Secure Computation of Gaussian Process Regression for Data Analysis

Takayuki Nakachi
Information Technology Center
University of the Ryukyus
Senbaru Nishihara Okinawa, Japan
takayuki.nakachi@ieee.org

Yitu Wang
NTT Network Innovation Laboratories
Nippon Telegraph and Telephone Corporation
Yokosuka-city Kanagawa, Japan
yitu.wang.dp@hco.ntt.co.jp

***Abstract*—We propose Gaussian process regression (GPR) for encrypted data generated based on random unitary transformation. Edge cloud computing is spreading into many application fields, including data analysis services. However, new privacy concerns are being raised regarding things such as data leakage from unreliable providers and accidents. Therefore, we examine a secure GPR method that incorporates privacy protection. We prove that our GPR for encrypted data performs the same as GPR for non-encrypted data. Finally, we determine the effectiveness of our method by experimenting with diabetes data from the medical analysis field and synthetic data.**

***Index Terms*—Gaussian process, random unitary transformation, machine learning**

I. Introduction

Edge/cloud computing has rapidly become widespread in fields such as big data analytics. However, edge/cloud computing depends on the reliability of service providers, so privacy concerns have arisen in response to unreliability and the unauthorized use or loss of data resulting from accidents [1].

Secure computation in the encrypted domain has been studied as a response to these concerns. Most studies on encrypted data processing have used homomorphic encryption and secure multiparty computation [2] [3]. However, these approaches incur high computation complexity, which causes heavy loads on encrypting sites and large cipher text sizes. Cancelable biometrics [4] are being studied as methods that realize low complexity processing in the encrypted domain. Random projection (RP) is a method that projects an input signal into a lower dimensional sub-space using a random matrix generated from random numbers. BioHashing [5] is an encryption based on RP that transforms the input data into a binary string called the "hash code". Note that these encryption schemes are irreversible. Such irreversibility is preferable for security, but it is difficult to guarantee deterministically that analysis of the encrypted data does not degrade performance.

We use the random unitary transform [6] in this study because it offers secure but practical computation in big data analytics. The random unitary transform has much lower computation complexity and a smaller cipher text size than either homomorphic encryption or secure multiparty computation. Moreover, the random unitary transform guarantees the reversibility of the transform. The reversibility guarantees deterministically that analysis of the encrypted data does not degrade performance. Based on the random unitary transform, we have proposed a secure sparse coding method for image compression, pattern recognition, and data analysis [7]- [10].

The Gaussian process (GP) based on Bayesian nonparametrics has been attracting attention in the machine learning field [11]. GPs can encode domain and expert knowledge into kernel functions from model linear to non-linear data. Highly accurate estimation is possible by optimizing the hyperparameters of kernel functions based on the Bayes' theorem. In addition, it can output prediction uncertainty based on the amount of data used for learning. A neural network with an infinite number of units in a single hidden layer can be represented by a GP [12]. The neural network GP (NNGP) has been reported as equivalent to deep learning that corresponds to a multi-layer neural network [13].

We propose Gaussian process regression (GPR) for the encrypted data generated by the random unitary transform. We prove that our GPR for encrypted data performs the same as GPR for non-encrypted data. Simulation results showed that it is possible to solve a regression problem while keeping the input data secure when we made predictions for clinical data of diabetes without deteriorating the estimation performance compared with GPR for non-encrypted data.

The organization of this paper is as follows. Section II overviews GPR. Section III proposes a secure algorithm for GPR. Section IV explains simulation results. Section V gives conclusions and future work.

II. Gaussian Process Regression

In this section, we overview the GP and its regression.

A. Gaussian Process

We consider a regression problem in which the input is $\boldsymbol{x} \in \mathbb{R}^D$ (a row vector) and the output is $y \in \mathbb{R}$. Let $\mathcal{D}_{train} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ be a pair of N training data, and define $\boldsymbol{X}$ and $\boldsymbol{Y}$ by the following equation.

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}, \quad \boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}. \tag{1}$$

Given $\boldsymbol{X}$ and noisy output observation

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \epsilon, \tag{2}$$

where ϵ is an i.i.d. Gaussian noise with zero mean and σ^2 variance accounting for the measurement/modeling errors, GP seeks to infer the latent function $f(\boldsymbol{X})$. It is expressed by

$$f(\boldsymbol{X}) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})), \tag{3}$$

which is completely defined by the mean function (usually set to zero without loss of generality) and the so-called kernel function $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$. The kernel function is the symmetric and positive semi-definite covariance matrix with $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The radial basis function (RBF) kernel is often used in the GP. It is defined by the following equation.

$$\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_1 \ exp\left(-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{\theta_2}\right) + \sigma^2 \delta(i,j), \tag{4}$$

where $\delta(i,j)$ is a function that returns 1 when $i = j$ and 0 otherwise. The hyperparameters, including the parameters in $\boldsymbol{K}$ and σ, can be well trained by optimizing the negative log marginal likelihood

$$\min_{\mathbf{K},\sigma} \boldsymbol{Y}^T (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \mathbf{I})^{-1} \boldsymbol{Y} + \log_2 |\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \mathbf{I}|, \tag{5}$$

which can be solved by the efficient gradient descent algorithm via the partial derivatives of the marginal likelihood regarding the hyperparameters [14].

B. Gaussian Process Regression

Next, we consider solving the regression problem based on the GP to obtain the prediction of target value $y^* \in \mathbb{R}$ for a new input $\boldsymbol{x}^* \in \mathbb{R}^D$. We define a pair of datasets $\mathcal{D}_{test} = \{\boldsymbol{X}^*, \boldsymbol{Y}^*\}$, where $\boldsymbol{X}^*$ and $\boldsymbol{Y}^*$ are given by

$$\boldsymbol{X}^* = \begin{bmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \\ \vdots \\ \boldsymbol{x}_M^* \end{bmatrix}, \quad \boldsymbol{Y}^* = \begin{bmatrix} y_1^* \\ y_2^* \\ \vdots \\ y_M^* \end{bmatrix}, \tag{6}$$

where M is the number of test datasets. Thus, the joint prior distribution of $\boldsymbol{Y}$ with $f(\boldsymbol{X}^*)$ is obtained based on

$$\begin{bmatrix} \boldsymbol{Y} \\ f(\boldsymbol{X}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \mathbf{I} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}^*) \\ \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) \end{bmatrix} \right), \tag{7}$$

where $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}^*)$ denotes the covariance between the N training points and the M testing point. By conditioning the joint Gaussian prior distribution on $\boldsymbol{Y}$, the posterior distribution of $f(\boldsymbol{X}^*)$ can be analytically derived as

$$p(f(\boldsymbol{X}^*)|(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{X}^*)) \sim \mathcal{N}(f(\boldsymbol{X}^*), \sigma^2(\boldsymbol{X}^*)), \tag{8}$$

where the prediction mean and variance are respectively given as

$$\begin{aligned} f(\boldsymbol{X}^*) &= \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X})^T [\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{Y} \\ \sigma^2(\boldsymbol{X}^*) &= \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*) - \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X})^T [\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \\ &\quad + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}^*). \end{aligned} \tag{9}$$

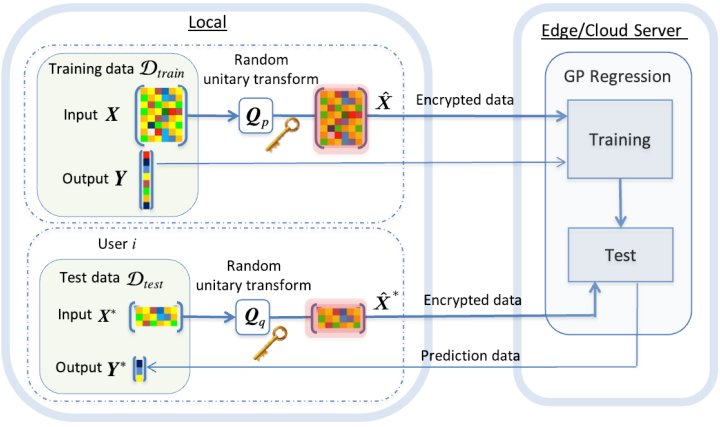


Fig. 1. System configuration of secure GP regression.

III. Secure Computation of Gaussian Process Regression

In this section, we propose secure GPR that allows computation in the encrypted domain.

A. System Configuration

Figure 1 shows a system configuration of secure GP regression. At the local site, N pairs of training data set $\mathcal{D}_{train} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ are prepared. First, the input $\boldsymbol{X}$ for training is transformed into the encrypted input $\hat{\boldsymbol{X}}$ by using the random unitary matrix $\boldsymbol{Q_p}$ generated with a private key p. Then, the encrypted input $\hat{\boldsymbol{X}}$ and the output $\boldsymbol{Y}$ are transferred to the edge/cloud. At the edge/cloud, the kernel function and its hyperparameters θ are estimated using $\hat{\boldsymbol{X}}$ and $\boldsymbol{Y}$.

Next, the encrypted input $\hat{\boldsymbol{X}}^*$ for testing is generated using the random unitary matrix $\boldsymbol{Q_q}$ with a private key q. Then, the encrypted input $\hat{\boldsymbol{X}}^*$ is transferred to the edge/cloud. At the edge/cloud site, the kernel function is calculated using $\hat{\boldsymbol{X}}^*$ and the pre-delivered $\hat{\boldsymbol{X}}$ and $\boldsymbol{Y}$ and the mean value $f(\hat{\boldsymbol{X}}^*)$ and the variance $\sigma^2(\hat{\boldsymbol{X}}^*)$ are estimated. Here, if the keys of random unitary transformation used for training and testing data generation are the same ($p = q$), the mean value and variance by our secure GPR are the same as the values estimated by GPR for the non-encrypted signals.

As described above, the mean value $f(\hat{\boldsymbol{X}}^*)$ and the variance $\sigma^2(\hat{\boldsymbol{X}}^*)$ can be calculated without decrypting the input $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{X}}^*$. It is also possible to protect the data even if it leaks from the edge/cloud because of unreliability or an accident.

B. Secure Computation

In our secure GP regression, the random unitary transforms $\boldsymbol{Q}_p$ and $\boldsymbol{Q}_q \in \mathbb{R}^{D \times D}$ with a private key p and a private key q are used to encrypt $\boldsymbol{X}$ for training and $\boldsymbol{X}^*$ for testing, respectively:

$$\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{Q}_p = \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\boldsymbol{x}}_2 \\ \vdots \\ \hat{\boldsymbol{x}}_N \end{bmatrix}, \quad \hat{\boldsymbol{X}}^* = \boldsymbol{X}^*\boldsymbol{Q}_q = \begin{bmatrix} \hat{\boldsymbol{x}}_1^* \\ \hat{\boldsymbol{x}}_2^* \\ \vdots \\ \hat{\boldsymbol{x}}_M^* \end{bmatrix} \tag{10}$$

The D elements of each sample $\boldsymbol{x}_i$ and $\boldsymbol{x}_i^*(i = 1, 2, \cdots, N)$ are encrypted. Given encrypted input $\hat{\boldsymbol{X}}$ and noisy output observation

$$\boldsymbol{Y} = f(\hat{\boldsymbol{X}}) + \epsilon, \tag{11}$$

GP seeks to infer the latent function $f(\hat{\boldsymbol{X}})$. As in the case where the input data is not encrypted, the joint prior distribution of $\boldsymbol{Y}$ and $f(\hat{\boldsymbol{X}}^*)$ are given by the following equation:

$$\begin{bmatrix} \boldsymbol{Y} \\ f(\hat{\boldsymbol{X}}^*) \end{bmatrix} \sim \mathcal{N}\Bigg(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}}) + \sigma^2 \mathbf{I} & \boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}}^*) \\ \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}) & \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}^*) \end{bmatrix} \Bigg). \tag{12}$$

By conditioning the joint Gaussian prior distribution on $\boldsymbol{Y}$, the posterior distribution of $f(\hat{\boldsymbol{X}}^*)$ can be analytically derived as

$$p(f(\hat{\boldsymbol{X}}^*)|(\hat{\boldsymbol{X}}, \boldsymbol{Y}, \hat{\boldsymbol{X}}^*)) \sim \mathcal{N}(f(\hat{\boldsymbol{X}}^*), \sigma^2(\hat{\boldsymbol{X}}^*)), \tag{13}$$

where the prediction mean and variance are given by

$$\begin{aligned} f(\hat{\boldsymbol{X}}^*) &= \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}})^T [\boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}}) + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{Y} \\ \sigma^2(\hat{\boldsymbol{X}}^*) &= \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}^*) - \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}})^T [\boldsymbol{K}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}}) \\ &\quad + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{K}(\hat{\boldsymbol{X}}^*, \hat{\boldsymbol{X}}^*). \end{aligned} \tag{14}$$

Equation (14) shows that the prediction mean and variance are determined by the kernel function of encrypted input $\hat{\boldsymbol{X}}$, $\hat{\boldsymbol{X}}^*$, and output $\boldsymbol{Y}$.

C. Encryption Based on Random Unitary Transform

Secure sparse coding methods based on random unitary transformation have been explored in previous studies [7]-[10]. This paper was inspired by those studies. An input $\boldsymbol{x_i} \in \mathbb{R}^D$ is encrypted by a unitary matrix $\boldsymbol{Q}_p \in \mathbb{C}^{D \times D}$ with a private key p as follows:

$$\hat{\boldsymbol{x_i}} = \boldsymbol{x_i} \boldsymbol{Q}_p, \tag{15}$$

where $\hat{\boldsymbol{x_i}}$ is an encrypted input signal. Note that the unitary matrix $\boldsymbol{Q}_p$ satisfies

$$\boldsymbol{Q}_p^H \boldsymbol{Q}_p = \boldsymbol{I}, \tag{16}$$

where $[\cdot]^H$ stands for the Hermitian transpose operation and $\boldsymbol{I}$ stands for the identity matrix. In addition to the unitary matrix, $\boldsymbol{Q}_p$ must have randomness for generating the encrypted data. Gram-Schmidt orthogonalization is a typical method of generating $\boldsymbol{Q}_p$. The encrypted data has the following properties:

· Property 1: L2 norm isometry

$$||\boldsymbol{x_i}||_2^2 = ||\hat{\boldsymbol{x_i}}||_2^2$$

· Property 2: Conservation of Euclidean distances

$$||\boldsymbol{x_i} - \boldsymbol{x_j}||_2^2 = ||\hat{\boldsymbol{x_i}} - \hat{\boldsymbol{x_j}}||_2^2$$

· Property 3: Conservation of inner products

$$\boldsymbol{x}_i^H \boldsymbol{x}_j = \hat{\boldsymbol{x}}_i^H \hat{\boldsymbol{x}}_j$$

We consider applying the random unitary transform to a kernel function. When using an RBF kernel for the encrypted input $\hat{\boldsymbol{X}}$ for training, the following relation is satisfied:

$$\begin{aligned} K(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j) &= \theta_1 \ exp\left(-\frac{|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j|^2}{\theta_2} \right) + \sigma^2 \delta(i, j) \\ &= \theta_1 \ exp\left(-\frac{|(\boldsymbol{x}_i - \boldsymbol{x}_j)\boldsymbol{Q}_p|^2}{\theta_2} \right) + \sigma^2 \delta(i, j) \\ &= \theta_1 \ exp\left(-\frac{|\boldsymbol{x}_i - \boldsymbol{x}_j|^2}{\theta_2} \right) + \sigma^2 \delta(i, j) \\ &= K(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{17}$$

Aside from the RBF kernel, most typical kernels (i.e., the rational quadratic kernel, the Matérn kernel, etc.) also satisfy $K(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. However, some kernels do not hold. Similarly, the following relation holds for the encrypted input $\hat{\boldsymbol{X}}^*$ for testing:

$$K(\hat{\boldsymbol{x}}_i^*, \hat{\boldsymbol{x}}_j^*) = K(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*). \tag{18}$$

However, the kernel function between the encrypted input $\hat{\boldsymbol{X}}$ for training and the encrypted input $\hat{\boldsymbol{X}}^*$ for testing do not match the kernel function for the non-encrypted data. This is because the encrypted training signal $\hat{\boldsymbol{X}}$ and the encrypted test signal $\hat{\boldsymbol{X}}^*$ are encrypted using different random unitary transforms $\boldsymbol{Q}_p$ and $\boldsymbol{Q}_q$.

$$\begin{aligned} K(\hat{\boldsymbol{x}}_i^*, \hat{\boldsymbol{x}}_j) &= \theta_1 \ exp\left(-\frac{|\hat{\boldsymbol{x}}_i^* - \hat{\boldsymbol{x}}_j|^2}{\theta_2} \right) + \sigma^2 \delta(i, j) \\ &= \theta_1 \ exp\left(-\frac{|\boldsymbol{x}_i \boldsymbol{Q}_q - \boldsymbol{x}_j \boldsymbol{Q}_p|^2}{\theta_2} \right) + \sigma^2 \delta(i, j) \\ &\neq K(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{19}$$

When the random unitary transform is generated using the same private key ($p = q$) for training and testing, $K(\hat{\boldsymbol{x}}_i^*, \hat{\boldsymbol{x}}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is satisfied. Therefore, from Eqs. (9) and and (14), when using the same private key ($p = q$), the prediction mean and variance estimated by secure GPR are equal to those by GPR for non-encrypted data:

$$f(\hat{\boldsymbol{X}}^*) = f(\boldsymbol{X}^*) \tag{20}$$

$$\sigma^2(\hat{\boldsymbol{X}}^*) = \sigma^2(\boldsymbol{X}^*). \tag{21}$$

We can use the keys p and q to control privacy. For legitimate users, we distribute the same key ($p = q$).

IV. Numerical Demonstrations

We performed the following experiments using diabetes data from the medical analysis field and synthetic data to investigate the effectiveness of our scheme.

A. Simulation Conditions

The diabetes data includes quantitative measures for 442 diabetes patients on 10 baseline variables such as age, gender, BMI, and disease progression one year after baseline [15] [16]. Our secure GPR (secGPR) predicted a measure of disease progression from the 10 baseline variables. The input $\boldsymbol{X}$ was

TABLE I
MEAN SQUARE ERROR (MSE) AND PEARSON PRODUCT-MOMENT CORRELATION COEFFICIENT (PPMCC) BETWEEN DISEASE PROGRESSION $\boldsymbol{Y}^*$ ESTIMATED BY GPR AND $\boldsymbol{S}^*$ ESTIMATED BY SECGPR.

(a) RBF kernel

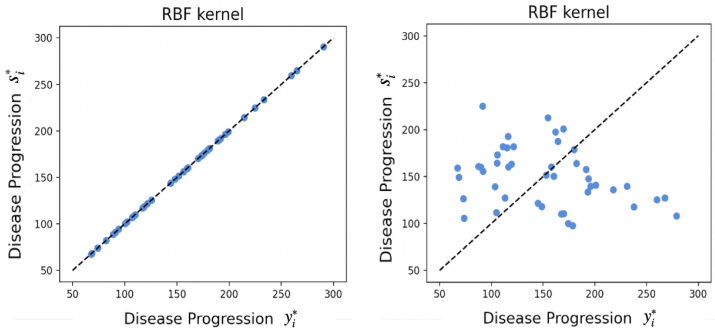
Private key	$p = q$	$p \neq q$
MSE	2.16×10^{-24}	8044
PPMCC	1.0	-0.30

(b) Rational quadratic kernel

Private key	$p = q$	$p \neq q$
MSE	2.10×10^{-18}	8040
PPMCC	1.0	-0.30

(c) Matérn kernel

Private key	$p = q$	$p \neq q$
MSE	1.16×10^{-24}	7611
PPMCC	1.0	-0.19



(a) $p = q$ (b) $p \neq q$

Fig. 2. Relationship between disease progression y_i^* estimated by GPR and s_i^* estimated by secGPR when using RBF kernel.

set as the test data of 10 baseline variables (i.e., $D = 10$), and the output $\boldsymbol{Y}$ was set as the disease progression. Data from 353 patients was used for learning ($N = 397$) and data from 89 patients was used for testing ($M = 45$).

The input $\boldsymbol{X}$ was encrypted using the random unitary transform $\boldsymbol{Q}_p$ that was generated by the following equation.

$$\boldsymbol{Q}_p = \boldsymbol{H}_{PR}\boldsymbol{G}_p, \tag{22}$$

where $\boldsymbol{G}_p$ is generated by the Gram-Schmidt orthogonalization. $\boldsymbol{H}_{PR}$ is a permutation matrix of $D \times D$ dimension that randomly replaces each element of the input signal $\boldsymbol{x} \in \mathbb{R}^D$. An example of a matrix when the number of dimensions of the input data is D = 4 is shown below.

$$\boldsymbol{H}_{PR} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{23}$$

Since both $\boldsymbol{H}_{PR}$ and $\boldsymbol{G}_p$ have a unitary matrix, $\boldsymbol{Q}_p$ satisfies the condition of the random unitary matrix:

$$\begin{aligned} \boldsymbol{Q}_p^H \boldsymbol{Q}_p &= (\boldsymbol{H}_{PR}\boldsymbol{G}_p)^H(\boldsymbol{H}_{PR}\boldsymbol{G}_p) = (\boldsymbol{G}_P^H \boldsymbol{H}_{PR}^H)(\boldsymbol{H}_{PR}\boldsymbol{G}_p) \\ &= \boldsymbol{I}. \end{aligned} \tag{24}$$

B. Estimation Accuracy

The estimation accuracy of secGPR was compared with that of GPR for non-encrypted data based on the similarity between disease progression $\boldsymbol{Y}^*$ estimated by GPR and $\boldsymbol{S}^*$ estimated by secGPR. The MSE and PPMCC were used as similarity indexes:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i^* - s_i^*)^2 \tag{25}$$

$$PPMCC = \frac{\sum_{i=1}^{N}(y_i^* - \overline{y}^*)(s_i^* - \overline{s}^*)}{\sqrt{\sum_{i=1}^{N}(y_i^* - \overline{y}^*)^2}\sqrt{\sum_{i=1}^{N}(s_i^* - \overline{s}^*)^2}} \tag{26}$$

where $y_i^* \in \boldsymbol{Y}^*$ is disease progression estimated by GPR and $s_i^* \in \boldsymbol{S}^*$ is disease progression estimated by secGPR. $\overline{y}^*$ and $\overline{s}^*$ are the respective average values. When the PPMCC is close to 1, there is a strong correlation, and secGPR can estimate the same values as GPR. We evaluated three kernels (RBF, rational quadratic, and Matérn). Two patterns were verified for the random unitary transform for encrypting the training and testing data: one when the same private key ($p = q$) was used and the other when different private keys ($p \neq q$) were used.

Table I shows the MSE and PPMCC between $\boldsymbol{Y}^*$ and $\boldsymbol{S}^*$. When the private keys are the same ($p = q$), secGPR has small a MSE and the PPMCC = 1, meaning the estimation performance of secGPR does not deteriorate compared with GPR. However, when the private keys are different, the MSE is large and the PPMCC is small. Figure 2 shows the relationship between the disease progression y_i^* and s_i^* when using the RBF kernel. It shows that the output $\boldsymbol{Y}^*$ estimated by secGPR is almost the same as that of GPR when $p = q$. Similar results were obtained when used with other two kernels. Table I and Fig. 2 show that security can be controlled using $\boldsymbol{Q}_p$ for training and $\boldsymbol{Q}_q$ for testing.

C. Security Strength

1) Key Space: We evaluated the safety of the encrypted input $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{Q}_p$ in terms of the key space of $\boldsymbol{Q}_p$. The key space is calculated assuming a case of restoration by brute force attack. We consider a case of the random unitary transform being generated by Eq. (22) (i.e., $\boldsymbol{Q}_p = \boldsymbol{H}_{PR}\boldsymbol{G}_p$).

First, elements of the unitary transform are limited to real numbers (orthogonal matrix) for $\boldsymbol{G}_p$. The degree of freedom is D^2, which is equal to the number of matrix elements. However, the unitary matrix is subject to the following conditions:

1) The column vectors of the unitary matrix are orthogonal to each other. The number of conditional expressions imposed is ${}_D\mathrm{C}_2 = D(D-1)/2$ (number of combinations that select two from D column vectors) from the condition.
2) The norm of each column vector = 1. The number of conditional expressions imposed is D from the condition.

TABLE II
ABSOLUTE VALUE OF PPMCC FOR DIABETES INPUT $\boldsymbol{X}$ ($D = 10$).

	Ave	Max	Min
PPMCC	0.122	0.526	8.51×10^{-6}

Therefore, the degree of freedom is $D^2 - [D(D-1)/2 + D] = D(D-1)/2$ for the random unitary transform $\boldsymbol{G}_p$. Assuming each element is represented by an 8-bit fixed point number, the size of the key space is $8^{D(D-1)/2}$. Next, the combination pattern is $D!$ for the permutation matrix $\boldsymbol{H}_{PR}$. Therefore, the size of the key space of the random unitary transform $\boldsymbol{Q}_p (= \boldsymbol{H}_{PR}\boldsymbol{G}_p)$ is $8^{D(D-1)/2} \times D!$. Compared with the key space used in the Advanced Encryption Standard, the key space is wider than the 128-bit case and narrower than the 256-bit space when the number of elements used in this simulation is $D = 10$. If D is 13 or more, it will be wider than the 256-bit key space.

2) Irreversibility: We investigated the security strength of the encrypted input $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{Q}_{\boldsymbol{p}}$ via simulations. The security strength was evaluated based on the absolute value of the PPMCC between the original input $\boldsymbol{X}$ and the decrypted input $\hat{\boldsymbol{X}}\boldsymbol{Q}_q^H$ that was attacked by the illegitimate users (i.e., $p \neq q$). Generally, the two samples can be regarded as uncorrelated when the absolute value of the PPMCC between two data samples is less than 0.2. We assumed 100 legitimate users (i.e., generated 100 kinds of random unitary matrices $\boldsymbol{Q}_p$). Then the illegitimate users tried to decrypt each piece of encrypted input using 100 kinds of random unitary matrices $\boldsymbol{Q}_q$ that differed from the ones used in encryption. We tested 10,000 matrix combination patterns.

Table II shows the average, maximum, and minimum values of the absolute PPMCC for diabetes data. The absolute PPMCC is small on average, but the maximum value is relatively large. The key space is not considered to be large enough when the dimension $D = 10$. Figure 3 shows the absolute value of the PPMCC for synthetic input $\boldsymbol{X}$. The input data $\boldsymbol{X}$ (in which each element follows a normal Gaussian distribution) was generated with different dimensions ($D = 5, 10, 20, 50, 100$). The absolute value of the PPMCC clearly decreases as the dimension D increases. If D is greater than around 30, both the average value and the maximum value are less than 0.2. From the perspective of irreversibility, security is stronger when the dimension D is higher. Future study will need to consider security when the D value is small.

V. CONCLUSIONS AND FUTURE WORK

We proposed GPR for encrypted data generated based on random unitary transformation. Our practical GPR scheme enables computation on encrypted data. We proved that our GPR for encrypted data has exactly the same regression performance as the non-encrypted variants of the GPR scheme. We determined its effectiveness using diabetes data from the medical analysis field and synthetic data.

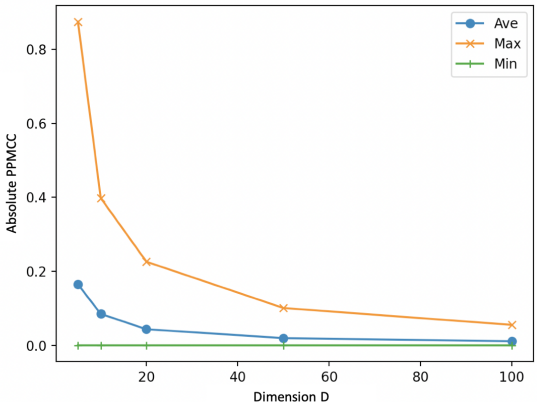


Fig. 3. Absolute value of PPMCC for synthetic input $\boldsymbol{X}$.

We will research further on the security strength when the dimension of the input data is low. In addition, we will consider applications for high-dimensional cases such as image processing.

REFERENCES

[1] C. T. Huang, L. Huang, Z. Qin, H. Yuan, L. Zhou, V. Varadharajan, and C-C. J. Kuo, "Survey on securing data storage in the cloud," APSIPA Transactions on Signal and Information Processing, vol. 3, e7, 2014.

[2] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 82–105, January 2013.

[3] A. Chatterjee and I. Sengupta, "Sorting of fully homomorphic encrypted cloud data: Can partitioning be effective?," IEEE Transactions on Services Computing, vol. 13, no. 3, pp. 545–558, May–June 2020.

[4] V. Patel, N. Ratha, and R. Chellappa, "Cancelable biometrics: A review," IEEE Signal Process. Mag., vol. 32, no. 5, pp. 54–65, 2015.

[5] A. B. J. Teoh, A. Goh, and D. C. L. Ngo, "Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs," IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 1892–1901, 2006.

[6] I. Nakamura, Y. Tonomura, and H. Kiya, "Unitary transform-based template protection and its application to l2-norm minimization problems," IEICE Trans. Inf. & Syst., vol. E99-D, no. 1, p. 60–68, 2016.

[7] T. Nakachi, Y. Bandoh, and H. Kiya, "Secure overcomplete dictionary learning for sparse representation," IEICE Transactions on Information and Systems, vol. E103.D, no. 1, pp. 50–58, January 2020.

[8] T. Nakachi and H. Kiya, "Secure OMP computation maintaining sparse representations and its application to EtC systems," IEICE Transactions on Information and Systems, vol. E103.D, no. 9, pp. 1988–1997, 2020.

[9] Y. Wang and T. Nakachi, "A privacy-preserving learning framework for face recognition in edge and cloud networks," IEEE Access, vol. 8, pp. 136056–136070, 2020.

[10] Y. Bandoh, T. Nakachi, and H. Kiya, "Distributed secure sparse modeling based on random unitary transform," IEEE Access, vol. 8, pp. 211762–211772, 2020.

[11] H. Liu, Y. S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 1, no. 1, pp. 1–19, January 2020.

[12] R. M. Neal, "Bayesian learning for neural networks," Springer-Verlag New York, Inc. 1996.

[13] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," 2017, https://arxiv.org/abs/1711.00165.

[14] C. Rasmussen and C. Williams, "Gaussian processes for machine learning," *MIT Press*, 2006.

[15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," Annals of Statistics, vol. 32, no. 2, pp. 407–499, 2004.

[16] scikit-learn, https://scikit-learn.org/stable/index.html.