

ORIGINAL PAPER

A protection method of trained CNN model with a secret key from unauthorized access

APRILPYONE MAUNGMAUNG  AND HITOSHI KIYA

In this paper, we propose a novel method for protecting convolutional neural network models with a secret key set so that unauthorized users without the correct key set cannot access trained models. The method enables us to protect not only from copyright infringement but also the functionality of a model from unauthorized access without any noticeable overhead. We introduce three block-wise transformations with a secret key set to generate learnable transformed images: pixel shuffling, negative/positive transformation, and format-preserving Feistel-based encryption. Protected models are trained by using transformed images. The results of experiments with the CIFAR and ImageNet datasets show that the performance of a protected model was close to that of non-protected models when the key set was correct, while the accuracy severely dropped when an incorrect key set was given. The protected model was also demonstrated to be robust against various attacks. Compared with the state-of-the-art model protection with passports, the proposed method does not have any additional layers in the network, and therefore, there is no overhead during training and inference processes.

Keywords: Block-wise transformation, model protection, model watermarking

Received 08 January 2021; Revised 31 May 2021

1. INTRODUCTION

Convolutional neural networks (CNNs) are a type of deep neural network (DNN) inspired by the human visual system. Recent advances in deep learning show that CNNs have led to major breakthroughs in computer vision [1]. Impressively, the last ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2017 proved that the image classification accuracy has surpassed the level of human performance (i.e. error rate of 2.25%). There is no doubt that CNNs have dominated visual recognition systems in many different applications.

However, training successful CNNs is very expensive because it requires a huge amount of data and fast computing resources (e.g. GPU-accelerated computing). For example, the ImageNet dataset contains about 1.2 million images, and training on such a dataset takes days and weeks even on GPU-accelerated machines. In fact, collecting images and labeling them will also consume a massive amount of resources. Moreover, algorithms used in training a CNN model may be patented or have restricted licenses. Therefore, trained CNNs have great business value. Considering the expenses necessary for the expertise, money, and time taken to train a CNN model, a model should be regarded as a kind of intellectual property. While distributing a trained

model, an illegal party may also obtain a model and use it for its own service.

To protect the copyrights of trained models, researchers have adopted digital watermarking technology to embed watermarks into the models [2–9]. These studies focus on identifying the ownership of a model in question. In reality, a stolen model can be directly used by an attacker without arousing suspicion. In addition, the stolen model can be exploited in many different ways such as through model inversion attacks [10] and adversarial attacks [11]. To the best of our knowledge, the consequences of a stolen model have not been considered before in model protection research except for ownership verification. In this paper, we focus on protecting a model from misuse when it has been stolen by taking inspiration from an adversarial defense.

Recently, a key-based adversarial defense was proposed to combat adversarial examples [12, 13], which was in turn inspired by perceptual image encryption methods, which were proposed for privacy-preserving machine learning [14] and encryption-then-compression systems [15–21]. The uniqueness of the key for the model in [12, 13] motivated us to use a key-based transformation technique for model protection.

Therefore, for the first time, in this paper, we propose a model protection method with a secret key set in such a way that a stolen model cannot be used without a key set. Specifically, the proposed method preprocesses input images with a secret key set and trains a model by using such preprocessed images. The preprocessing technique used in

Tokyo Metropolitan University, Tokyo, Japan

Corresponding author:

H. Kiya

Email: kiya@tmu.ac.jp

the proposed method is a low-cost block-wise operation. In addition, the proposed method does not modify the network, and therefore, there is no overhead for both training and inference time. In an experiment, the performance of a model protected by the proposed method is demonstrated not only to be close to that of a non-protected one when the key set is correct but also to significantly drop upon using an incorrect key set. We make the following contributions in this paper.

- We demonstrate block-wise image transformation with a secret key to be effective for model protection.
- We conduct extensive experiments on different datasets including ImageNet and carry out key estimation attacks and fine-tuning attacks.

The rest of this paper is structured as follows. Section II presents related work on conventional model watermarking, its problems and learnable image encryption (LIE). Section III puts forward the proposed model protection method. Experiments and results are presented in Section IV. Section V includes discussion and analysis, and Section VI concludes this paper. This paper is an extension of the work in [22].

II. RELATED WORK

We review the existing model watermarking schemes of image classifiers and discuss problems with them. In addition, we also overview learnable encryption methods by which the proposed method has been inspired.

A) Model watermarking

Digital watermarking technology is widely used to combat copyright infringement for multimedia data [23]. An owner embeds a watermark into multimedia content (such as images, audio, etc.). When the protected content is stolen, the embedded watermark is extracted and used to verify ownership. In a similar fashion, to prevent the illegal distribution of DNN models, digital watermarking techniques are used to embed watermarks into proprietary DNN models. There are mainly two scenarios in DNN model watermarking: white-box and black-box.

A model watermarking scenario in white-box settings requires access to model weights for embedding and extracting a watermark. Uchida et al. first proposed a white-box model-watermarking method [2]. A watermark is embedded in one or more layers of model weights by using “an embedding regularizer,” which is an additional regularization term in the loss function during training. Similarly, there are other studies that follow the use of an additional regularization term as in [4, 6, 8].

Extracting watermarks in white-box settings requires access to the model weights. To overcome this limitation, another model watermarking scenario for black-box settings was proposed, where an inspector observes the input and output of a model in doubt to verify the ownership of

the model. In the black-box scenario, adversarial examples are exploited as a backdoor trigger set [7, 9], or a set of training examples is utilized so that a watermark pattern can be extracted from the inference of a model by using a specific set of training examples [3–5]. Therefore, access to the model weights is not required to verify ownership in black-box settings.

The above-mentioned model-watermarking schemes focus on ownership verification. Thus, a stolen model can be directly used and exploited without arousing suspicion because the performance of a protected model (i.e. fidelity) is independent of the embedded watermark. In contrast, the proposed model protection is not a watermarking method, and it is more relevant to authorization or digital rights management because only the rightful user who has the correct key set can use a model to full capacity. Although frameworks for model watermarking and the proposed model protection are different, they are both necessary for dealing with digital rights management in different applications.

B) Model watermarking with passports

Fan et al. [4] pointed out that conventional ownership verification schemes are vulnerable against ambiguity attacks [24] where two watermarks can be extracted from the same protected model, causing confusion regarding ownership. Therefore, Fan et al. [4] introduced passports and passport layers, which allow us to verify ownership with the correct passports. However, the passport in [4] is a set of extracted features of a secret image/images or equivalent random patterns from a pre-trained model. In addition, a network has to be modified with additional passport layers to use passports. Therefore, there are significant overhead costs in both the training and inference phases, in addition to user-unfriendly management of lengthy passports in [4].

In this paper, we aim to protect a model by embedding a secret key with minimal impact on model performance. Similar to the study in [4], a correct key is required for correct inference. However, the proposed method does not introduce any overhead in training or inference. Ownership is automatically verified upon being given the correct key.

C) Learnable image encryption

LIE is to perceptually encrypt images to mainly protect visual information on plain images while maintaining the network ability to learn the encrypted ones for classification tasks. Conventional LIE methods are classified into two classes in terms of application: LIE for privacy-preserving deep learning [15, 16, 18, 25–28] and LIE for adversarial robustness [12, 13].

LIE methods for privacy-preserving have two requirements: protecting visual information and maintaining a high classification accuracy under the use of encrypted images. In a block-wise manner, a color image is divided into blocks, and each block is processed by using a series of encryption with a common key to all blocks [15] or

with different keys [25]. In a pixel-wise manner, negative/positive transformation to each pixel and color shuffling across three channels are exploited to produce learnable encrypted images [16, 18]. In contrast, a transformation network is trained in cooperation with a pre-trained classification model to generate images without visual information on plain images. One such study utilized a generative adversarial network [26]. To improve classification accuracy and robustness against various attacks, transformation networks have been proposed that use U-Net as in [27, 28].

LIE methods for adversarial defenses [12, 13] have three requirements: a high classification accuracy, robustness against adversarial attacks, and resistance to key estimation attacks. The methods in this class do not aim to protect the visual information of plain images. Instead, a key is used to control the model's decision. In this paper, we do not propose a new encryption method. We adopt the methods in [12, 13] for a new application, model protection, for the first time. The proposed model protection method is carried out on the basis of LIE methods for adversarial defenses [12, 13], but hyperparameters are carefully tuned for model protection purposes. LIE methods have never been applied to model protection applications. The contribution in this paper is to introduce some conventional image encryption algorithms into a model protection task.

III. PROPOSED MODEL-PROTECTION METHOD

A) Notation

The following notations are utilized throughout this paper:

- w , h , and c are used to denote the width, height, and number of channels of an image.
- The tensor $x \in [0, 1]^{c \times h \times w}$ represents an input color image.
- The tensor $x' \in [0, 1]^{c \times h \times w}$ represents a transformed image.
- M is the block size of an image.
- Tensors $x_b, x'_b \in [0, 1]^{h_b \times w_b \times p_b}$ are a block image and a transformed block image, respectively, where $w_b = w/M$ is the number of blocks across width w , $h_b = h/M$ is the number of blocks across height h , and $p_b = M \times M \times c$ is the number of pixels in a block.
- A pixel value in a block image (x_b or x'_b) is denoted by $x_b(i, j, k)$ or $x'_b(i, j, k)$, where $i \in \{0, \dots, h_b - 1\}$, $j \in \{0, \dots, w_b - 1\}$, and $k \in \{0, \dots, p_b - 1\}$ are indices corresponding to the dimension of x_b or x'_b .
- B is a block of an image, and its dimension is $M \times M \times c$.
- \hat{B} is a flattened version of block B , and its dimension is $1 \times 1 \times p_b$.
- K denotes a set of keys.
- A password required for format-preserving encryption, which refers to encrypting in such a way that the output is in the same format as its input, is denoted as *password*.

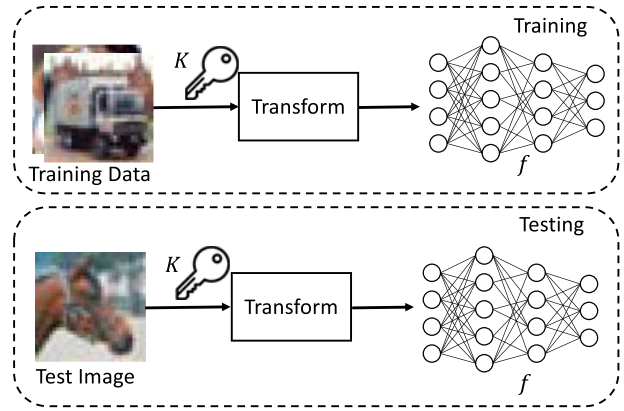


Fig. 1. Overview of image classification with proposed model protection method.

- $\text{Enc}(n, \text{password})$ denotes format-preserving Feistel-based encryption (FFX) [29] with a length of 3, where n is an integer (used only in FFX encryption).
- An image classifier is denoted as $f(\cdot)$.

B) Requirements of proposed scheme

We consider a model protection scenario that aims to fulfill the following requirements:

- Usability: A rightful user with key set K can access a model without any noticeable overhead in both training and inference time, and performance degradation. The key management should be easy.
- Unusability: Ideally, stolen models should not be usable in any case without key set K . In addition, even when the adversary retrains a stolen model with a forged key set, the performance of the model should heavily drop.

C) Overview

An overview of image classification with the proposed method is depicted in Fig. 1. In the proposed model protection, input images are transformed by using secret key set K before training or testing a model. Model f is trained by using the transformed images. To test a trained model, test images are also transformed with the same key set K before testing.

The block-wise transformation consists of three steps: block segmentation, block-wise transformation, and block integration (see Fig. 2). The process of the block-wise transformation is shown as follows.

- Block segmentation:** The process of block segmentation is illustrated in Fig. 2.
 - Step 1: An input image x is divided into blocks such that $\{B_{11}, B_{12}, \dots, B_{h_b, w_b}\}$.
 - Step 2: Each block in x is flattened to obtain $\{\hat{B}_{11}, \hat{B}_{12}, \dots, \hat{B}_{h_b, w_b}\}$.
 - Step 3: The flattened blocks are concatenated in such a way that the relative spatial location among blocks in x_b is the same as that among blocks in x .

- (ii) **Block-wise transformation:** Given a key set K , x_b is transformed by using a block-wise transformation algorithm, $t(x_b, K)$. The transformed block image is written as

$$x'_b = t(x_b, K). \quad (1)$$

- (iii) **Block Integration:** The transformed blocks in x'_b are integrated back to the original dimension (i.e. $c \times h \times w$) in the reverse order to the block segmentation process for obtaining a transformed image x' .

D) Block-wise transformation with secret key

We introduce three block-wise transformations: pixel shuffling (SHF), negative/positive transformation (NP), and FFX for realizing $t(x_b, K)$. We selected these three transformations from among LIE methods for model protection applications because the requirements of LIE methods for these applications are expected to be similar to those for adversarial defenses.

A set of keys K consists of one or more keys depending on the desired number of transformations. For example, if one transformation (SHF) is used, $K = \{\alpha\}$, if two (SHF and NP) are used, $K = \{\alpha, \beta\}$, and if three (SHF, NP, and FFX) are used, $K = \{\alpha, \beta, \gamma\}$, where α is for SHF, β is for NP, and γ is for FFX, respectively.

Key α is a permutation vector, and it is defined as

$$\alpha = [\alpha_{1, \cdot}, \alpha_{k, \cdot}, \alpha_{k', \cdot}, \dots, \alpha_{p_b, \cdot}], \quad \alpha_k \in \{1, \dots, p_b\}, \quad (2)$$

where $\alpha_k \neq \alpha_{k'}$ if $k \neq k'$.

Key β is a binary vector, and it is given by

$$\beta = [\beta_1, \dots, \beta_k, \dots, \beta_{p_b}], \quad \beta_k \in \{0, 1\}, \quad (3)$$

where the value of the occurrence probability $P(\beta_k)$ is 0.5.

Key γ is defined as

$$\gamma = [\gamma_1, \dots, \gamma_k, \dots, \gamma_{p_b}], \quad \gamma_k \in \{0, 1\}, \quad (4)$$

where the value of the occurrence probability $P(\gamma_k)$ is 0.5.

When SHF is used, x'_b is obtained as:

$$x'_b(i, j, \alpha_k) = x_b(i, j, k). \quad (5)$$

When NP is used, every pixel value in x_b needs to be at 255 scale with 8 bits (i.e. multiply x_b by 255), and x'_b is obtained as:

$$x'_b(i, j, k) = \begin{cases} x_b(i, j, k) & (\beta_k = 0) \\ x_b(i, j, k) \oplus (2^L - 1) & (\beta_k = 1), \end{cases} \quad (6)$$

where \oplus is an exclusive or (XOR) operation, L is the number of bits used in $x_b(i, j, k)$, and $L = 8$ is used in this paper. After the transformation, every pixel value in x'_b is converted back to $[0, 1]$ scale (i.e. divide x'_b by 255).

When FFX is used, every pixel value in x_b also needs to be at 255 scale with 8 bits (i.e. multiply x_b by 255). In addition,

FFX also requires a *password* for FFX [29], and x'_b is obtained as:

$$x'_b(i, j, k) = \begin{cases} x_b(i, j, k) & (\gamma_k = 0) \\ \text{Enc}(x_b(i, j, k), \text{password}) & (\gamma_k = 1). \end{cases} \quad (7)$$

Note that FFX [29] takes an integer value and outputs an integer; therefore, pixel values should be at $[0, 255]$ scale. The pixel value $x_b(i, j, k) \in \{0, 1, \dots, 254, 255\}$ is encrypted by FFX with a length of 3 digits to cover the whole range from 0 to 255. Therefore, FFX transforms each pixel with an integer value of ($[0, 255]$ scale) into a pixel with an integer value of ($[0, 999]$ scale), preserving the integer format. Another notable thing is that a *password* for FFX can be arbitrary, and the only important thing is the location of the encrypted pixels that are determined by the key γ .

The overall block-wise transformation is detailed in Algorithm 1. An example of images transformed by different transformations is shown in Fig. 3. The three transformations (SHF, NP, and FFX) were confirmed to have different performances in terms of classification accuracy and key estimation attack in [12, 13], so these transformations are compared again under model protection in this paper. In particular, parameter M affects the classification accuracy and resistance to key estimation attacks.

Algorithm 1 Block-wise transformation with secret key

Input: x, K

Output: x'

- 1: Divide x into blocks, $\{B_{11}, \dots, B_{h_b, w_b}\}$
 - 2: Flatten blocks to vectors, $\{\hat{B}_{11}, \dots, \hat{B}_{h_b, w_b}\}$
 - 3: Concatenate flattened blocks to obtain x_b
 - 4: // To transform x_b given K
 - 5: **if** SHF **then**
 - 6: $x'_b \leftarrow x_b[:, :, \alpha]$
 - 7: **end if**
 - 8: **if** NP **then**
 - 9: // Make pixel values be at 255 scale
 - 10: $x_b \leftarrow x_b \cdot 255$
 - 11: $x'_b[:, :, \beta] \leftarrow 255 - x_b[:, :, \beta]$
 - 12: $x'_b \leftarrow x'_b / 255$
 - 13: **end if**
 - 14: **if** FFX **then**
 - 15: // Make pixel values be at 255 scale
 - 16: $x_b \leftarrow x_b \cdot 255$
 - 17: $x'_b[:, :, \gamma] \leftarrow \text{Enc}(x_b[:, :, \gamma], \text{password})$
 - 18: $\max \leftarrow$ the maximum value of the encryption
 - 19: $x'_b \leftarrow x'_b / \max$
 - 20: **end if**
 - 21: $x' \leftarrow$ Integrate blocks in x'_b
-

E) Robustness against attacks

A threat model includes a set of assumptions such as an attacker's goals, knowledge, and capabilities. An attacker

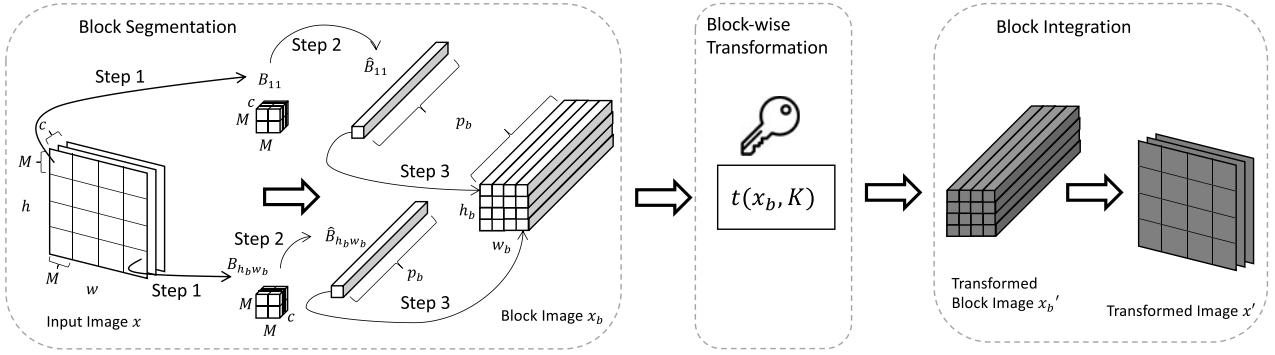


Fig. 2. Process of block-wise transformation.

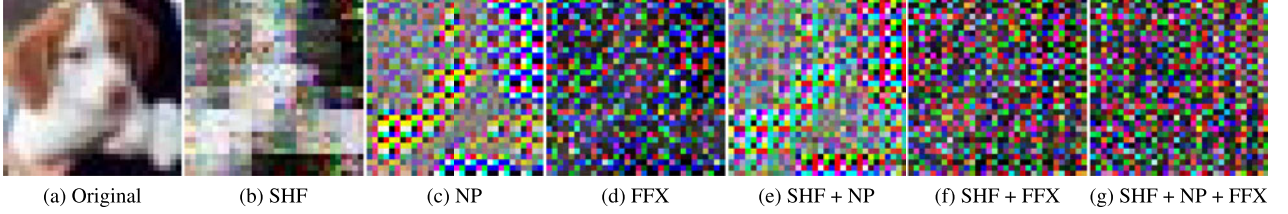


Fig. 3. Example of block-wise transformed images ($M = 4$) with key set K . (a) Original. (b) SHF. (c) NP. (d) FFX. (e) SHF + NP. (f) SHF + FFX. (g) SHF + NP + FFX.

may steal a model to achieve different goals. In this paper, we consider the attacker's goal to be to make use of a stolen model by estimating a key set or fine-tuning the stolen model for different purposes. In this regard, we assume that the attacker knows the transformation details such as the block size and type of transformation, and a small subset of the training dataset. Therefore, the attacker may observe the accuracy of his or her test dataset to estimate a key set or fine-tune the stolen model. We carry out the following possible attacks with the intent of stealing a model to evaluate the robustness of the proposed method. In experiments, the proposed method will be demonstrated to be robust against attacks.

1) KEY ESTIMATION ATTACK

We consider a scenario where a model is stolen and transformation details are known except the secret key. The key may be estimated by brute-force checking possible keys. The key space \mathcal{K} of each transformation is given by

$$\mathcal{K}_\alpha(c \times M \times M) = (c \times M \times M)! \quad (\text{SHF}), \quad (8)$$

$$\mathcal{K}_\beta(c \times M \times M) = 2^{(c \times M \times M)} \quad (\text{NP}), \text{ and} \quad (9)$$

$$\mathcal{K}_\gamma(c \times M \times M) = 2^{(c \times M \times M)} \quad (\text{FFX}). \quad (10)$$

Therefore, the key space will vary with respect to block size M and the type of block-wise transformation used for protecting a model.

The attacker may estimate the key heuristically by observing the accuracy over a batch of images. Algorithm 2 describes the process of estimating key set K . First, we randomly initialize key set $K' = \{\alpha', \beta', \gamma'\}$ in accordance with the transformation. Next, we also initialize a set of index pairs \mathcal{P} as $\mathcal{P} = \{(1, 2), (1, 3), \dots, (c \times M \times M - 1, c \times M \times$

$M)\}$ for keys $\alpha', \beta',$ or γ' . The number of all possible combinations of pairs for each key can be computed as a binomial coefficient given by

$$|\mathcal{P}| = {}_n C_r = \frac{n!}{r!(n-r)!}, \quad (11)$$

where $n = c \times M \times M$, and $r = 2$. For each index pair, we swap the pair in $\alpha', \beta',$ or γ' if the swap improves the accuracy as shown in Algorithm 2.

Key estimation attacks do not guarantee that the attacker will find the correct key because the attacker does not know the actual performance of the correct key. However, the attacker may perform fine-tuning attacks to exploit a stolen model as below.

2) FINE-TUNING ATTACK WITH INCORRECT KEY SET AND SMALL DATASET

Fine-tuning (transfer learning) [30] is to train a model on top of pre-trained weights. Since fine-tuning alters the weights of the model, an attacker may use fine-tuning as an attack to overwrite a protected model with the intent of forging keys. The goal of this attack is to replace the key set with a different key set by retraining a protected model with a small subset of a dataset. We can consider such an attack scenario where the adversary has a subset of dataset D' and retrains the model with a forged key set (K').

3) FINE-TUNING ATTACK WITH NEW DATASET

We assume an attacker may steal a protected model and fine-tune the model with a new dataset. The goal of this attack is to replace a protected model with an unprotected one without any key by using transfer learning.

In practice, CNNs are not trained from the beginning with random weights because creating a large dataset like

Algorithm 2 Key estimation**Input:** Input images with labels**Output:** K'

```

1: Initialize  $K' = \{\alpha', \beta', \gamma'\}$  in accordance with the trans-
   formation
2: Initialize  $\mathcal{P} = \{(1, 2), (1, 3), \dots, (i, j), \dots, (c \times M \times$ 
    $M - 1, c \times M \times M)\}$ 
3: accuracy  $\leftarrow$  Calculate accuracy of input images
4: for Each index pair  $(i, j)$  in  $\mathcal{P}$  do
5:   if SHF then
6:      $(\alpha'_i, \alpha'_j) \leftarrow (\alpha'_i, \alpha'_j)$ 
7:     current_accuracy  $\leftarrow$  Calculate accuracy of input
       images
8:     if current_accuracy > accuracy then
9:       accuracy  $\leftarrow$  current_accuracy
10:    else
11:       $(\alpha'_i, \alpha'_j) \leftarrow (\alpha'_j, \alpha'_i)$ 
12:    end if
13:  end if
14:  if NP then
15:     $(\beta'_i, \beta'_j) \leftarrow (\beta'_i, \beta'_j)$ 
16:    current_accuracy  $\leftarrow$  Calculate accuracy of input
       images
17:    if current_accuracy > accuracy then
18:      accuracy  $\leftarrow$  current_accuracy
19:    else
20:       $(\beta'_i, \beta'_j) \leftarrow (\beta'_j, \beta'_i)$ 
21:    end if
22:  end if
23:  if FFX then
24:    Swap  $\gamma'_i$  with  $\gamma'_j$  in key  $\gamma'$ 
25:     $(\gamma'_i, \gamma'_j) \leftarrow (\gamma'_j, \gamma'_i)$ 
26:    current_accuracy  $\leftarrow$  Calculate accuracy of input
       images
27:    if current_accuracy > accuracy then
28:      accuracy  $\leftarrow$  current_accuracy
29:    else
30:       $(\gamma'_i, \gamma'_j) \leftarrow (\gamma'_j, \gamma'_i)$ 
31:    end if
32:  end if
33: end for

```

ImageNet is difficult and expensive. Therefore, CNNs are usually pre-trained with a larger dataset (e.g. ImageNet), known as transfer learning [30]. There are two major transfer-learning scenarios:

- **Fixed CNN:** A pre-trained CNN model is used as a fixed feature extractor, and the last fully connected layer is replaced with a targeted number of classes. In other words, convolutional layers are frozen, and only the last fully connected layer is trained with random initialization from scratch.
- **Fine-tuned CNN:** In this scenario, the CNN is fine-tuned from a pre-trained model. Here, it is possible that some convolutional layers can be fixed or the whole CNN is fine-tuned.

IV. EXPERIMENTS AND RESULTS

To verify the effectiveness of the proposed model protection method, we ran a number of experiments on different datasets. All the experiments were carried out in PyTorch [31] platform.

A) Datasets

We conducted image classification experiments on datasets with different scales, namely, the CIFAR (both 10 and 100 classes) [32] and ImageNet [33] datasets.

For the CIFAR-10 and CIFAR-100 datasets, we used a batch size of 128 and live augmentation (random cropping with padding of 4 and random horizontal flip) on training sets. Both datasets consist of 60 000 color images (dimensions of $32 \times 32 \times 3$) where 50 000 images are for training and 10 000 for testing. There are 10 classes (6000 images for each class) for the CIFAR-10 dataset and 100 classes (600 images for each class) for the CIFAR-100 dataset.

ImageNet comprises 1.28 million color images for training and 50 000 color images for validation. We progressively resized images during training, starting with larger batches of smaller images to smaller batches of larger images. We adapted three phases of training from the DAWN Bench top submissions as mentioned in [34]. Phases 1 and 2 resized images to 160 and 352 pixels, respectively, and phase 3 used the entire image size from the training set. The augmentation methods used in the experiment were random resizing, cropping (sizes of 128, 224, and 288, respectively, for each phase), and random horizontal flip.

B) Networks

We utilized deep residual networks [35] with 18 layers (ResNet18) and trained for 200 epochs with cyclic learning rates [36] and mixed precision training [37] for the CIFAR datasets. The parameters of the stochastic gradient descent (SGD) optimizer were a momentum of 0.9, weight decay of 0.0005, and maximum learning rate of 0.2. For the ImageNet dataset, we used ResNet50 with pre-trained weights. We adapted the training settings from [34] with the removal of weight decay regularization from batch normalization layers. The network was trained for 15 epochs in total for the ImageNet dataset.

C) Classification performance

We trained protected models by using images transformed by various transformations (both single and combined transformations) with different block sizes (i.e. $M \in \{2, 4, 8, 16\}$) on three different datasets (CIFAR-10, CIFAR-100, and ImageNet). The models are named after the shorthand of the respective transformations. For example, the model trained by using images transformed by SHF transformation is denoted as SHE, that by NP transformation as NP, and so on. We tested the protected models under three

Table 1. Accuracy (%) of protected models and baseline model for three datasets

| | Model | CIFAR-10 | | | CIFAR-100 | | | ImageNet | | |
|----------|----------------|-----------------------|----------------|--------------|-----------------------|----------------|-------------|-----------------------|----------------|-------------|
| | | Correct (K) | Incorrect (K') | Plain | Correct (K) | Incorrect (K') | Plain | Correct (K) | Incorrect (K') | Plain |
| $M = 2$ | SHF | 94.76 | 36.36 | 31.43 | 77.03 | 8.6 | 6.09 | 73.00 | 46.57 | 40.35 |
| | NP | 95.32 | 18.44 | 13.91 | 77.88 | 2.19 | 1.19 | 73.04 | 6.53 | 0.98 |
| | FFX | 93.80 | 15.69 | 38.84 | 74.36 | 3.41 | 11.50 | 72.43 | 0.12 | 0.23 |
| | SHF + NP | 94.50 | 20.65 | 11.79 | 76.76 | 2.52 | 1.49 | 72.90 | 4.87 | 1.12 |
| | SHF + FFX | 93.02 | 15.3 | 19.60 | 73.17 | 2.77 | 5.24 | 72.30 | 0.47 | 0.19 |
| | SHF + NP + FFX | 92.82 | 14.03 | 18.69 | 73.08 | 1.46 | 1.94 | 71.96 | 0.16 | 0.18 |
| $M = 4$ | SHF | 92.58 | 20.23 | 27.77 | 72.05 | 4.9 | 5.85 | 72.41 | 13.06 | 32.98 |
| | NP | 93.41 | 12.67 | 12.17 | 73.11 | 1.32 | 1.62 | 72.63 | 0.68 | 0.36 |
| | FFX | 92.29 | 18.38 | 37.06 | 69.92 | 3.9 | 12.73 | 72.17 | 0.15 | 0.15 |
| | SHF + NP | 92.37 | 12.11 | 12.35 | 71.27 | 1.35 | 1.93 | 72.15 | 0.21 | 0.25 |
| | SHF + FFX | 90.71 | 12.31 | 20.75 | 68.48 | 1.85 | 4.16 | 71.96 | 0.14 | 0.17 |
| | SHF + NP + FFX | 90.50 | 10.6 | 13.10 | 68.20 | 1.09 | 1.53 | 71.68 | 0.12 | 0.16 |
| $M = 8$ | SHF | 86.40 | 17.0 | 14.42 | 62.18 | 2.2 | 2.87 | 70.85 | 1.25 | 11.74 |
| | NP | 91.54 | 71.35 | 79.51 | 67.07 | 1.8 | 1.65 | 71.83 | 0.26 | 0.12 |
| | FFX | 92.00 | 47.07 | 37.25 | 69.60 | 9.66 | 11.12 | 71.46 | 0.3 | 0.09 |
| | SHF + NP | 86.47 | 12.16 | 14.75 | 62.75 | 1.42 | 1.56 | 71.14 | 0.19 | 0.86 |
| | SHF + FFX | 86.01 | 11.81 | 15.20 | 61.37 | 1.29 | 1.70 | 70.77 | 0.11 | 0.14 |
| | SHF + NP + FFX | 85.49 | 10.23 | 10.31 | 60.96 | 1.06 | 1.02 | 70.18 | 0.1 | 0.11 |
| $M = 16$ | SHF | 77.24 | 10.57 | 13.36 | 50.87 | 1.39 | 1.36 | 67.03 | 0.23 | 4.22 |
| | NP | 92.68 | 88.27 | 89.00 | 70.68 | 48.17 | 47.38 | 70.19 | 0.97 | 5.52 |
| | FFX | 91.38 | 72.91 | 29.35 | 69.78 | 34.92 | 9.19 | 69.24 | 2.07 | 0.14 |
| | SHF + NP | 77.52 | 10.66 | 11.70 | 50.33 | 1.0 | 1.02 | 67.50 | 0.11 | 0.18 |
| | SHF + FFX | 76.28 | 10.2 | 12.79 | 49.53 | 1.21 | 1.45 | 63.75 | 0.16 | 0.13 |
| | SHF + NP + FFX | 75.78 | 10.0 | 9.92 | 49.63 | 1.0 | 1.04 | 63.43 | 0.09 | 0.12 |
| | Baseline | 95.45 (not protected) | | | 77.67 (not protected) | | | 73.70 (not protected) | | |

Best results are given in **bold**.

conditions: with correct key set K , with incorrect key set K' , and with plain images (without any transformation).

In the experiments, the correct keys were generated by using a random number generator from the PyTorch platform [31] with a seed value of 42 (64-bit integer), and we used a publicly available library for FFX (pyffx [38]) with the password string "password." The incorrect keys were also generated by using the same random number generator from the PyTorch platform [31] with 1000 random seed values (64-bit integer).

Table 1 summarizes the simulation results for all three datasets, where the classification accuracy for incorrect key set K' was averaged over 1000 random key sets.

SHF: for SHF, as block size M was increased, the classification accuracy decreased when using correct key set K . Under the use of incorrect key set K' or plain images, the accuracy significantly dropped as block size M was increased, suggesting resistance against unauthorized access. Therefore, the selection of M controls the trade-off between classification accuracy and resistance against illegal usage.

NP, FFX: for NP, the highest accuracy values were achieved for all transformations on all the datasets for each block size with correct key set K . When using $M = 8$ or 16 on the CIFAR-10 and CIFAR-100 datasets, NP had the highest accuracy, and FFX had the second highest accuracy under the use of incorrect key set K' or plain images except for $M = 8$ on the CIFAR-100 dataset.

In contrast, on ImageNet, the accuracy was significantly low for both NP and FFX under the use of incorrect key set K' or plain images because the size of the images in the ImageNet dataset is larger than that of CIFAR-10 and CIFAR-100.

Combined: the models with combined transformations such as SHF + NP and SHF + FFX decreased the classification accuracy, compared with those with NP and FFX, when using $M = 8$ or 16. Under the use of incorrect key set K' or plain images, the accuracy severely dropped for all block sizes. The advantage of a combined transformation is that it can increase the key space, but it slightly reduces the classification accuracy.

From the empirical results, generally, the performance of an incorrect key set depends on the number of classes in a dataset. When using a dataset with a large number of classes such as CIFAR-100 and ImageNet, the accuracy of the incorrect key set was low due to difficulty in classifying images transformed by using an incorrect key set or plain images. In summary, the proposed method had a high classification accuracy (i.e. close to baseline accuracy) when correct key set K was given. In contrast, the accuracy deteriorated significantly when using incorrect key set K' or plain images. Since models with $M = 4$ provided a good trade-off between classification accuracy and resistance against unauthorized access, we focused on $M = 4$ in the following sections to further evaluate against attacks on the CIFAR-10 and CIFAR-100 datasets.

Table 2. Accuracy (%) of protected models ($M = 4$) under use of estimated key set K'

| Model | CIFAR-10 | CIFAR-100 |
|----------------|--------------------|--------------------|
| | Estimated (K') | Estimated (K') |
| SHF | 25.66 | 10.20 |
| NP | 37.44 | 8.78 |
| FFX | 80.97 | 49.18 |
| SHF + NP | 14.53 | 2.78 |
| SHF + FFX | 15.04 | 2.36 |
| SHF + NP + FFX | 11.00 | 1.47 |

D) Robustness against key estimation attack

The proposed method was evaluated in a scenario involving a key estimation attack in accordance with Algorithm 2. As described in Section III-E-1), elements in each key were rearranged in accordance with the improvement in accuracy, and the resulting estimated key set K' was used to evaluate the performance of the protected models.

Table 2 captures the classification performance of the models under the use of estimated key set K' on the CIFAR-10 and CIFAR-100 datasets. The estimated keys were not good enough to provide a reasonable accuracy except for FFX, which replaces a pixel value with a random value by using a password, so the encrypted pixel value contains almost no information. Therefore, the location of the un-encrypted pixel values plays an important role in the model’s decision-making process, as the encrypted pixel values are not important. This property helps an attacker to effectively find a good key when performing key estimation attacks (Algorithm 2). In contrast, in the other two transformations (SHF and NP), transformed pixel values have some information, and both positions of un-encrypted pixels and pixel values are important. Therefore, the indication to search for a good key was difficult for SHF and NP compared to FFX.

E) Robustness against fine-tuning attack with incorrect key and small dataset

We ran an experiment with different sizes for the adversary’s dataset (i.e. $|D'| \in \{100, 500, 1000, 10\ 000\}$) for the CIFAR-10 and CIFAR-100 datasets. We retrained the models with D' for 30 epochs. Table 3 shows the results of fine-tuning attacks for both datasets. Table 1 shows the performance of the incorrect key set before the model weights were modified. In contrast, fine-tuning with $|D'| = 100$ modified the

Table 4. Accuracy (%) of protected models under fine-tuning attacks with new dataset (CIFAR-100 to CIFAR-10) for fixed CNN and fine-tuned CNN

| Model | Fixed CNN | Fine-tuned CNN |
|----------------|-----------|----------------|
| | SHF | 44.14 |
| NP | 36.70 | 83.52 |
| FFX | 52.29 | 86.18 |
| SHF + NP | 37.67 | 82.40 |
| SHF + FFX | 37.00 | 79.67 |
| SHF + NP + FFX | 26.47 | 71.10 |
| Plain | 72.85 | 90.60 |

weights. Therefore, the models in Table 1 are different from those in Table 3, so the accuracy was not equal to that of the incorrect key set in Table 1.

Although the accuracy improved with respect to the size of the adversary’s dataset, it was still lower than the performance of the correct key set K as presented in Table 1. Therefore, the results show that the compromised models were not as good as the original models, even when fine-tuning with a small subset of a dataset. As a result, the attacker is not able to use the model to full capacity, suggesting robustness against this type of attack.

F) Robustness against fine-tuning attack with new dataset

We simulated this attack scenario by fine-tuning the CIFAR-100 to the CIFAR-10 under both conditions (fixed CNN and fine-tuned CNN). We fine-tuned the CIFAR-100 model for 25 epochs. The parameters of the SGD optimizer were a learning rate of 0.001 and a momentum of 0.9, and a StepLR scheduler was used with a step size of 7 and a gamma value of 0.1.

Table 4 shows the results of fine-tuning the CIFAR-100 dataset to the CIFAR-10 dataset. Training only the last layer (fixed CNN) did not provide good accuracy even for the non-protected models (plain). However, for “fine-tuned CNN,” the non-protected (plain) model was fine-tuned to an accuracy of 90.60%, which was closer to the baseline accuracy (i.e. 95.45%). In contrast, the fine-tuned accuracies of the protected models were lower (83.46% for SHF, 83.52% for NP, and 86.18% for FFX) than that of the plain model. For the fine-tuned CNN, the results show that the protected models were still transferable, although this was not as good as fine-tuning from the non-protected models.

Table 3. Accuracy (%) of protected models under fine-tuning attacks with incorrect key and small dataset

| Model | CIFAR-10 | | | | CIFAR-100 | | | |
|----------------|--------------|--------------|---------------|----------------|--------------|--------------|---------------|----------------|
| | $ D' = 100$ | $ D' = 500$ | $ D' = 1000$ | $ D' = 10000$ | $ D' = 100$ | $ D' = 500$ | $ D' = 1000$ | $ D' = 10000$ |
| SHF | 12.69 | 38.33 | 46.73 | 86.31 | 2.65 | 8.52 | 12.30 | 58.99 |
| NP | 10.57 | 37.25 | 47.41 | 87.06 | 3.00 | 8.19 | 11.57 | 60.70 |
| FFX | 10.15 | 32.30 | 40.52 | 86.04 | 1.00 | 6.58 | 9.62 | 58.64 |
| SHF + NP | 14.03 | 37.50 | 46.36 | 85.37 | 2.70 | 7.88 | 11.77 | 57.80 |
| SHF + FFX | 12.54 | 46.28 | 55.11 | 83.74 | 2.54 | 18.21 | 29.83 | 56.14 |
| SHF + NP + FFX | 11.15 | 39.59 | 48.13 | 83.27 | 3.23 | 12.09 | 24.14 | 54.51 |

Table 5. Comparison of proposed protected model NP and state-of-the-art passport-protected model in terms of classification accuracy (%) for CIFAR datasets

| Model | Correct K /Passports | Estimated K' /Passports | Training overhead | Inference overhead | Network modification | Key management |
|----------------------------|---------------------------|------------------------------|-------------------|--------------------|-------------------------|----------------|
| CIFAR-10 | | | | | | |
| NP (Proposed) | 93.41 | 37.44 | Negligible | Negligible | No | Easy |
| Scheme \mathcal{V}_1 [4] | 94.62 | 70.00 | 15–30% [4] | 10% [4] | Yes | Difficult |
| CIFAR-100 | | | | | | |
| NP (Proposed) | 73.10 | 8.78 | Negligible | Negligible | No | Easy |
| Scheme \mathcal{V}_1 [4] | 75.52 | 35.00 | 15–30% [4] | 10% [4] | Yes | Difficult |

G) Comparison with state-of-the-art methods

Both watermarking and encryption algorithms can be used to protect the copyright of digital products, but the former is imperceptible, while the latter is a means of direct encryption. It is difficult to compare both approaches in terms of their aims and robustness against attacks. In particular, in conventional model watermarking methods, the embedded watermark is independent of model performance. Therefore, we compared the proposed protected model (NP) with the state-of-the-art passport protected model, Scheme \mathcal{V}_1 [4] in terms of classification accuracy with/without correct key/passports, overheads, network modification and key management, for both the CIFAR-10 and CIFAR-100 datasets. Scheme \mathcal{V}_1 [4] was not trained and tested using the same settings as the proposed method because the network in \mathcal{V}_1 was modified with passport layers and the hyperparameters were based on the modified network. In contrast, the proposed method used a standard ResNet18 and was trained with cyclic learning rates [36] and mixed precision training [37].

CIFAR-10: In terms of accuracy when the correct key/passport was given, the accuracy of \mathcal{V}_1 was slightly higher than that of NP at 1.21%. However, it was confirmed that if block size $M = 2$ was used, NP achieved higher accuracy than \mathcal{V}_1 (i.e. 95.32%). When estimated incorrect key set was given, the accuracy of NP significantly dropped. In contrast, when reverse-engineered (i.e. estimated) passports were used, the accuracy of \mathcal{V}_1 was high (70%).

CIFAR-100: Similarly, for CIFAR-100, the accuracy of \mathcal{V}_1 was also slightly higher. However, when an estimated key was given, the proposed method was more resistant than \mathcal{V}_1 .

In terms of overhead, \mathcal{V}_1 modifies a network with additional passport layers; therefore, it introduces a training and inference overhead for both datasets. The overheads in [4] are based on the relative recorded time taken as mentioned in the paper by the original authors. In contrast, the proposed model NP does not have any noticeable overhead, and there is no modification in the network for both datasets [4]. Moreover, the block-wise transformation in the proposed model protection can be efficiently implemented with vectorized operations; therefore, pre-processing with the block-wise transformation does not cause any noticeable overheads in both training and testing. From a key management perspective, \mathcal{V}_1 requires a trained model to generate passports, and the proposed model NP does not need

any model to generate keys. Therefore, the key management of the proposed method is simple and straightforward (Table 5).

V. ANALYSIS AND DISCUSSION

In this section, we analyze the block-wise transformations utilized in the proposed model-protection method in terms of pixel correlation and key sensitivity. Then, we discuss possible key improvement, the selection of transformations, and the application range of the proposed scheme.

A) Image correlation analysis

To gain insights into the classification performance of block-wise transformed images, we carried out adjacent pixel correlation tests on a test image, “dog,” in the horizontal, vertical, and diagonal directions as shown in Fig. 4. From the figure, all transformations were confirmed to maintain some correlation between pixels differently. The pixel correlation distribution of the image transformed by SHF was similar to that of the plain image. For NP and FFX, the pixel correlation distributions were different from that of the plain image. In addition, the pixel correlation of FFX was slightly weak due to the use of FFX, compared with the other ones, so this property might have caused a lower accuracy than those of the other transformations in Table 1. Accordingly, there is some correlation between pixels in the transformed images, so block-wise transformations can achieve a high classification accuracy.

B) Key sensitivity test

We carried out key sensitivity tests for models with $M = 4$ and 8 on the CIFAR-10 and CIFAR-100 datasets. We define key sensitivity as the difference in accuracy between the correct key set and the modified key set (i.e. the correct key set with a small change), which is given by

$$\text{Key Sensitivity} = \text{ACC} - \text{ACC}', \quad (12)$$

where ACC is the classification accuracy with a correct key set, and ACC' is that with a key set that has a small change from the correct key set.

To make a small change, we swapped two random elements in the correct key for SHF, and one element in the

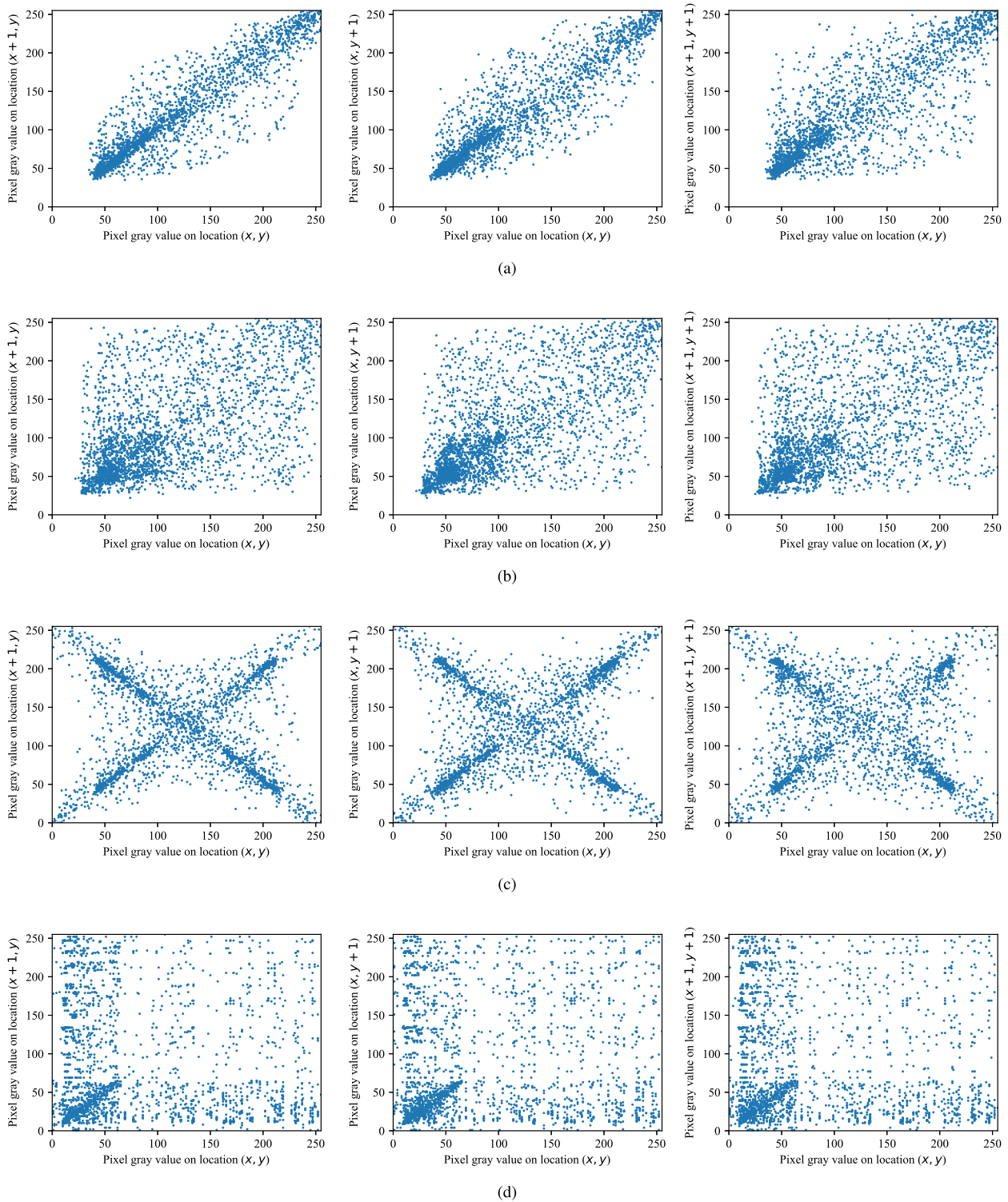


Fig. 4. Horizontal, vertical, and diagonal correlation test results for plain image and images transformed by SHF, NP, and FFX ($M = 4$). (a) represents horizontal, vertical, and diagonal correlation distribution of plain image, (b) represents that of image transformed by SHF, (c) represents that of image transformed by NP, and (d) represents that of image transformed by FFX.

correct key was flipped for NP and FFX (i.e. “o” to “1” and “1” to “o”). Table 6 shows the result of the key sensitivity tests, where the values in the table were averaged over $c \times M \times M$ times to cover changes in the different positions of the keys. From the results, we observed that low key

sensitivity values reflected a higher accuracy for the incorrect keys, and high ones corresponded to a lower accuracy for the incorrect ones, as shown in Table 1. The key sensitivity in the table gives some insights into the difference among transformations.

Table 6. Key sensitivity of various transformations with $M = 4$ and 8

| Transformation | CIFAR-10 | | CIFAR-100 | |
|----------------|----------|---------|-----------|---------|
| | $M = 4$ | $M = 8$ | $M = 4$ | $M = 8$ |
| SHF | 1.79 | 0.31 | 4.5 | 0.34 |
| NP | 9.68 | 0.11 | 18.9 | 1.97 |
| FFX | 5.17 | 0.29 | 10.43 | 0.52 |
| SHF + NP | 3.8 | 0.59 | 7.68 | 1.04 |
| SHF + FFX | 5.15 | 0.67 | 8.63 | 1.04 |
| SHF + NP + FFX | 5.65 | 0.72 | 10.13 | 1.06 |

C) Discussion

Key improvement: When $M = 2$, the key space for the block-wise transformations is relatively small, so brute force attacks are possible. To improve the key space, there are two ways: (1) to use a larger block size such as 8×8 , 8×4 , etc. and (2) to use a combined transformation such as SHF + NP or SHF + FFX. Note that a value of M affects not only the key space but also the classification accuracy and key sensitivity. Accordingly, users are requested to find a good trade-off among them.

Selection of transformations: Classification accuracy and model protection performance depend on the type of transformation and block size M . We recommend the following selection of transformations accordingly. When a higher classification accuracy is required, NP or a combined transformation such as SHF + NP or SHF + FFX with a small block size M is recommended. When higher protection performance is preferred, a larger M with SHF or a combined transformation is suitable.

Application range: In this paper, the proposed model-protection method focuses on image classification tasks because the three encryption methods used in this paper are designed for image classification tasks. When these encryption methods are applied to other tasks such as image segmentation and image retrieval, the performance may drop compared with that of using plain images. Therefore, the proposed model protection is limited to image classification tasks, and novel image transformations are expected to be designed for applying other tasks.

VI. CONCLUSION

We proposed a model protection method that utilizes block-wise transformations with a secret key set to transform input images. Specifically, the transformation methods are pixel shuffling, negative/positive transformation, and FFX. The performance accuracy of a protected model was closer to that of a non-protected model when the key set was correct, and it dropped drastically when an incorrect key set was given, suggesting that a protected model is not usable even when the model is stolen. The proposed method is also applicable to large datasets like the ImageNet dataset, which has never been tested by previous model-protection methods. Moreover, the proposed model-protection method does not introduce any overhead in both training and inference time. It is also robust against fine-tuning attacks in

which the adversary has a small subset of a training dataset to adapt a new forged key set and key estimation attacks.

FINANCIAL SUPPORT

This study was partially supported by JSPS KAKENHI (Grant Number JP21H01327) and JST CREST (Grant Number JPMJCR20D3).

REFERENCES

- [1] LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *Nature*, **521** (7553) (2015), 436.
- [2] Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S.: Embedding watermarks into deep neural networks, in *Proc. 2017 ACM Int. Conf. Multimedia Retrieval*, 2017, 269–277.
- [3] Merrer, E. L.; Pérez, P.; Trédan, G.: Adversarial frontier stitching for remote neural network watermarking. *Neural Comput. Appl.*, **32** (13) (2020), 9233–9244.
- [4] Fan, L.; Ng, K.; Chan, C.S.: Rethinking deep neural network ownership verification: embedding passports to defeat ambiguity attacks, in *Adv. Neural Inf. Process. Syst.*, 2019, 4716–4725.
- [5] Sakazawa, S.; Myodo, E.; Tasaka, K.; Yanagihara, H.: Visual decoding of hidden watermark in trained deep neural network, in *2nd IEEE Conf. Multimedia Inf. Process. Retrieval*, 2019, 371–374.
- [6] Rouhani, B. D.; Chen, H.; Koushanfar, F.: Deepsigns: a generic watermarking framework for IP protection of deep learning models, *arXiv:1804.00750*, 2018. [Online]. Available: <http://arxiv.org/abs/1804.00750>.
- [7] Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M.P.; Huang, H.; Molloy, I.: Protecting intellectual property of deep neural networks with watermarking, in *Proc. of the 2018 on Asia Conf. Comput. Commun. Security*, 2018, 159–172.
- [8] Chen, H.; Rouhani, B. D.; Koushanfar, F.: Deepmarks: a digital fingerprinting framework for deep neural networks, *arXiv:1804.03648*, 2018. [Online]. Available: <http://arxiv.org/abs/1804.03648>.
- [9] Adi, Y.; Baum, C.; Cissé, M.; Pinkas, B.; Keshet, J.: Turning your weakness into a strength: Watermarking deep neural networks by backdooring, in *27th USENIX Security Symposium*, 2018, 1615–1631.
- [10] Fredrikson, M.; Jha, S.; Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures, in *Proc. of the 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, 1322–1333.
- [11] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R.: Intriguing properties of neural networks, in *Int. Conf. Learning Representations*, 2014.
- [12] AprilPyone, M.; Kiya, H.: Encryption inspired adversarial defense for visual classification, in *2020 IEEE Int. Conf. Image Processing (ICIP)*, 2020, 1681–1685.
- [13] AprilPyone, M.; Kiya, H.: Block-wise image transformation with secret key for adversarially robust defense. *IEEE Trans. Inf. Forensics Sec.*, **16** (2021), 2709–2723.
- [14] Kawamura, A.; Kinoshita, Y.; Nakachi, T.; Shiota, S.; Kiya, H.: A privacy-preserving machine learning scheme using ETC images. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, **103** (12) (2020), 1571–1578.
- [15] Tanaka, M.: Learnable image encryption, in *2018 IEEE Int. Conf. Consumer Electronics-Taiwan (ICCE-TW)*, 2018, 1–2.

- [16] Sirichotedumrong, W.; Kinoshita, Y.; Kiya, H.: Pixel-based image encryption without key management for privacy-preserving deep neural networks. *IEEE Access*, (7) (2019), 177 844–177 855.
- [17] Chuman, T.; Sirichotedumrong, W.; Kiya, H.: Encryption-then-compression systems using grayscale-based image encryption for jpeg images. *IEEE Trans. Inf. Forensics Sec.*, **14** (6) (June 2019), 1515–1525.
- [18] Sirichotedumrong, W.; Maekawa, T.; Kinoshita, Y.; Kiya, H.: Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain, in *2019 IEEE Int. Conf. Image Processing (ICIP)*, 2019, 674–678.
- [19] Sirichotedumrong, W.; Kiya, H.: Grayscale-based block scrambling image encryption using ycbcr color space for encryption-then-compression systems. *APSIPA Trans. Signal Inf. Process.*, **8** (2019), E7.
- [20] Kurihara, K.; Imaizumi, S.; Shiota, S.; Kiya, H.: An encryption-then-compression system for lossless image compression standards. *IEICE Trans. Inf. Syst.*, **100** (1) (2017), 52–56.
- [21] Chuman, T.; Kurihara, K.; Kiya, H.: Security evaluation for block scrambling-based ETC systems against extended jigsaw puzzle solver attacks, in *2017 IEEE Int. Conf. Multimedia and Expo (ICME)*, 2017, 229–234.
- [22] AprilPyone, M.; Kiya, H.: Training DNN model with secret key for model protection, in *2020 IEEE 9th Global Conf. Consumer Electronics (GCCE)*, 2020, 818–821.
- [23] Swanson, M.D.; Kobayashi, M.; Tewfik, A.H.: Multimedia data-embedding and watermarking technologies. *Proc. IEEE*, **86** (6) (1998) 1064–1087.
- [24] Craver, S.; Memon, N.D.; Yeo, B.; Yeung, M.M.: Resolving rightful ownerships with invisible watermarking techniques limitations, attacks, and implications. *IEEE J. Sel. Areas Commun.*, **16** (4) (1998), 573–586.
- [25] Madono, K.; Tanaka, M.; Onishi, M.; Ogawa, T.: Block-wise scrambled image recognition using adaptation network, *arXiv:2001.07761*, 2020. [Online]. Available: <https://arxiv.org/abs/2001.07761>.
- [26] Sirichotedumrong, W.; Kiya, H.: A GAN-based image transformation scheme for privacy-preserving deep neural networks, in *2020 28th European Signal Processing Conf. (EUSIPCO)*, 2020, 745–749.
- [27] Ito, H.; Kinoshita, Y.; Kiya, H.: Image transformation network for privacy-preserving deep neural networks and its security evaluation, *arXiv:2008.03143*, 2020. [Online]. Available: <https://arxiv.org/abs/2008.03143>.
- [28] Ito, H.; Kinoshita, Y.; Kiya, H.: A framework for transformation network training in coordination with semi-trusted cloud provider for privacy-preserving deep neural networks, in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conf. (APSIPA ASC)*, 2020, 1420–1424.
- [29] Bellare, M.; Rogaway, P.; Spies, T.: Addendum to “The FFX mode of operation for format-preserving encryption,” *A parameter collection for enciphering strings of arbitrary radix and length, Draft 1.0*, NIST, 2010.
- [30] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition, in *Int. Conf. Learning Representations*, 2015.
- [31] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S.: Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems*, 2019, 8024–8035.
- [32] Krizhevsky, A.; Hinton, G.: Learning multiple layers of features from tiny images, University of Toronto, Tech. Rep., 2009.
- [33] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L.: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, **115** (3) (2015), 211–252.
- [34] Wong, E.; Rice, L.; Kolter, J.Z.: Fast is better than free: revisiting adversarial training, in *Int. Conf. Learning Representations*, 2020.
- [35] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep residual learning for image recognition, in *Proc. of the IEEE Conf. Computer Vision and Pattern Recognition*, 2016, 770–778.
- [36] Smith, L. N.; Topin, N.: Super-convergence: very fast training of residual networks using large learning rates, *arXiv:1708.07120*, 2017. [Online]. Available: <http://arxiv.org/abs/1708.07120>.
- [37] Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G. F.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; Wu, H.: Mixed precision training, *arXiv:1710.03740*, 2017. [Online]. Available: <http://arxiv.org/abs/1710.03740>.
- [38] Dollinger, J.: Pure Python format preserving encryption. <https://github.com/emulbreh/pyffx>.

AprilPyone MaungMaung received a BCS degree from the International Islamic University Malaysia in 2013 under an Albukhary Foundation Scholarship and an MCS degree from the University of Malaya in 2018 under the International Graduate Research Assistantship Scheme. Since 2019, he has been a Ph.D. student under the Tokyo Human Resources Fund for City Diplomacy Scholarship at Tokyo Metropolitan University. He received an IEEE ICCE-TW Best Paper Award in 2016. His research interests are in the area of information security. He is a student member of IEEE.

Hitoshi Kiya received his B.E. and M.E. degrees from the Nagaoka University of Technology, Japan, in 1980 and 1982, respectively, and his Dr.Eng. degree from Tokyo Metropolitan University in 1987. In 1982, he joined Tokyo Metropolitan University, where he became a Full Professor in 2000. From 1995 to 1996, he attended The University of Sydney, Australia, as a Visiting Fellow. He is a fellow of IEEE, IEICE, and ITE. He was a recipient of numerous awards, including 10 best paper awards. He served as the President of APSIPA from 2019 to 2020, and the Regional Director-at-Large for Region 10 of the IEEE Signal Processing Society from 2016 to 2017. He was also the President of the IEICE Engineering Sciences Society from 2011 to 2012. He has been an editorial board member of eight journals, including *IEEE Transactions on Signal Processing*, *IEEE Transactions on Image Processing*, and *IEEE Transactions on Information Forensics and Security*, and a member of nine technical committees, including the APSIPA Image, Video, and Multimedia Technical Committee (TC), and IEEE Information Forensics and Security TC. He has organized a lot of international conferences in such roles as the TPC Chair of IEEE ICASSP 2012 and as the General Co-Chair of IEEE ISCAS 2019.