APSIPA Transactions on Signal and Information Processing, 2024, 13, e26

Original Paper

Speech Privacy-preserving Methods Using Secret Key for Convolutional Neural Network Models and Their Robustness Evaluation

Niwa Shoko, Sayaka Shiota* and Hitoshi Kiya

Tokyo Metropolitan University, 6-6, Asahigaoka, Hino, Tokyo, Japan

ABSTRACT

In this paper, we propose privacy-preserving methods with a secret key for convolutional neural network (CNN)-based models in speech processing tasks. In environments where untrusted third parties, like cloud servers, provide CNN-based systems, ensuring the privacy of speech queries becomes essential. This paper proposes encryption methods for speech queries using secret keys and a model structure that allows for encrypted queries to be accepted without decryption. Our approach introduces three types of secret keys: Shuffling, Flipping, and random orthogonal matrix (ROM). In experiments, we demonstrate that when the proposed methods are used with the correct key, identification performance did not degrade. Conversely, when an incorrect key is used, the performance significantly decreased. Particularly, with the use of ROM, we show that even with a relatively small key space, high privacy-preserving performance can be maintained many speech processing tasks. Furthermore, we also demonstrate the difficulty of recovering original speech from encrypted queries in various robustness evaluations.

*Corresponding author: Sayaka Shiota, sayaka@tmu.ac.jp

Received 25 March 2024; revised 21 July 2024; accepted 28 July 2024
ISSN 2048-7703; DOI 10.1561/116.20240020

Keywords: Privacy-preserving, waveform encryption, spectrogram encryption, secret key

1 Introduction

Speech data usually includes personal information such as age, gender, language, and speaking content. To exploit such information, CNN models for tasks such as automatic speech recognition, speech synthesis, and speaker verification have been actively studied [26, 28]. In recent years, CNN models and speech data are increasingly uploaded to or stored on cloud servers via the Internet, and CNN models are run on cloud servers. However, since cloud services are managed by external providers, various threats such as data leakage due to malicious attacks from outside or inside are a concern [22, 20]. When using CNN models on a cloud service, it is necessary to provide a trained model and query data to the cloud service. Therefore, when cloud services are insecure, models and queries face threats. To prevent such risks, it is important to preserve privacy before sending data to insecure services. The issue of privacy has been gradually gaining attention as the latest topic in the research field of speech processing [24, 5, 6, 29, 23]. However, most of the existing methods for preserving the privacy of speech focus on concealing information about the speaker of the speech [24], and little is mentioned about concealing the content of speech [27]. There are also problems such as model performance degradation when the existing methods are in use.

In the research field of image processing, many privacy-preserving methods have been proposed for CNN-based systems [9, 12]. These methods propose a framework wherein models can process encrypted images using a secret key without the need for decryption, thereby protecting the visual information. Inspired by the research, we have proposed a simple privacy-preserving method using a secret key for speech data [16]. This approach regarded the privacy-preserving of speech data as protection of the auditory information. Thus, this research on the privacy-preserving of speech data aimed to control the performance of speech processing systems by using a secret key. This paper expands the robustness of this initial research and presents the robustness evaluation for speech privacy-preserving. The privacy-preserving methods used in this paper, e.g. Shuffling, Flipping, ROM are common in biometric template protection [10, 11] and privacy-preserving image classification areas. Our contributions are not only to apply them to speech areas but to also propose the method that can avoid the performance degradation of models. The proposed method encrypts the speech queries with secret keys and uses a model structure that allows encrypted speech queries to be accepted without decryption. To realize this framework, we assume that the first layer of the CNN model is a convolutional layer, and that the kernel size and stride size of the first convolutional layer are equal. For model encryption, the kernel of the first convolutional layer of the model is encrypted using a matrix corresponding to the matrix used as the secret key. This operation allows encrypted speech data to be input directly into the model without decryption, as it cancels out the effect of encryption on the input speech data. Our approach introduces three types of secret keys: permutation matrix, sign inversion, and random orthogonal matrix. To validate the advantages of our approach, which include task independence and the absence of the need to retrain the model as long as certain conditions are met, we conducted performance evaluations of our privacy-preserving methods using three tasks: automatic speaker verification (ASV), automatic speech recognition (ASR), and acoustic scene classification (ASC) task. The experimental results show that CNN models can be used with the same accuracy as before encryption, when speech encrypted with the correct secret key is input to the model encrypted with the correct secret key. It is also shown that the accuracies of the CNN models are significantly reduced when the input speech is encrypted with an incorrect secret key. In particular, it is shown that using a random orthogonal matrix as a secret key can preserve speech privacy while maintaining a large key space, even when the block size is small. Furthermore, experiments done to evaluate robustness show that when audio encrypted with an incorrect secret key is input to a model encrypted with the correct secret key, the performance of the model decreases steadily for larger block sizes, and stable privacy-preserving performance is obtained. In addition, it is shown that speech data encrypted with the proposed methods cannot be reconstructed unless the correct secret key is used.[1]

The following is the structure of the paper. In Section 2, we describe the privacy-preserving scenario that we assume. Section 3 gives the details on the proposed methods and in Section 4, we show the experimental results. In Section 5, we conclude the study and describe our future work.

2 Privacy-preserving scenario

The scenario assumed for the privacy-preserving frameworks in this paper is illustrated in Figure 1. Figure 1 consists of the model owner, the authorized user, and the external provider. The external provider is assumed to be untrusted, while the model owner and the authorized user are considered secure. First, the model owner trains a CNN model to process plain speech data, such as spectrograms and waveforms, within a secure environment. The trained model is encrypted with a secret key. Subsequently, the model owner

[1]The code used to generate the secret key, encrypt the model, and encrypt query speech data is available at https://github.com/kiyalab-tmu/SecretKeyVoicePrivacyPreserving-CNN

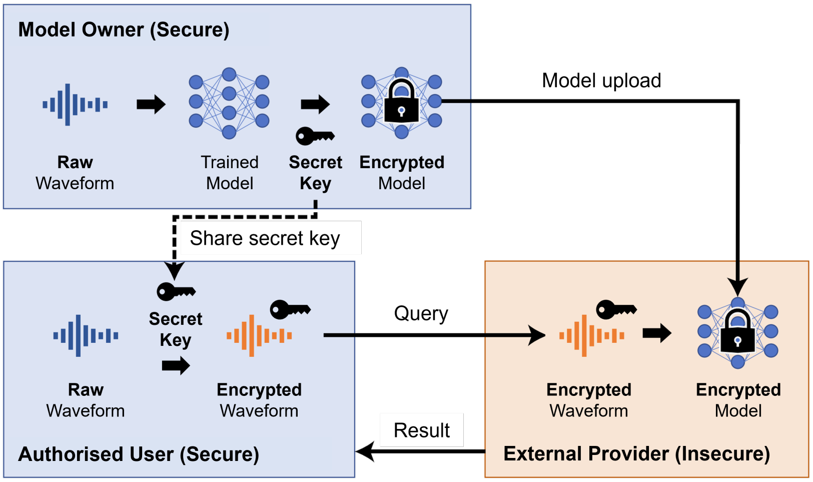


Figure 1: Privacy-preserving scenario

provides the encrypted model to an external provider, such as a cloud service, and shares the secret key used for the model encryption with an authorized user. It is assumed that the external provider, being managed by a third party, is not within a secure environment. When the authorized user utilizes the encrypted CNN model from the external provider, the user encrypts the query speech data using the secret key received from the model owner and uploads the encrypted speech data to the external provider. Finally, the external provider inputs the encrypted speech data into the encrypted model and returns the result to the authorized user. In this scenario, even if the external provider is not secure, only authorized users possessing the correct secret key can utilize the encrypted model as intended by the model owner, and only encrypted speech, with the privacy information concealed, is stored with the external provider. This ensures that privacy is maintained even in the event of a data leakage.

Similar to this task, the VoicePrivacy challenge [24] is famously related to speech privacy in the speech processing area. However, the privacy-preserving scenario of the VoicePrivacy challenge considers that the speaker wishes to keep their identity confidential while not protecting the content of the speech. In this case, the attacker aims to identify the speaker from the speech data. On the other hand, our paper presents a scenario where the user encrypts the raw speech data to preserve personal information, including the speech's content or a speaker's information. The attacker aims to steal and misuse speech data. Our method conceals auditory information by changing the raw speech data with a secret key.

3 Proposed methods

This section describes the encryption methods for query speech data, the methods for model encryption, and the advantage of using an orthogonal matrix as the secret key in the proposed methods. The proposed methods are familiar in the image processing area. However, we show the theoretical correctness of adapting the framework to speech data, which are represented in one or two dimensions.

3.1 Query encryption

We present a procedure for encrypting speech data using a secret key. Speech data consists of variable-length one-dimensional data, such as waveforms, or variable-length two-dimensional data, such as spectrograms obtained by applying a short-time Fourier transform to waveforms. Here, we propose three encryption methods using three kinds of orthogonal matrix, that is, the Shuffling in Section 3.1.1, Flipping in Section 3.1.2, and random orthogonal matrix (ROM) in Section 3.1.3, which are detailed below.

3.1.1 Shuffling

In this subsection, we describe an encryption method for speech data using "Shuffling", which uses a permutation matrix as the secret key. Algorithm 1 outlines the algorithm for encrypting speech data using Shuffling.

1. Speech data $\boldsymbol{X}$ is divided into blocks of block size M as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{11} & \dots & \boldsymbol{X}_{1j} & \dots & \boldsymbol{X}_{1t} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{X}_{i1} & \dots & \boldsymbol{X}_{ij} & \dots & \boldsymbol{X}_{it} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{X}_{f1} & \dots & \boldsymbol{X}_{fj} & \dots & \boldsymbol{X}_{ft} \end{bmatrix}. \tag{1}$$

   When defining the size of the speech data $\boldsymbol{X}$ as an $F \times T$ matrix, F represents the size in the frequency direction, and T represents the size in the time direction. For one-dimensional data such as a speech waveform, F is set to 1, while T is set to $[T/M]$, where M is the block size. For two-dimensional data like a spectrogram, F is determined by the frequency resolution, while both T and F are set to $[T/M]$ and $[F/M]$, respectively.

Algorithm 1 Shuffling speech data encryption

```
Require: $\boldsymbol{X}$: speech data, $M$: block size, $\boldsymbol{K}_\mathrm{s}$: secret key
Ensure: $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})}$: encrypted speech data
 1: $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})} \leftarrow O$
 2: $f \leftarrow [F/M]$
 3: $t \leftarrow [T/M]$
 4: for $((i = 0; i < f; i = i + M))$ do
 5:     for $((j = 0; j < t; j = j + M))$ do
 6:         $\boldsymbol{X}_{ij} \leftarrow \boldsymbol{X}[i : i + M, j : j + M]$
 7:         $\hat{\boldsymbol{X}}_{ij} \leftarrow \mathrm{flatten}(\boldsymbol{X}_{ij})$
 8:         $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})} \leftarrow \hat{\boldsymbol{X}}_{ij}[\boldsymbol{K}_\mathrm{s}]$
 9:         $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_\mathrm{s})} \leftarrow \mathrm{reshape}(\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})})$
10:         $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})}[i : i + M, j : j + M] \leftarrow \boldsymbol{X}_{ij}^{(\boldsymbol{K}_\mathrm{s})})$
11:     end for
12: end for
13: return $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})}$
```

Each block $\boldsymbol{X}_{ij}$ within the matrix $\boldsymbol{X}$ is further defined as follows:

$$\boldsymbol{X}_{ij} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}. \tag{2}$$

The size of the block $\boldsymbol{X}_{ij}$ can be expressed as $n \times m$, when $\boldsymbol{X}$ represents one-dimensional data, $n = 1$ and $m = M$, and when $\boldsymbol{X}$ represents two-dimensional data, n and m are both set to M.

2. To encrypt each block $\boldsymbol{X}_{ij}$ using a secret key, $\boldsymbol{X}_{ij}$ is flattened to a one-dimensional vector $\hat{\boldsymbol{X}}_{ij}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}_{ij} &= \mathrm{flatten}(\boldsymbol{X}_{ij}) \\ &= [x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_N], \end{aligned} \tag{3}$$

where N denotes the total number of elements in the block $\boldsymbol{X}_{ij}$, i.e., $N = n \times m$. The flatten function re-indexes the index of each element x as a row vector and behaves as follows: $[x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_N] = [x_{11}, x_{12}, \dots, x_{1m}, x_{21}, \dots, x_{nm}]$.

3. The secret key $\boldsymbol{K}_\mathrm{s}$ is generated as follows:

$$\boldsymbol{K}_\mathrm{s} = \{k_\mathrm{s}(1), k_\mathrm{s}(2), \dots, k_\mathrm{s}(k), \dots, k_\mathrm{s}(N)\}, \tag{4}$$

where the symbols l and k in (4) are indices of k_s, $k_\mathrm{s}(k) \in \{1, 2, \ldots, N\}$, $k_\mathrm{s}(k) \neq k_\mathrm{s}(l)$, $k, l \in \{1, 2, \ldots, N\}$, $k \neq l$. Algorithm 2 outlines the procedure for generating the secret key $\boldsymbol{K}_\mathrm{s}$.

Algorithm 2 Secret key generation in Shuffling

```
Require: $M$: block size
Ensure: $\boldsymbol{K}_\mathrm{s}$: secret key
1: $\boldsymbol{K}_\mathrm{s} \leftarrow [0, 1, \ldots, M-1]$
2: for $i = M-1$ to 1 do
3:     $j \leftarrow$ random integer from 0 to $i$
4:     Swap $\boldsymbol{K}_\mathrm{s}[i]$ and $\boldsymbol{K}_\mathrm{s}[j]$
5: end for
6:
7: return $\boldsymbol{K}_\mathrm{s}$
```

4. The permutation matrix $\boldsymbol{K}'_\mathrm{s}$ used for encryption is generated as follows:

$$\boldsymbol{K}'_\mathrm{s} = [\boldsymbol{e}_{k_\mathrm{s}(1)}, \boldsymbol{e}_{k_\mathrm{s}(2)}, \ldots, \boldsymbol{e}_{k_\mathrm{s}(k)}, \ldots, \boldsymbol{e}_{k_\mathrm{s}(N)}]. \tag{5}$$

   Let $\boldsymbol{e}_{k_\mathrm{s}(i)}$ denote the unit vector.

5. The matrix product of the one-dimensional vector $\hat{\boldsymbol{X}}_{ij}$ and the permutation matrix $\boldsymbol{K}'_\mathrm{s}$ is calculated to obtain the encrypted row vector $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})t} &= \boldsymbol{K}'_\mathrm{s} \hat{\boldsymbol{X}}_{ij}^{t} \\ &= \left[\boldsymbol{e}_{s(1)}, \ldots, \boldsymbol{e}_{s(N)}\right] [x_1, \ldots, x_N]^t \\ &= \left[x_1^{(k_\mathrm{s})}, \ldots, x_N^{(k_\mathrm{s})}\right]^t. \end{aligned} \tag{6}$$

6. Using the reshape function, the one-dimensional vector $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})}$ is transformed to a matrix of the same shape as the block $\boldsymbol{X}_{ij}$ to obtain an encrypted block $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_\mathrm{s})}$ as follows:

$$\begin{aligned} \boldsymbol{X}_{ij}^{(\boldsymbol{K}_\mathrm{s})} &= \mathrm{reshape}(\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{s})}) \\ &= \begin{bmatrix} x_{11}^{(k_\mathrm{s})} & x_{12}^{(k_\mathrm{s})} & \ldots & x_{1m}^{(k_\mathrm{s})} \\ x_{21}^{(k_\mathrm{s})} & x_{22}^{(k_\mathrm{s})} & \ldots & x_{2m}^{(k_\mathrm{s})} \\ \vdots & \vdots & & \vdots \\ x_{n1}^{(k_\mathrm{s})} & x_{n2}^{(k_\mathrm{s})} & \ldots & x_{nm}^{(k_\mathrm{s})} \end{bmatrix}. \end{aligned} \tag{7}$$

7. All the blocks of the speech data $\boldsymbol{X}$ that were divided in step 1 are processed in steps 2 - 6 to obtain the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})}$. $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})}$ can be expressed as follows:

$$\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{s})} = \begin{bmatrix} \boldsymbol{X}_{11}^{(\boldsymbol{K}_\mathrm{s})} & \dots & \boldsymbol{X}_{1t}^{(\boldsymbol{K}_\mathrm{s})} \\ \vdots & \ddots & \vdots \\ \boldsymbol{X}_{f1}^{(\boldsymbol{K}_\mathrm{s})} & \dots & \boldsymbol{X}_{ft}^{(\boldsymbol{K}_\mathrm{s})} \end{bmatrix}. \tag{8}$$

The steps from 1 to 7 demonstrate the transformation of plain speech data into encrypted speech data with the secret key $\boldsymbol{K}_\mathrm{s}$.

3.1.2 Flipping

In this subsection, we describe an encryption method for speech data using "Flipping", which uses a sign inversion as the secret key. Algorithm 3 outlines the algorithm for encrypting speech data using Flipping.

1. The speech data $\boldsymbol{X}$ is divided into blocks $\boldsymbol{X}_{ij}$ of block size M according to (1).

2. To encrypt each block $\boldsymbol{X}_{ij}$ using a secret key, $\boldsymbol{X}_{ij}$ is flattened to a one-dimensional vector $\hat{\boldsymbol{X}}_{ij}$ according to (3).

3. The secret key $\boldsymbol{K}_\mathrm{f}$ is generated. $\boldsymbol{K}_\mathrm{f}$ is denoted as follows:

$$\boldsymbol{K}_\mathrm{f} = \{k_\mathrm{f}(1), k_\mathrm{f}(2), \dots, k_\mathrm{f}(k), \dots, k_\mathrm{f}(N)\}, \tag{9}$$

where $k_\mathrm{f}(k) \in \{0, 1\}$, $P(X = k_\mathrm{f}(k)) = 0.5$, $1 \le k \le N$, $Pr(X = p)$ represents the probability that X takes the value p. Algorithm 4 outlines the procedure for generating the secret key $\boldsymbol{K}_\mathrm{f}$.

4. The matrix $\boldsymbol{K}'_\mathrm{f}$ used for encryption is generated as follows:

$$\boldsymbol{K}'_\mathrm{f}(k) = \begin{cases} -1 & (k_\mathrm{f}(k) = 1) \\ 1 & (k_\mathrm{f}(k) = 0) \end{cases}. \tag{10}$$

5. The Hadamard product of $\hat{\boldsymbol{X}}_{ij}$ and $\boldsymbol{K}'_\mathrm{f}$ is calculated to obtain the encrypted row vector $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{f})}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{f})} &= \boldsymbol{K}'_\mathrm{f} \odot \hat{\boldsymbol{X}}_{ij} \\ &= \left[\boldsymbol{K}'_\mathrm{f}(1), \boldsymbol{K}'_\mathrm{f}(2), \dots, \boldsymbol{K}'_\mathrm{f}(N)\right] \odot [x_1, \dots, x_N] \\ &= \left[x_1^{(k_\mathrm{f})}, \dots, x_N^{(k_\mathrm{f})}\right]. \end{aligned} \tag{11}$$

Algorithm 3 Flipping speech data encryption

```
Require: $\boldsymbol{X}$: speech data, $M$: block size, $\boldsymbol{K}_\mathrm{f}$: secret key
Ensure: $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}$: encrypted speech data
 1: $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})} \leftarrow O$
 2: $f \leftarrow [F/M]$
 3: $t \leftarrow [T/M]$
 4: for $((i = 0; i < f; i = i + M))$ do
 5:    for $((j = 0; j < t; j = j + M))$ do
 6:       $\boldsymbol{X}_{ij} \leftarrow \boldsymbol{X}[i : i + M, j : j + M]$
 7:       $\hat{\boldsymbol{X}}_{ij} \leftarrow \text{flatten}(\boldsymbol{X}_{ij})$
 8:       if $\boldsymbol{K}_\mathrm{f}[j] = 1$ then
 9:          $\hat{\boldsymbol{X}}^{(\boldsymbol{K}_\mathrm{f})}_{ij}[j] \leftarrow -\hat{\boldsymbol{X}}_{ij}[j]$
10:       end if
11:       $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}_{ij} \leftarrow \text{reshape}(\hat{\boldsymbol{X}}^{(\boldsymbol{K}_\mathrm{f})}_{ij})$
12:       $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}[i : i + M, j : j + M] \leftarrow \boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}_{ij}$
13:    end for
14: end for
15: return $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}$
```

Algorithm 4 Secret key generation in Flipping

```
Require: $M$: block size
Ensure: $\boldsymbol{K}_\mathrm{f}$: secret key
 1: Initialize $\boldsymbol{K}_\mathrm{f}$ with zeros of size $M$
 2: for each element in $\boldsymbol{K}_\mathrm{f}$ do
 3:    $\boldsymbol{K}_\mathrm{f} \leftarrow$ random number in $[0, 1)$
 4: end for
 5: $\boldsymbol{K}_\mathrm{f} \leftarrow (\boldsymbol{K}_\mathrm{f} \times 2)//1$
 6:
 7: return $\boldsymbol{K}_\mathrm{f}$
```

6. The one-dimensional vector $\hat{\boldsymbol{X}}^{(\boldsymbol{K}_\mathrm{f})}_{ij}$ is reshaped using the reshape function so that $\hat{\boldsymbol{X}}^{(\boldsymbol{K}_\mathrm{f})}_{ij}$ equals the unencrypted block $\boldsymbol{X}_{ij}$ according to (7), and the encrypted block $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}_{ij}$ is obtained.
7. All the blocks of the speech data $\boldsymbol{X}$ that were divided in step 1 are processed in steps 2 - 6 to obtain the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{f})}$.

The steps from 1 to 7 demonstrate the transformation of plain speech data into encrypted speech data with the secret key $\boldsymbol{K}_\mathrm{f}$.

3.1.3 Random orthogonal matrix

In this section, we describe an encryption method for speech data using “ROM”, which uses a randomly generated orthogonal matrix as a secret key. Algorithm 5 outlines the algorithm for encrypting speech data using ROM.

Algorithm 5 ROM speech data encryption

```
Require: $\boldsymbol{X}$: speech data, $M$: block size, $\boldsymbol{K}_{\mathrm{r}}$: secret key
Ensure: $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{r}})}$: encrypted speech data
 1: $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{r}})} \leftarrow O$
 2: $f \leftarrow [F/M]$
 3: $t \leftarrow [T/M]$
 4: for $((i = 0; i < f; i = i + M))$ do
 5:     for $((j = 0; j < t; j = j + M))$ do
 6:         $\boldsymbol{X}_{ij} \leftarrow \boldsymbol{X}[i : i + M, j : j + M]$
 7:         $\hat{\boldsymbol{X}}_{ij} \leftarrow \mathrm{flatten}(\boldsymbol{X}_{ij})$
 8:         $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})} \leftarrow \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})} \boldsymbol{K}_{\mathrm{r}}$
 9:         $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})} \leftarrow \mathrm{reshape}(\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})})$
10:         $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{r}})}[i : i + M, j : j + M] \leftarrow \boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})}$
11:     end for
12: end for
13: return $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{r}})}$
```

1. The speech data $\boldsymbol{X}$ is divided into blocks $\boldsymbol{X}_{ij}$ of block size M according to (1).
2. To encrypt each block $\boldsymbol{X}_{ij}$ using a secret key, $\boldsymbol{X}_{ij}$ is flattened to a one-dimensional vector $\hat{\boldsymbol{X}}_{ij}$ according to (3).
3. The secret key $\boldsymbol{K}_{\mathrm{r}}$ is generated. $\boldsymbol{K}_{\mathrm{r}}$ is denoted as follows:

$$\boldsymbol{K}_{\mathrm{r}} = \begin{bmatrix} k_{11} & \dots & k_{1N} \\ \vdots & \ddots & \vdots \\ k_{N1} & \dots & k_{NN} \end{bmatrix} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \dots, \boldsymbol{k}_N]\,. \tag{12}$$

   Algorithm 6 outlines the procedure for generating the secret key $\boldsymbol{K}_{\mathrm{r}}$.

4. The matrix product of $\hat{\boldsymbol{X}}_{ij}$ and $\boldsymbol{K}_{\mathrm{r}}$ is calculated to obtain the encrypted row vector $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})} &= \hat{\boldsymbol{X}}_{ij} \boldsymbol{K}_{\mathrm{r}} \\ &= [x_1, \dots, x_N]\,[\boldsymbol{k}_1, \dots, \boldsymbol{k}_N] \\ &= \left[x_1^{(k_{\mathrm{r}})}, \dots, x_N^{(k_{\mathrm{r}})}\right]. \end{aligned} \tag{13}$$

Algorithm 6 Secret key generation in ROM [14]

```
Require: $M$: block size
Ensure: $\boldsymbol{K}_\mathrm{r}$: secret key
 1: $\boldsymbol{A} \leftarrow$ random normal matrix of size $M \times M$
 2: $(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow$ QR decomposition of $\boldsymbol{A}$
 3: for $i = 1$ to $M$ do
 4:     if $\boldsymbol{R}[i, i] < 0$ then
 5:         $Q[:, i] \leftarrow -Q[:, i]$
 6:     end if
 7: end for
 8: $\boldsymbol{K}_\mathrm{r} \leftarrow Q$
 9:
10: return $\boldsymbol{K}_\mathrm{r}$
```

5. The one-dimensional vector $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{r})}$ is reshaped using the reshape function so that $\hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_\mathrm{r})}$ is equal to the unencrypted block $\boldsymbol{X}_{ij}$ according to (7), and the encrypted block $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_\mathrm{r})}$ is obtained.
6. All the blocks of the speech data $\boldsymbol{X}$ that were divided in step 1 are processed in steps 2 - 5 to obtain the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_\mathrm{r})}$.

3.2 Model encryption

To input speech data encrypted using the procedure shown in Section 3.1 directly into the trained model without decrypting it, it is required to transform a part of the trained model. Therefore, we assume that the first layer of the CNN model is a convolutional layer, and that the kernel size and stride size of the first convolutional layer are equal. This is because equal kernel and stride sizes allow the convolution process to be performed for each encrypted block. Let $\boldsymbol{E}$ be the kernel of the first convolutional layer in which encryption is applied as shown in Figure 2, and let P be the kernel size. Then, the kernel $\boldsymbol{E}$ can be expressed as follows:

$$\boldsymbol{E} = \begin{bmatrix} e_{11} & e_{12} & \dots & e_{1b} \\ e_{21} & e_{22} & \dots & e_{2b} \\ \vdots & \vdots & & \vdots \\ e_{a1} & e_{a2} & \dots & e_{ab} \end{bmatrix}, \tag{14}$$

where the size of the kernel $\boldsymbol{E}$ can be expressed as $a \times b$. When the speech data $\boldsymbol{X}$ is one-dimensional data, $a = 1$ and $b = P$, and when it is two-dimensional data, $a = P$ and $b = P$. In addition, in this paper, the kernel

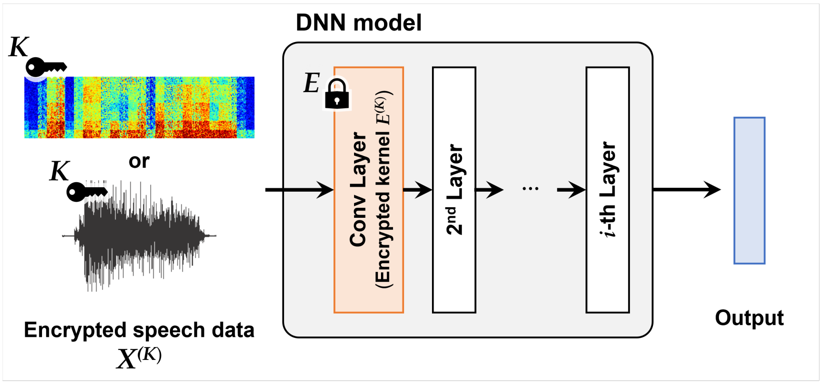


Figure 2: Encrypted models for accepting encrypted speech data

size and the block size M used for encryption are assumed to be equal, i.e., $P = M$. On the basis of these assumptions, when plain speech data $\boldsymbol{X}$ is input to the convolution layer, a convolution operation is performed using the divided block $\boldsymbol{X}_{ij}$ and the kernel $\boldsymbol{E}$ as follows:

$$z = \boldsymbol{X}_{ij} \cdot \boldsymbol{E}. \tag{15}$$

We consider a scenario where the input data consists of encrypted speech data and aim to eliminate the effects of encryption without decrypting the encrypted speech data by using the kernel $\boldsymbol{E}$. The encryption procedures for the kernel $\boldsymbol{E}$ for Shuffling, Flipping, and ROM are detailed in Sections 3.2.1, 3.2.2, and 3.2.3, respectively.

3.2.1 Shuffling

The encryption procedure of the model when using the secret key $\boldsymbol{K}_{\mathrm{s}}$ obtained from Shuffling will be explained. First, the kernel $\boldsymbol{E}$ is flattened to a one-dimensional row vector using the flatten function to obtain $\hat{\boldsymbol{E}}$. Next, the matrix product of the permutation matrix $\boldsymbol{K}'_{\mathrm{s}}$ and the transposed $\hat{\boldsymbol{E}}$ is calculated to obtain the encrypted column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{s}})}$ as follows:

$$\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{s}})} = \boldsymbol{K}'_{\mathrm{s}}\hat{\boldsymbol{E}}^{t} = \left[e_{11}^{(k_{\mathrm{s}})}, e_{12}^{(k_{\mathrm{s}})}, \cdots, e_{ab}^{(k_{\mathrm{s}})}\right]^{t}. \tag{16}$$

The column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{s}})}$ is reshaped to be a matrix of the same size as the unencrypted kernel $\boldsymbol{E}$ to obtain the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{s}})}$ as follows:

$$\begin{aligned}\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{s}})} &= \mathrm{reshape}(\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{s}})}) \\ &= \begin{bmatrix} e_{11}^{(k_{\mathrm{s}})} & e_{12}^{(k_{\mathrm{s}})} & \cdots & e_{1b}^{(k_{\mathrm{s}})} \\ e_{21}^{(k_{\mathrm{s}})} & e_{22}^{(k_{\mathrm{s}})} & \cdots & e_{2b}^{(k_{\mathrm{s}})} \\ \vdots & \vdots & & \vdots \\ e_{a1}^{(k_{\mathrm{s}})} & e_{a2}^{(k_{\mathrm{s}})} & \cdots & e_{ab}^{(k_{\mathrm{s}})} \end{bmatrix}. \end{aligned} \tag{17}$$

When calculating convolution with the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{s}})}$ and the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{s}})}$, the computation for each encrypted block $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{s}})}$ is as follows:

$$\begin{aligned} z^{(\boldsymbol{K}_{\mathrm{s}})} &= \boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{s}})} \cdot \boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{s}})} = \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{s}})} \hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{s}})} \\ &= \hat{\boldsymbol{X}}_{ij} \boldsymbol{K}_{\mathrm{s}}' \boldsymbol{K}_{\mathrm{s}}'^{t} \hat{\boldsymbol{E}}^{t} = \boldsymbol{X}_{ij} \cdot \boldsymbol{E} = z. \end{aligned} \tag{18}$$

Since the permutation matrix is a kind of orthogonal matrix, the product of $\boldsymbol{K}_{\mathrm{s}}'$ and $\boldsymbol{K}_{\mathrm{s}}'^{t}$ is a unit matrix according to the property of the orthogonal matrix. By inputting speech data encrypted with the same secret key used to encrypt the model into the encrypted model, the internal computations produce identical results as if no encryption had been performed. Therefore, it is possible to input encrypted speech data into the model without decryption, allowing for the correct utilization of the model while preserving the privacy of the speech data. On the other hand, when encrypted speech data, encrypted using a different secret key from that used for the model encryption, is input into the encrypted model, the results differ from those obtained without encryption. Therefore, it is hard to use the model correctly without prior knowledge of the secret key. In this framework, since the encryption process is applied to the original speech data after it has been recorded, the presence of noise in the original speech data does not affect the encryption. As shown in (18), when the correct key is used, the impact of the secret key is canceled out during the inner product calculation. While the performance of the original model may degrade if it is not robust to noise, this is due to the inherent characteristics of the model and not due to the proposed encryption method. Our method does not interfere with the inner product computation the model performs when the correct key is used.

3.2.2 Flipping

The encryption procedure of the model when using the secret key $\boldsymbol{K}_{\mathrm{f}}$ obtained from Flipping will be explained. First, the kernel $\boldsymbol{E}$ is flattened to a one-dimensional row vector using the flatten function to obtain $\hat{\boldsymbol{E}}$. Next, the

Hadamard product of $\boldsymbol{K}'_{\mathrm{f}}$ and $\hat{\boldsymbol{E}}$ is calculated to obtain the encrypted column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{f}})}$ as follows:

$$\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{f}})} = \boldsymbol{K}'^{t}_{\mathrm{f}} \odot \hat{\boldsymbol{E}}^{t} = \left[e_{11}^{(k_{\mathrm{f}})}, e_{12}^{(k_{\mathrm{f}})}, \cdots, e_{ab}^{(k_{\mathrm{f}})}\right]^{t}. \tag{19}$$

The column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{f}})}$ is reshaped to be a matrix of the same size as the unencrypted kernel $\boldsymbol{E}$ to obtain the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{f}})}$ according to (17). When calculating convolution with the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{f}})}$ and the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{f}})}$, the computation for each encrypted block $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{f}})}$ is as follows:

$$\begin{aligned} z^{(\boldsymbol{K}_{\mathrm{f}})} &= \boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{f}})} \cdot \boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{f}})} = \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K}_{\mathrm{f}})} \hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{f}})} = (\boldsymbol{K}'_{\mathrm{f}} \odot \boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{f}})})(\boldsymbol{K}'^{t}_{\mathrm{f}} \odot \hat{\boldsymbol{E}}^{t}) \\ &= \big([\boldsymbol{K}'_{\mathrm{f}}(1) \dots \boldsymbol{K}'_{\mathrm{f}}(N)] \odot [x_1 \dots x_N]\big) \left(\begin{bmatrix} \boldsymbol{K}'_{\mathrm{f}}(1) \\ \vdots \\ \boldsymbol{K}'_{\mathrm{f}}(N) \end{bmatrix} \odot \begin{bmatrix} e_{11} \\ \vdots \\ e_{ab} \end{bmatrix} \right) \\ &= [\boldsymbol{K}'_{\mathrm{f}}(1)x_1 \dots \boldsymbol{K}'_{\mathrm{f}}(N)x_N] \begin{bmatrix} \boldsymbol{K}'_{\mathrm{f}}(1)e_{11} \\ \vdots \\ \boldsymbol{K}'_{\mathrm{f}}(N)e_{ab} \end{bmatrix} = \boldsymbol{X}_{ij} \cdot \boldsymbol{E} = z. \end{aligned} \tag{20}$$

Since $\boldsymbol{K}'_{\mathrm{f}}$ is a matrix consisting of -1 or 1, we can obtain completely the same results before and after using the proposed method by inputting speech data encrypted with the secret key $\boldsymbol{K}_{\mathrm{f}}$ to the model encrypted with the same secret key $\boldsymbol{K}_{\mathrm{f}}$. Therefore, as well as with Shuffling, it is possible to input the encrypted speech data into the model without decrypting it.

3.2.3 Random orthogonal matrix

The encryption procedure of the model when using the secret key $\boldsymbol{K}_{\mathrm{r}}$ obtained from ROM will be explained. First, the kernel $\boldsymbol{E}$ is flattened to a one-dimensional row vector using the flatten function to obtain $\hat{\boldsymbol{E}}$. Next, the matrix product of $\boldsymbol{K}^{t}_{\mathrm{r}}$ and $\hat{\boldsymbol{E}}^{t}$ is calculated to obtain the encrypted column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{r}})}$ as follows:

$$\hat{\boldsymbol{E}}^{(\boldsymbol{K})} = \boldsymbol{K}^{t}_{\mathrm{r}} \hat{\boldsymbol{E}}^{t} = \left[e_{11}^{(k)}, e_{12}^{(k)}, \cdots, e_{ab}^{(k)}\right]^{t}. \tag{21}$$

The column vector $\hat{\boldsymbol{E}}^{(\boldsymbol{K}_{\mathrm{r}})}$ is reshaped to be a matrix of the same size as the unencrypted kernel $\boldsymbol{E}$ to obtain the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{r}})}$ according to (17). When calculating convolution with the encrypted speech data $\boldsymbol{X}^{(\boldsymbol{K}_{\mathrm{r}})}$ and the encrypted kernel $\boldsymbol{E}^{(\boldsymbol{K}_{\mathrm{r}})}$, the computation for each encrypted block $\boldsymbol{X}_{ij}^{(\boldsymbol{K}_{\mathrm{r}})}$ is as follows:

$$\begin{aligned} z^{(\boldsymbol{K})} &= \boldsymbol{X}_{ij}^{(\boldsymbol{K})} \cdot \boldsymbol{E}^{(\boldsymbol{K})} = \hat{\boldsymbol{X}}_{ij}^{(\boldsymbol{K})} \hat{\boldsymbol{E}}^{(\boldsymbol{K})} \\ &= \hat{\boldsymbol{X}}_{ij} \boldsymbol{K}_{\mathrm{r}} \boldsymbol{K}_{\mathrm{r}}^{t} \hat{\boldsymbol{E}}^{t} = \boldsymbol{X}_{ij} \cdot \boldsymbol{E} = z. \end{aligned} \tag{22}$$

The matrix product of $\boldsymbol{K}_{\mathrm{r}}$ and $\boldsymbol{K}_{\mathrm{r}}^{t}$ is a unit matrix according to the property of the orthogonal matrix. By inputting speech data encrypted with a secret key $\boldsymbol{K}_{\mathrm{r}}$ into a model encrypted with the same secret key $\boldsymbol{K}_{\mathrm{r}}$, completely the same results can be obtained before and after using the proposed method. Therefore, as well as with Shuffling and Flipping, it is possible to input the encrypted speech data into the model without decoding it.

3.3 Key space of secret key

In this section, we discuss the key space size for the secret keys utilized in Shuffling, Flipping, and ROM. Concerning Shuffling, as depicted in (4), the secret key $\boldsymbol{K}_{\mathrm{s}}$ is a matrix for rearranging the indices of elements within each block $\boldsymbol{X}_{ij}$ in any order, e.g., $\boldsymbol{K}_{\mathrm{s}} = [3, 1, 2]$ when the input data $\boldsymbol{X}$ is one-dimensional and $M = 3$. Therefore, when the speech data $\boldsymbol{X}$ is one-dimensional, the secret key $\boldsymbol{K}_{\mathrm{s}}$ can have $M!$ possible patterns, and when the speech data $\boldsymbol{X}$ is two-dimensional, it can have $(M \times M)!$ possible patterns. Concerning Flipping, as depicted in (9), the secret key $\boldsymbol{K}_{\mathrm{f}}$ is a bit sequence consisting of 0 or 1, e.g., $\boldsymbol{K}_{\mathrm{f}} = [0, 0, 1]$ when the input data is one-dimensional and $M = 3$. Therefore, the secret key $\boldsymbol{K}_{\mathrm{f}}$ can only be used in 2^M ways when the speech data $\boldsymbol{X}$ is one-dimensional and in $2^{M \times M}$ ways when the speech data $\boldsymbol{X}$ is two-dimensional. Since each bit in K_f is generated independently with a probability of 0.5, all 2^M or $2^{M \times M}$ patterns occur with equal probability. Concerning ROM, as depicted in (12), the secret key $\boldsymbol{K}_{\mathrm{r}}$ is an orthogonal matrix and is composed of randomly generated real-valued elements, including negative values. As an example of the secret key $\boldsymbol{K}_{\mathrm{r}}$, the matrix for the case where the input data is one-dimensional and $M = 3$ is shown below:

$$\boldsymbol{K}_{\mathrm{r}} = \begin{bmatrix} 0.9898 & -0.0661 & -0.1264 \\ 0.1309 & 0.7732 & 0.6205 \\ 0.0568 & -0.6307 & 0.7740 \end{bmatrix}. \tag{23}$$

As shown in (23), $\boldsymbol{K}_{\mathrm{r}}$ is a matrix of $M \times M$ when the speech data $\boldsymbol{X}$ is one-dimensional and $M^2 \times M^2$ when $\boldsymbol{X}$ is two-dimensional, allowing for the generation of many patterns of secret keys. Focusing on periods of silence in the speech data, there is a risk that the secret key can be easily estimated by a third party if the keyspace is small. Therefore, it is better to have a large key space for the secret key to make the prediction of the secret key more difficult.

Despite the limitation that the secret key must be an orthogonal matrix, ROM has the advantage of the key space of the secret key being larger than that of Shuffling and Flipping, making the prediction of the secret key much more difficult.

3.4 Key generation procedure

This section provides the concrete procedure to generate a secret key pair.

For Shuffling, the Python library `torch.randperm(n=M)` is employed to generate the secret key $\boldsymbol{K}_{\mathrm{s}}$. Then, its transpose matrix $\boldsymbol{K}_{\mathrm{s}}^{t}$ is calculated with `torch.t`.

For ROM, the Python library `scipy.stats.ortho_group.rvs(dim=M)` is employed to generate the secret key $\boldsymbol{K}_{\mathrm{r}}$. Then, its transpose matrix $\boldsymbol{K}_{\mathrm{r}}^{t}$ is calculated with `torch.t`.

For the evaluation, many secret key pairs are generated by changing the seed value, and the keys of different pairs are regarded as incorrect.

4 Experiment

4.1 Evaluation of privacy-preserving performance

4.1.1 Experimental conditions

In this experiment, we evaluated the privacy-preserving performance of the proposed methods using ASV, ASR, and ASC tasks.

ASV is a technology used to verify the identity of a speaker by analyzing their speech characteristics. The task of ASV involves determining whether a claimed speaker matches the true identity by comparing speech samples. Within the privacy-preserving scenario of the ASV task, encryption is used with the aim of ensuring that authentication is only successful when the correct secret key is used, while performance significantly deteriorates when an incorrect key is utilized. The ASV experiment is assumed to assess privacy-preserving performance against the speaker's identity. For the ASV system, we used an x-vector-based ASV system [21] with a self-supervised-learning (SSL) based front-end model [28, 1]. For the ASV task, we trained a HuBERT model [4] with the LibriSpeech corpus [18] following the Fairseq recipe [17]. The HuBERT model is used as an SSL-based front-end model, and it is regarded as one of the state-of-the-art systems. The input features for the HuBERT model are waveforms. The structure and hyperparameters of the HuBERT model were the same as those of HuBERT Base [4], except that the stride size P of the first convolutional layer was changed to ten. The speech expression outputted from the HuBERT model was inputted to an x-vector-based embedding network. This network was trained with the VoxCeleb1 corpus [15], using the same hyperparameters as in Yang *et al.* [28]. For the ASV evaluation, we used the VoxCeleb1 test set, and the input waveforms were encrypted by the proposed encryption methods. The block size M for the encryption methods was set to 10. Equal error rate (EER) was used as the evaluation metric.

ASR is the process of transcribing speech content into text. Within the privacy-preserving scenario of the ASR task, encryption is performed with the aim of ensuring that recognition is only successful when the correct secret key is used, while the performance of the speech recognition deteriorates significantly when an incorrect key is utilized. The ASR experiment is assumed to assess privacy-preserving performance against the speech content. For the ASR task, we trained a transformer model with the LibriSpeech corpus following the ESPnet2 recipe [26]. The transformer architecture and hyperparameters were the same as in Karita *et al.* [7], except for the input feature and the stride size of the first convolutional layer. The acoustic features are input as two-dimensional log-mel spectrogram features by arranging the 80-dimensional log-mel filterbank features extracted for each frame. The stride size P of the first convolutional layer was set to three to use the proposed methods. For the ASR evaluation, we used the LibriSpeech test clean subset, and the input log-mel spectrogram features were encrypted by the proposed methods. The block size M for the encryption was set to three to match the kernel size P. Word error rate (WER) was used as an evaluation metric. We also evaluated how block size influences the performance of the proposed methods in concealing the speech content within the encrypted speech. Under $M = 5, 10, 20, 128$, speech waveforms encrypted using the proposed methods were input to the plain pre-trained speech recognition model published in Karita *et al.* [7].

ASC is the task of categorizing audio recordings depending on the type or category of the surrounding environment in which they were captured [13]. Unlike ASR, ASC focuses on classifying the acoustic event of various environments. ASC systems analyze features extracted from audio signals, such as spectrograms, and use machine learning algorithms to classify the audio into predefined categories or classes. The ASC experiment is assumed to assess privacy-preserving performance against an acoustic event. For the ASC task, we used the ConvMixer [25] model trained on the SINS [3] dataset. We used only the SINS data labeled Absence, Cooking, Dishwashing, Eating, Other, Vacuumcleaner, Watching TV, Working, Calling, and Visit, and data labeled Calling and Visit were combined into a single class and labeled Social Activity. The input features for the ConvMixer model are spectrograms, which are two-dimensional speech data. The data were clipped so that the length of each sample was four seconds, and spectrograms were generated from the clipped data. The structure and hyperparameters of the ConvMixer model were the same as those of ConvMixer-768/32 [25], except that the stride size P and the kernel size of the first convolutional layer was changed to eight. For the ASC evaluation, we used the test set of the SINS dataset, and input spectrograms were encrypted by the proposed methods. The block size M for the encryption methods was set to eight. Accuracy, which indicates the percentage of correct classifications, was used as an index to evaluate the classification results.

4.1.2 Experimental results

Tables 1, 2, and 3 show the results of the ASV, ASR and ASC experiments, respectively. In these experiments, the plain model, i.e., no encryption, and the models encrypted by Shuffling, Flipping and ROM were used. "Correct key" refers to a situation where the encryption key used for the model matches the encryption key used for the query. "Incorrect key" refers to a situation where the encryption key used for the model does not match the encryption key used for the query. The results of "Incorrect key" are based on the average values obtained when using five different incorrect secret keys. "Plain" in the context of query encryption can be regarded as a form of an incorrect key, indicating that encryption has not been performed.

Table 1 shows the results of the ASV experiments. The EERs for "Correct key" were completely the same as those of the plain model. In the "Incorrect key" and "Plain" cases, the EERs were higher than those for "Correct key." These results show that only authorized users who know the correct secret key can correctly use the encrypted model in the ASV task by using the proposed methods. Furthermore, these results show that there is not much difference in the trend of the results between the methods, and that all of the methods succeed in concealing the speaker identity.

Table 1: EER(%) in encryption scenario ($M = 10$) for ASV.

Model encryption	Query encryption		
	Plain	Correct key	Inorrect key
Plain	7.91	-	-
Shuffling	36.7	7.91	33.3
Flipping	37.2	7.91	34.1
ROM	35.3	7.91	35.1

Table 2 shows the results of the ASR experiments. The WERs of the encrypted models for "Correct key" were completely the same as those of the plain model. In the "Incorrect key" and "Plain" case, the WERs were higher than those of the "Correct key" case, especially in the Flipping and ROM case. Therefore, in the "Incorrect key" and "Plain" case, the encrypted models hardly extracted the speech content, so the encryption by the proposed methods was highly anonymous. As well as with the ASV results, these results also show that only authorized users who know the correct key can correctly use the encrypted model in the ASR task within the proposed methods.

Table 2: WER(%) in encryption scenario ($M = 3$) for ASR.

Model encryption	Query encryption		
	Plain	Correct key	Incorrect key
Plain	4.4	-	-
Shuffling	10.9	4.4	14.18
Flipping	98.1	4.4	98.26
ROM	99.7	4.4	97.62

Table 3 shows the results of the ASR experiments. The accuracies of the encrypted models for "Correct key" were completely the same as those of the plain model. In the "Incorrect key" case, the accuracies were higher than those of the "Correct key" case. In the "Plain" scenario, the accuracy was also significantly increased since it can be regarded as one of the "Incorrect key" cases. These results also show that only authorized users who know the correct key can correctly use the encrypted model in the ASC task by using the proposed methods. On the basis of the privacy-preserving performance evaluation experiments described above, it is confirmed that the proposed methods prevent unauthorized users who do not know the secret key used to encrypt the model from using the model with high performance.

Table 3: Accuracy (%) in encryption scenario ($M = 8$) for ASC.

Model encryption	Query encryption		
	Plain	Correct key	Incorrect key
Plain	85.4	-	-
Shuffling	64.6	85.4	60.5
Flipping	1.97	85.4	1.99
ROM	1.97	85.4	1.99

Encrypted speech can also be obtained as severely noisy speech, so speech waveforms encrypted with Shuffling, Flipping, and ROM were input to the ASR model to investigate the encryption robustness of the proposed methods and the plain model, and the results are shown in Table 4. The WERs for the encrypted speech were higher in all conditions than the WER for the unencrypted speech, 2.7%, and we can see that for all encryption methods, the WER increased as the block size M increased. Furthermore, when M was small, for example $M = 5, 10$, the WER was higher when the waveform was encrypted by Flipping or ROM than when it was encrypted by Shuffling, indicating that the privacy-preserving performances of Flipping or ROM were

Table 4: Comparison of WER (%) of ASR model on LibriSpeech corpus (test clean subsets) encrypted using Shuffling, Flipping, and ROM. WER for unencrypted query input to plain model is 2.7%.

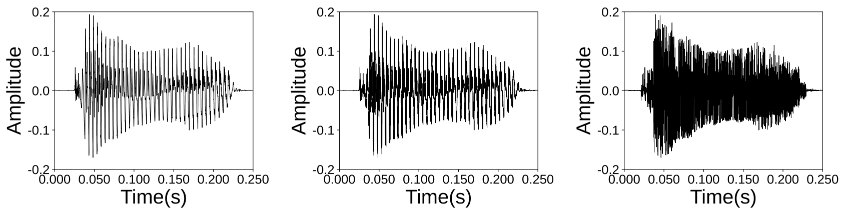
M	Shuffling	Flipping	ROM
5	13.9	40.5	28.9
10	30.6	68.1	57.6
20	63.9	82.9	85.4
128	94.9	95.0	94.8

better. In particular, for encryption using ROM, the key space is sufficiently large even when M is small, making it difficult to predict the secret key.

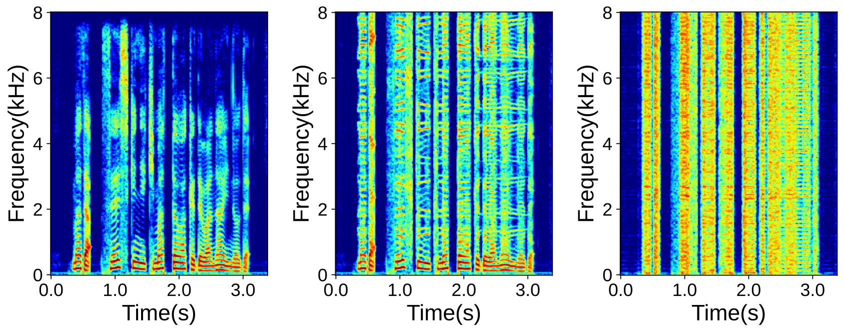
4.2 Comparison of encrypted speech data

In this section, we investigated the characteristics of each proposed encryption method in some examples. The original speech waveform is shown in Figures 3(a), Figures 3(b) and 3(c) are waveforms encrypted by Shuffling from the original waveform with different key sizes ($M = 10, 128$), and Figures 3(d), 3(e), and 3(f) are their spectrograms, respectively. Figure 4 shows the waveforms encrypted by Flipping and ROM and their spectrograms. The key sizes are the same as the Flipping case. From these figures, it can be confirmed that the speech waveforms encrypted by the proposed encryption methods are significantly different from the original speech waveform. The spectrograms of each speech waveform show that the original speech waveform is significantly different due to the encryption using the proposed methods, and the characteristics of the original speech waveform are also significantly different. It was also found that the larger the value of the block size M used for the proposed encryption methods, the larger the change in the waveforms. With a larger value of M, the speech content can be concealed more effectively. Additionally, from the perspective of key space, a larger value of M makes it more difficult to estimate the secret key.

The original speech waveform is shown in Figure 5(a), and Figures 5(b) and 5(c) are spectrograms encrypted from the original spectrogram by Shuffling with different key sizes ($M = 8, 32$). By comparing Figures 5(a) and 5(b) it can be seen that the harmonic structures of each spectrogram are distorted. Comparing Figures 5(b) and 5(c), we can see that the larger the value of M, the greater the range of movement for the positions of values in each block of the spectrograms. The spectrograms encrypted by Flipping and ROM are shown in Figure 6. By comparing Figures 5(a) and 6(a), and Figures 5(a) and 6(c), we can see that the magnitude of each value in the encrypted spectrogram changes randomly. Comparing Figures 6(a) and 6(b), and Figures 6(c) and 6(d), we can see that the spectrogram values in the block change regardless of



(a) Original waveform (b) Encrypted waveform ($M = 10$) (c) Encrypted waveform ($M = 128$)



(d) Original spectrogram (e) Encrypted spectrogram ($M = 10$) (f) Encrypted spectrogram ($M = 128$)

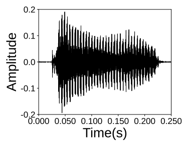
Figure 3: Examples of waveform encrypted by Shuffling

the block size M. From these figures, it can be visually confirmed that the encryption of the proposed methods succeeded in anonymizing the speech with encryption since the harmonic structure of the original spectrogram was hidden and, in particular, ROM obscured the speech segment and other information.

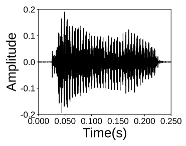
4.3 Robustness evaluation experiments

4.3.1 Experimental conditions

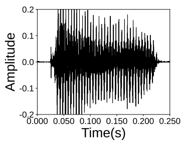
Four experiments were conducted to evaluate robustness against attacks with the proposed methods. In the first experiment, we investigated the changes in the spectrogram encrypted by ROM when decrypted with the correct key and an incorrect key. In the second experiment, the spectrogram encrypted by ROM was reconstructed into a speech waveform using Phase Gradient Heap Integration (PGHI) [19], one of the state-of-the-art phase reconstruction algorithms, and the reconstructed speech waveform was compared with the original one. In the third experiment, we prepared an ASV model encrypted with the correct secret key and input speech encrypted with 1000 incorrect keys to the model. The two ASV models used in this experiment were HuBERT [4] and RawNeXt [8]. As in the experiment in Section 4, we prepared two kinds



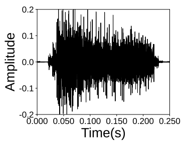
(a) Encrypted waveform ($M = 10$)



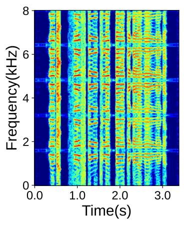
(b) Encrypted waveform ($M = 128$)



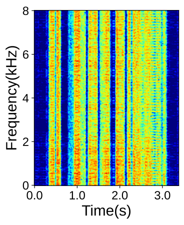
(e) Encrypted waveform ($M = 10$)



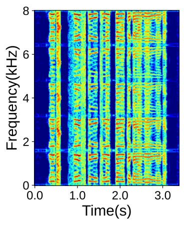
(f) Encrypted waveform ($M = 128$)



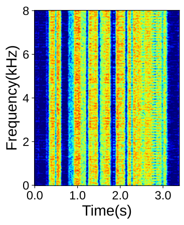
(c) Encrypted spectrogram ($M = 10$)



(d) Encrypted spectrogram ($M = 128$)



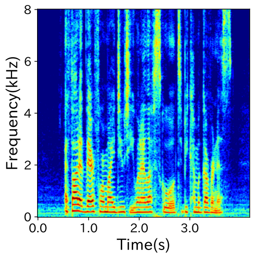
(g) Encrypted spectrogram ($M = 10$)



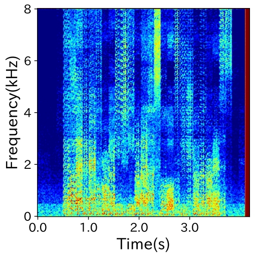
(h) Encrypted spectrogram ($M = 128$)

Flipping **ROM**

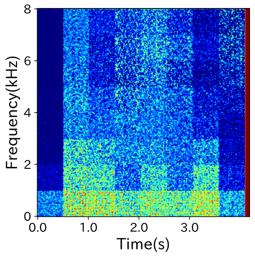
Figure 4: Examples of waveform encrypted by Flipping and ROM



(a) Original spectrogram



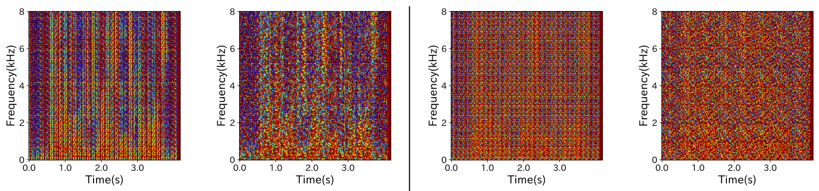
(b) Encrypted spectrogram ($M = 8$)



(c) Encrypted spectrogram ($M = 32$)

Figure 5: Examples of spectrogram encrypted by Shuffling

of HuBERT model that included the first convolutional layer with kernel sizes and stride sizes of five and ten, respectively. The first convolutional layers of HuBERT5 and HuBERT10 were encrypted by ROM. The RawNeXt model took speech waveforms as input features and was trained using the VoxCeleb2 corpus [2]. In this experiment, encryption with ROM was applied to the pre-trained models distributed in Kim *et al.* [8]. The RawNeXt model was originally set to a stride size and kernel size of three. For evaluation, we used the VoxCeleb1 corpus's test set [15], and EER was used as the evaluation measure. In the fourth experiment, the same experiment as the third experiment was



(a) Encrypted spectrogram ($M = 8$)

(b) Encrypted spectrogram ($M = 32$)

(c) Encrypted spectrogram ($M = 8$)

(d) Encrypted spectrogram ($M = 32$)

Flipping **ROM**

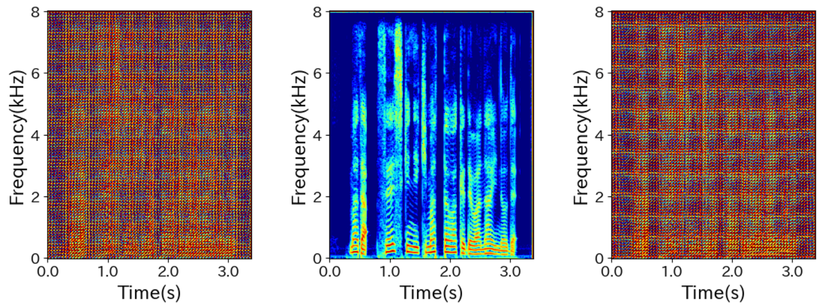
Figure 6: Examples of spectrogram encrypted by Flipping and ROM

applied to the ASR task. In the experiments, speech waveforms encrypted with 100 ROM secret keys were input to a trained ASR model proposed in Karita *et al.* [7]. Under $M = 3, 5, 10$, we used the test clean subset of LibriSpeech as the evaluation data.

4.3.2 Experimental results

Figure 7 shows the result of the first experiment of the robustness evaluation. Figure 7(a) is the spectrogram encrypted by using ROM from the original spectrogram in Figure 3(d) under block size $M = 3$. Figure 7(b) shows the spectrogram in Figure 7(a) decrypted using the correct key, while Figure 7(c) shows the spectrogram in Figure 7(a) decrypted using the incorrect key. Figures 3(d) and 7(b) show that the spectrogram decrypted with the correct key was exactly the same as the original spectrogram. On the other hand, Figure 7(c) shows that the original spectrogram information was not decrypted when an incorrect key was used as a decrypted key. Figures 3(d) and 7(b) show that the spectrograms were exactly the same as the original spectrograms when decrypted with the correct keys. On the other hand, Figures 3(d) and 7(c) show that the original spectrogram information was not decrypted when decrypted using an incorrect key.

The results of phase reconstruction using PGHI on the spectrogram encrypted using ROM are shown in Figure 8 as the second experiment of the robustness evaluation. Figure 8(a) is the original spectrogram, Figure 8(b) is the spectrogram obtained by encrypting the spectrogram in Figure 8(a) with ROM under block size $M = 3$, and Figure 8(c) is the spectrogram of the speech waveform obtained by applying PGHI to the spectrogram in Figure 8(b). The figures shown in the upper rows of Figures 8(a) and 8(c) are the original speech waveform and the speech waveform obtained by PGHI, respectively. Comparing Figure 8(a) with Figure 8(c), it can be seen that the structure of the original spectrogram was not reconstructed in the spectrogram after PGHI.

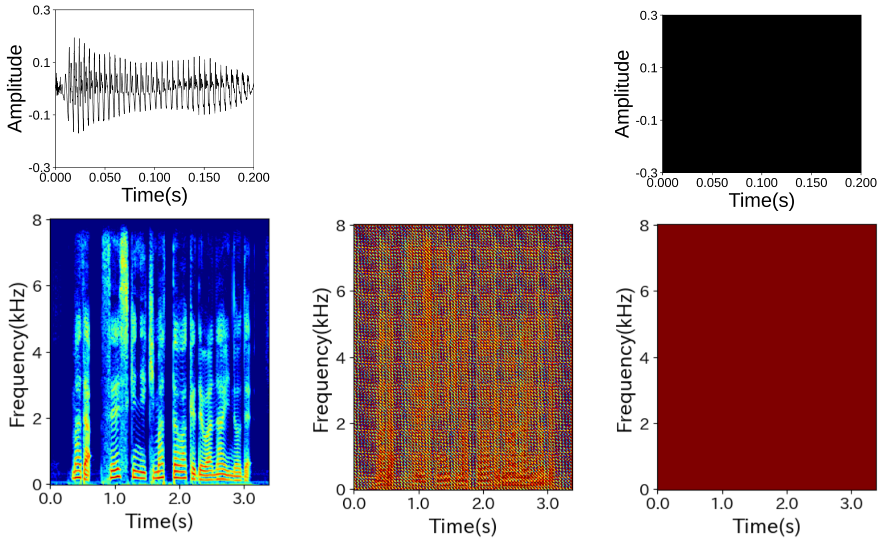


(a) Encrypted spectrogram ($M = 3$)

(b) Spectrogram decrypted with correct key

(c) Spectrogram decrypted with incorrect key

Figure 7: Examples of decryption for spectrograms encrypted by ROM under $M = 3$.



(a) Original

(b) Encrypted spectrogram ($M = 3$)

(c) Phase reconstruction ($M = 3$)

Figure 8: Example of phase reconstruction applied to encrypted spectrogram by ROM under $M = 3$.

In addition, comparing the waveforms shown in the upper rows of Figures 8(a) and 8(c), it can be seen that the original waveforms were not reconstructed at all. Therefore, it is confirmed that a spectrogram encrypted using the proposed methods can hardly reconstruct the original speech when the correct key is not known.

For the third experiment in the robustness evaluation, we analyzed how the proposed method behaves with incorrect keys in the ASV task. A violin plot of the distribution of EER when 1000 ROM secret keys were used as incorrect keys is shown in Figure 9. The EER for each model, when the speech was not encrypted, is represented by stars in Figure 9. The variance of EER for RawNeXt was 66.2, that for HuBERT5 was 24.1, and that for HuBERT10 was 2.51. Figure 9 shows that the smaller the block size used for encryption, the wider the EER distribution, and it also shown that the larger the block size, the larger the difference from the EER with the correct key. The reason why the distribution of RawNeXt's EER is biased toward low positions can be considered to be the small block size and the high generalization performance of the model. Furthermore, to investigate the characteristics of the generated keys, we measured the Euclidean distance between the speech encrypted with the correct key and the speech encrypted with incorrect keys. We presented the distributions for key groups with low EER and high EER in Figure 10. From this figure, we observed that keys with a low EER are not necessarily closer to the audio encrypted with the correct key. The characteristics of the secret keys should be investigated in the future.

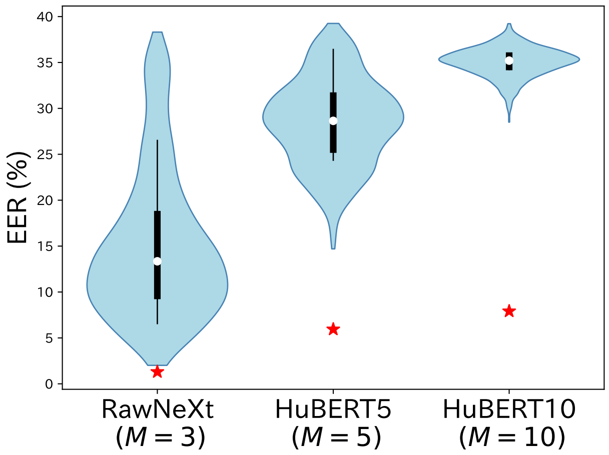


Figure 9: Distribution of EER (%) for speech waveforms encrypted using 1000 ROM secret keys and input to encrypted ASV model (asterisk: EER (%) for unencrypted ASV model)

For the fourth experiment in the robustness evaluation, we analyzed how ROM behaves with incorrect keys in the ASR task. Figure 11 shows the distribution of WER when speech waveforms encrypted with 100 ROM secret keys were input to the trained speech recognition model proposed in Karita *et al.* [7]. The red line in Figure 11 is the WER when plain speech was input to the plain model. As well as with the results of the third experiment, as

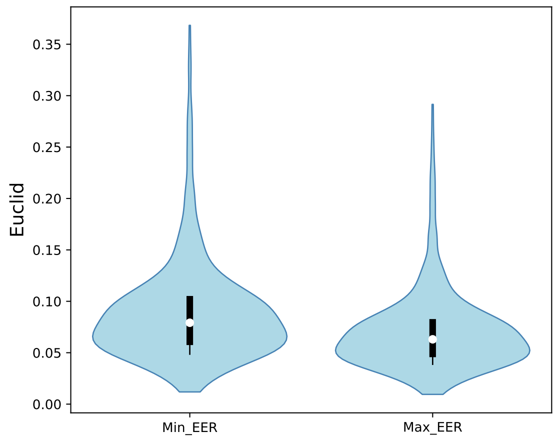


Figure 10: Distribution of Euclidean distance of speech waveforms encrypted with wrong ROM secret keys and speech waveforms encrypted with correct keys

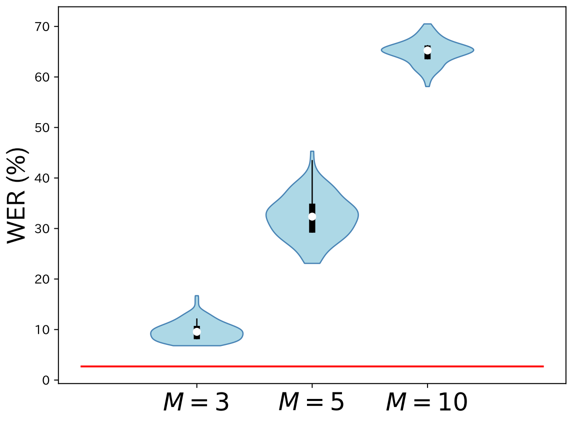


Figure 11: Distribution of WER(%) in case of speech waveforms encrypted using 100 ROM secret keys and input to ASR model [7] (line: WER (%) for unencrypted speech waveforms)

the value of M increased, the more the average WER rose, and the more the distribution became narrower. This implies that a larger value of M results in higher confidentiality for speech content. The ASV and ASR results confirm that the proposed methods provide stable privacy-preserving performance when the block size is large.

5 Conclusion

In this paper, we described privacy-preserving methods using secret keys based on Shuffling, Flipping, and ROM. The proposed methods can perform convolutional computation to cancel the effect of the orthogonal matrix secret key so that encrypted queries can be input to encrypted CNN models without decryption. In addition, when the user knows the correct key, there is no performance degradation at all. Experiments confirmed that, when using the proposed methods, users who do not know the secret key used to encrypt the model cannot use the model with high performance, and the larger the block size used for encryption, the more stable the privacy-preserving performance is. It was also confirmed that a third party who does not know the secret key cannot estimate or reconstruct the original speech data from the speech data encrypted using the proposed methods. For future work, we will develop an encryption method for speech data that is stable and robust against attacks, even when the block size is small, and we will investigate the effect of some noise reduction methods to confuse the inner product calculations. In addition, it is an important topic to expand our approach not only to CNN-based models but also to deep-learning models. As a challenging and crucial task, we also plan to explore research focusing on effectively concealing a part of critical components.

Acknowledgement

This work was supported in part by JSPS KAKENHI (Grant Number JP21H01327), JST CREST (Grant Number JPMJCR20D3) , and ROIS DS-JOINT (047RP2023) to S. Shiota.

References

[1] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”, In *Advances in Neural Information Processing Systems*, 33, 2020, 12449–60, ed. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

[2] J. S. Chung, A. Nagrani, and A. Zisserman, “VoxCeleb2: Deep Speaker Recognition”, In *Proc. Interspeech 2018*, 2018, 1086–90, DOI: 10.21437/Interspeech.2018-1929.

[3] G. Dekkers, S. Lauwereins, B. Thoen, M. W. Adhana, H. Brouckxon, B. Van den Bergh, T. Van Waterschoot, B. Vanrumste, M. Verhelst, and P. Karsmakers, “The SINS database for detection of daily activities in a home environment using an acoustic sensor network”, *Detection and Classification of Acoustic Scenes and Events 2017*, 2017, 1–5.

[4] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 2021, 3451–60, DOI: 10.1109/TASLP.2021.3122291.

[5] H. Kai, S. Takamichi, S. Shiota, and H. Kiya, “Lightweight and irreversible speech pseudonymization based on data-driven optimization of cascaded voice modification modules”, *Computer Speech & Language*, 72, 2022, 101315, ISSN: 0885-2308, DOI: https://doi.org/10.1016/j.csl.2021.101315, https://www.sciencedirect.com/science/article/pii/S0885230821001108.

[6] H. Kai, S. Takamichi, S. Shiota, and H. Kiya, “Robustness of Signal Processing-Based Pseudonymization Method Against Decryption Attack”, In *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, 287–93, DOI: 10.21437/Odyssey.2022-40.

[7] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, “A Comparative Study on Transformer vs RNN in Speech Applications”, In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, 449–56.

[8] J.-H. Kim, H.-J. Shim, J. Heo, and H.-J. Yu, “RawNeXt: Speaker Verification System For Variable-Duration Utterances With Deep Layer Aggregation And Extended Dynamic Scaling Policies”, In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, 7647–51, DOI: 10.1109/ICASSP43922.2022.9747594.

[9] H. Kiya, A. MaungMaung, Y. Kinoshita, S. Imaizumi, and S. Shiota, “An overview of compressible and learnable image transformation with secret key and its applications”, *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[10] A. Kong, K.-H. Cheung, D. Zhang, M. Kamel, and J. You, “An analysis of BioHashing and its variants”, *Pattern Recognition*, 39(7), 2006, 1359–68, ISSN: 0031-3203, DOI: https://doi.org/10.1016/j.patcog.2005.10.025, https://www.sciencedirect.com/science/article/pii/S0031320305004280.

[11] A. Lumini and L. Nanni, “An improved BioHashing for human authentication”, *Pattern Recognition*, 40(3), 2007, 1057–65, ISSN: 0031-3203, DOI: https://doi.org/10.1016/j.patcog.2006.05.030, https://www.sciencedirect.com/science/article/pii/S0031320306002482.

[12] A. Maungmaung and H. Kiya, “Privacy-Preserving Image Classification Using an Isotropic Network”, *IEEE MultiMedia*, 29(2), 2022, 23–33, DOI: 10.1109/MMUL.2022.3168441.

[13] A. Mesaros, T. Heittola, and T. Virtanen, “Acoustic scene classification: an overview of DCASE 2017 challenge entries”, 2018, 411–5.

[14] F. Mezzadri, “How to generate random matrices from the classical compact groups”, *Notices of the American Mathematical Society*, 54(5), 2007, 592–604.

[15] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, “Voxceleb: Large-scale speaker verification in the wild”, *Computer Speech & Language*, 60, 2020, 101027, ISSN: 0885-2308, DOI: https://doi.org/10.1016/j.csl.2019.101027, https://www.sciencedirect.com/science/article/pii/S0885230819302712.

[16] S. Niwa, S. Shiota, and H. Kiya, “A Privacy-Preserving Method Using Secret Key for Convolutional Neural Network-Based Speech Classification”, In *2023 31st European Signal Processing Conference (EUSIPCO)*, 2023, 76–80, DOI: 10.23919/EUSIPCO58844.2023.10289898.

[17] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”, In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[18] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books”, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, 5206–10, DOI: 10.1109/ICASSP.2015.7178964.

[19] Z. Průša, P. Balazs, and P. L. Søndergaard, “A Noniterative Method for Reconstruction of Phase From STFT Magnitude”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5), 2017, 1154–64.

[20] A. Singh and K. Chatterjee, “Cloud security issues and challenges: A survey”, *Journal of Network and Computer Applications*, 79, 2017, 88–115, ISSN: 1084-8045, DOI: https://doi.org/10.1016/j.jnca.2016.11.027, https://www.sciencedirect.com/science/article/pii/S1084804516302983.

[21] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-Vectors: Robust DNN Embeddings for Speaker Recognition”, *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, 5329–33, DOI: 10.1109/ICASSP.2018.8461375.

[22] H. Tabrizchi and M. Kuchaki Rafsanjani, “A survey on security challenges in cloud computing: issues, threats, and solutions”, *The journal of supercomputing*, 76(12), 2020, 9493–532.

[23] F. Teixeira, A. Abad, B. Raj, and I. Trancoso, “Towards End-to-End Private Automatic Speaker Recognition”, In *Proc. Interspeech 2022*, 2022, 2798–802, DOI: 10.21437/Interspeech.2022-10672.

[24] N. Tomashenko, X. Wang, X. Miao, H. Nourtel, P. Champion, M. Todisco, E. Vincent, N. Evans, J. Yamagishi, and J.-F. Bonastre, "The VoicePrivacy 2022 Challenge Evaluation Plan", [Online]. Available: https://www.voiceprivacychallenge.org/vp2020/docs/VoicePrivacy_2020_Eval_Plan_v1_4.pdf, 2020.

[25] A. Trockman and J. Z. Kolter, "Patches Are All You Need?", *Transactions on Machine Learning Research*, 2023, Featured Certification, ISSN: 2835-8856, https://openreview.net/forum?id=rAnB7JSMXL.

[26] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, and et al., "ESPnet: End-to-End Speech Processing Toolkit", In *Proc. INTERSPEECH 2018*, 2018, 2207–11, DOI: 10.21437/Interspeech.2018-1456.

[27] J. Williams, K. Pizzi, S. Das, and P.-G. Noé, "New challenges for content privacy in speech and audio", In *Proc. 2nd Symposium on Security and Privacy in Speech Communication*, 2022, 1–6, DOI: 10.21437/SPSC.2022-1.

[28] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K.-t. Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H.-y. Lee, "SUPERB: Speech Processing Universal PERformance Benchmark", In *Proc. Interspeech 2021*, 2021, 1194–8, DOI: 10.21437/Interspeech.2021-1775.

[29] S.-X. Zhang, Y. Gong, and D. Yu, "Encrypted Speech Recognition Using Deep Polynomial Networks", In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, 5691–5, DOI: 10.1109/ICASSP.2019.8683721.