnitude of the noise, it may be more desirable to measure how much it affects the actual performance of ridge regression. The second column of Fig. 5 plots the R^2 score of ridge regression for C. For all datasets, it was found that **AP-CE** maintained the R^2 score better than **Attriguard**. In particular, for **Credit** dataset, although **Attriguard** defended against the attack with a smaller noise, the performance of ridge regression was rather lower. The result seems to be because **AP-CE** searches a larger probability space for the noise than **Attriguard**, which finds only one noise per each value of private variable.

For continuous private variable, we used MSE as the loss function. The results for those variables are provided in Fig. 6. It is commonly shown that for all the datasets **AP** effectively increased the attacker's loss by a moderate sacrifice of the performance of ridge regression.

7 Conclusion

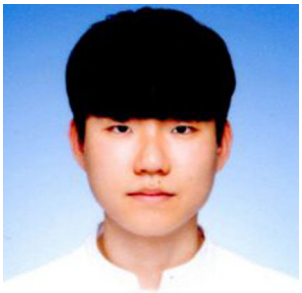
In this study, we proposed a privacy-preserving ridge regression algorithm with homomorphic encryption of multiple private variables. In addition, we suggested an adversarial perturbation method which can defend attribute inference attacks on the private variables. The experimental results showed that the computational time of our algorithm is faster in proportion to the number of variables than the gradient descent method. Moreover, compared to an existing defense method, our adversarial perturbation method can effectively prevent inference attacks while preserving the performance of the ridge regression. Our method can be developed more efficiently by using other plaintext packing strategies, or by designing more sophisticated SIMD operations. The extention of the proposed method to other machine learning algorithms needs to be further investigated.

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (NRF-2019R1A2C2002358) and in part by Institute of Information & communications Technology Planning & Evaluation (IITP) Grant funded by the Korean Government (MSIT) (No. 2022-0-00984).

References

1. Acharya MS, Armaan A, Antony AS (2019) A comparison of regression models for prediction of graduate admissions. In: 2019 International conference on computational intelligence in data science (ICCIDS). IEEE, pp 1–5
2. Aono Y, Hayashi T, Phong LT et al (2017) Input and output privacy-preserving linear regression. IEICE Trans Inf Syst 100(10):2339–2347
3. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 6(3):1–36
4. Byun J, Lee W, Lee J (2021) Parameter-free he-friendly logistic regression. Adv Neural Inf Process Syst 34:8457–8468
5. Cai Z, He Z, Guan X et al (2016) Collective data-sanitization for preventing sensitive information inference attacks in social networks. IEEE Trans Dependable Secure Comput 15(4):577–590
6. Cheon JH, Kim A, Kim M et al (2017) Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security. Springer, pp 409–437
7. De Spiegeleer J, Madan DB, Reyners S et al (2018) Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. Quantitative Finance 18(10):1635–1643
8. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
9. Feldman M, Friedler SA, Moeller J et al (2015) Certifying and removing disparate impact. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 259–268
10. Fredrikson M, Jha S, Ristenpart T (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp 1322–1333
11. Gai K, Qiu M, Zhao H (2017) Privacy-preserving data encryption strategy for big data in mobile cloud computing. IEEE Trans Big Data
12. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on theory of computing, pp 169–178
13. Gentry C, Halevi S, Smart NP (2012) Fully homomorphic encryption with polylog overhead. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 465–482
14. Guo L, Ying X, Wu X (2010) On attribute disclosure in randomization based privacy preserving data publishing. In: 2010 IEEE international conference on data mining workshops. IEEE, pp 466–473
15. Halevi S (2017) Homomorphic encryption. Tutorials on the foundations of cryptography. isc
16. Hardt M, Price E, Srebro N (2016) Equality of opportunity in supervised learning. In: Advances in neural information processing systems, pp 3315–3323
17. Janai J, Güney F, Behl A et al (2020) Computer vision for autonomous vehicles: problems, datasets and state of the art. Foundations and Trends®, in Computer Graphics and Vision 12(1–3):1–308
18. Jia J, Gong NZ (2018) Attriguard: a practical defense against attribute inference attacks via adversarial machine learning. In: 27th {USENIX} security symposium ({USENIX} security 18), pp 513–529
19. Kaissis GA, Makowski MR, Rückert D et al (2020) Secure, privacy-preserving and federated machine learning in medical imaging. Nature Mach Intell:1–7
20. Kosinski M, Stillwell D, Graepel T (2013) Private traits and attributes are predictable from digital records of human behavior. Proc National Acad Sci 110(15):5802–5805
21. Lee W, Ko H, Byun J et al (2021) Fair clustering with fair correspondence distribution. Inf Sci 581:155–178
22. Liu Q, Shen H, Sang Y (2014) A privacy-preserving data publishing method for multiple numerical sensitive attributes via clustering and multi-sensitive bucketization. In: 2014 Sixth international symposium on parallel architectures, algorithms and programming. ieee, pp 220–223

23. Lou Q, Feng B, Fox GC et al (2019) Glyph: fast and accurately training deep neural networks on encrypted data. arXiv:191107101
24. Ma L, Sun B (2020) Machine learning and ai in marketing–connecting computing power to human insights. Int J Res Mark 37(3):481–504
25. Morshed T, Alhadidi D, Mohammed N (2018) Parallel linear regression on encrypted data. In: 2018 16th Annual conference on privacy, security and trust (PST). IEEE, pp 1–5
26. Park S, Byun J, Lee J et al (2020) He-friendly algorithm for privacy-preserving svm training. IEEE Access 8:57,414–57,425
27. Park S, Byun J, Lee J (2022) Privacy-preserving fair learning of support vector machine with homomorphic encryption. In: Proceedings of the ACM Web Conference 2022, pp 3572–3583
28. Poursaeed O, Katsman I, Gao B et al (2018) Generative adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4422–4431
29. Qiu G, Gui X, Zhao Y (2020) Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking. IEEE Access 8:107,601–107,613
30. Sumathi M, Sangeetha S (2018) Enhanced elliptic curve cryptographic technique for protecting sensitive attributes in cloud storage. In: 2018 IEEE international conference on computational intelligence and computing research (ICCIC). IEEE, pp 1–5
31. Voigt P, Von Dem Bussche A (2017) The Eu General Data Protection Regulation (gdpr). A Practical Guide 1st Edn. Springer International Publishing, Cham



Junyoung Byun received the B.S. degree in industrial engineering from Seoul National University, Seoul, South Korea, in 2017. He is currently a Graduate Student with the Department of Industrial Engineering, Seoul National University. His research interests include privacy-preserving machine learning and fair machine learning.

Saerom Park received a B.S. and Ph.D. in industrial engineering from Seoul National University in 2013 and 2018. She is currently an Assistant Professor in the Department of Convergence Security Engineering, Sungshin Women's University in Seoul, South Korea. Her research focuses on trustworthy machine learning. Her research focus is on secure and stable machine learning. This includes stability analysis, robust training against adversaries, fair machine learning, and privacy-preserving machine learning.

Yujin Choi received B.S in industrial engineering and mathematics from Yonsei University in 2021. She is currently a Ph.D. candidate in industrial engineering at Seoul National University. Her research interests are privacy and ai generalization, including privacy-preserving machine learning and model analyzing via weight space and loss landscape.

Jaewook Lee received the B.S. degree in mathematics from Seoul National University, Seoul, South Korea, in 1993, and the Ph.D. degree in applied mathematics from Cornell University, in 1999. He is currently a Professor with the Department of Industrial Engineering, Seoul National University. His research interests include machine learning, neural networks, global optimization, and their applications to data mining and financial engineering.