IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 21, 2020, accepted November 4, 2020, date of publication November 9, 2020, date of current version November 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3036650

Deep Neural Networks Based Key Concealment Scheme

TAEHYUK KIM[1,2], TAEK YOUNG YOUN[3], (Member, IEEE), AND DOOHO CHOI[1,2], (Member, IEEE)

[1]Department of Information Security Engineering, University of Science and Technology (UST), Daejeon 34113, South Korea
[2]Cyber Security Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea
[3]College of Software Convergence, Dankook University, Gyeonggi-do 16890, South Korea

Corresponding author: Dooho Choi (dhchoi@etri.re.kr)

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government, Ministry of Science and ICT (MSIT) (<Q|Crypton>, No. 2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity).

ABSTRACT To keep the Internet-of-things (IoT) environment secure, employing a cryptographic function to various IoT devices has become vital. An important factor to consider is how to store a cryptographic key (or passwords) securely. A popular method is to store the key in the storage protected by some hardware-based security functions. This paper presents a novel concept to conceal cryptographic keys into deep neural networks (DNNs), named DNNs-based key concealment scheme. In this scheme, a key can be concealed into a proper deep neural network model which is trained with secret input data. We demonstrate the practical applicability of our concept by presenting an instance and a use-case scenario of the DNNs-based key concealment scheme and show its correctness. To prove its robustness, two fundamental security evaluation methods are proposed for investigating the security of the instantiation. To the best of our knowledge, this is the first attempt of its kind.

INDEX TERMS Key concealment, deep neural networks, key generation, noisy data.

I. INTRODUCTION

As the market for Internet-of-things (IoT) devices continues to grow, security issues have attracted much attention. Cryptographic systems can be used to resolve such issues within IoT environments. In a cryptographic system, a secure cryptographic key storage is a critical issue. This is because, according to Kerckhoffs's principle, one of the most significant principles in modern cryptography, a cryptographic system should be secure regardless of whether everything about the system except the key becomes public data. Many methods can be used to store and use cryptographic keys securely. One traditional method to store the key is to memorize and recall it whenever required. However, a drawback of this method is that it is difficult to recall various keys. Accordingly, assuming that an identical key is employed in numerous IoT devices, if any one key is compromised, then all the systems using the key are at risk.

There are two popular ways to be secure cryptographic keys without relying on the memory of people. One is using a secure storage, including trusted platform modules (TPM) or hardware security modules (HSM), to store the cryptographic keys. The disadvantage of using a hardware-based method is that it is more expensive than software-based methods. Therefore, implementing hardware-based modules in low-cost devices within the IoT environment may be difficult. Furthermore, there is a risk of losing the embedded secure storage, which is the same as losing a key. The other method uses secret information with noise, like biometric data, to register and regenerate a key whenever necessary. Representative methods include fuzzy commitment schemes [1] and fuzzy extractor [2], [3], which use an error correction code (ECC) as their chief idea.

Numerous key generation techniques include an ECC as the main factor and employ strategies analogous to the concept of the fuzzy extractor. Some authors, however, have proven that it is possible to generate a key dynamically without an ECC. For example, [4] provides a key generation technique without using an ECC to bind and release a key using fingerprints. Because an ECC is not employed, there is no trade-off in the security performance between the size of the registered key and the speed at which the key is generate.

The associate editor coordinating the review of this manuscript and approving it for publication was Ilsun You.

The authors register a bit of the key to a convertible authentic fingerprint template (using the cancelable biometric technique), if a bit of the key is 1. Otherwise, the key bits are registered to a synthetic template. If the converted cancelable template of the query fingerprint matches well with the registered template, then the key generating system generates a key bit of 1.

In order to extend the proposed design without the use of an ECC, we ask the following research questions:

- **Is it possible to conceal a key within a deep neural networks (DNNs) using noisy data?**

In this paper, we provide a new concept to hide a (crypto-)key within DNNs, which is called DNN-based key concealment scheme.

A. OUR CONTRIBUTIONS

Our contribution in this work is threefold: First, we define a conceptual mechanism of DNNs-based key concealment scheme to conceal a key within DNNs without any error correction. The DNNs-based key concealment scheme includes two main stages as follows:

- Key Concealment
  - **Feature generation block** generates features of noisy data.
  - **Key concealment block** binds a cryptographic key to features generated using the above block and generates neural networks for training the key concealed features.
  - Training the above two blocks to conceal the key. Noisy data collected from IoT devices are used for positive training, and synthetic data are used for negative training.
- Key Reproduction
  - Reproducing a concealed key from the trained DNNs with the noisy data collected when the key is needed.

Second, we give a detailed example using this mechanism to show its feasibility and propose a use-case scenario applying our DNN-based key concealment scheme in the IoT environment. In our example of Section III-B, images and an International Mobile Equipment Identity (IMEI) number which is a unique number for identifying a device on a mobile network are employed as the secret input.

Finally, we give the experimental result to show that our example of the DNNs-based key concealment scheme correctly reproduce the concealed key and two security evaluation methods are provided for verifying the robustness of the scheme. In the proposed scheme, the noisy data used in the key concealment and reproduction stage are secret data and trained DNNs that have optimized parameters are public data. Therefore, it is an essential security evaluation that there is no possibility of key exposure, even if the public data are analyzed. This kind of leakage can be evaluated by our security analysis methods. Furthermore, we show an evidence that the partial key information can be leaked if inappropriate data are used in the key concealment network for training.

Note that a preliminary version of this paper was presented at WISA2019 [5]. The main difference in this paper is that we present a use-case scenario in IoT environments. Also the results of correctness evaluation are improved by initializing weight values of feature extraction in a normal distribution.

B. ORGANIZATION OF THE PAPER

This paper is organized as follows. Section II discusses some necessary preliminary concepts. Section III defines the novel approach to conceal a key in DNNs. And, we propose an example and a use-case scenario to prove the feasibility of the scheme. Section IV, shows the correctness through experiments and proposes two fundamental security evaluation tools to confirm the robustness of the instantiation. Section V describes various open issues regarding this novel key concealment concept. Finally, Section VI concludes this paper.

II. PRELIMINARIES

A. CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network (CNN) [6], which is a type of DNN, is typically used to process multiple layered images. CNNs involve two stages: feature sampling and classification. In the feature sampling stage, particular feature maps of input data are extracted through convolutional and pooling layers. The convolutional layer executes a convolutional computation between images and a convolutional kernel set. The computation generates feature maps of the image by only activating the highlighted part of the kernel. The feature maps pass through a rectified linear unit (ReLU) [7]. The ReLU is a kind of activation functions which transform the summed weighted input into the output defined by the function. The feature maps from the ReLU are scaled down in the pooling layer. The most common pooling method, max pooling, calculates the maximum value from each chunk of the feature map. Through the above process, the feature maps are extracted and used in the classification stage. This stage includes a fully connected layer and a loss layer. In the fully connected layer, all neurons are fully connected to the feature maps. Therefore, the activation of the fully connected layer is computed by matrix multiplication and transmitted to the loss layer. In the loss layer, the difference between the label and the predicted value is computed. The weights are modified so as to minimize the deviation.

B. LOGISTIC REGRESSION WITH NEURAL NETWORK

Logistic regression [8] is a statistical model in which binomial dependent variables are modeled using logistic functions. By using this model, we could measure the probability of occurrence of a specific event. In machine learning, iterative training produces an optimized linear equation between inde-

pendent and dependent variables.

$$h = \begin{pmatrix} x_1 & x_2 & \cdots & x_i \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_i \end{pmatrix}$$

The sigmoid function yields a value y between 0 and 1 using the result of the linear equation h. The weight values, w_i, are optimized to reduce the difference between the binary input label and y.

III. DNN-BASED KEY CONCEALMENT SCHEME

This section defines the new key concealment mechanism called the DNNs-based key concealment scheme and validates its feasibility through a detailed example and a use-case scenario.

A. PROPOSED METHODOLOGY

We consider noisy data, for example images and fingerprints that can be used for training a key concealment network and reproducing the key using the key reproduction network. Let k be a cryptographic key, which is chosen randomly. The DNNs-based key concealment scheme includes two stages: key concealment and key reproduction. They are described in detail in the following sections.

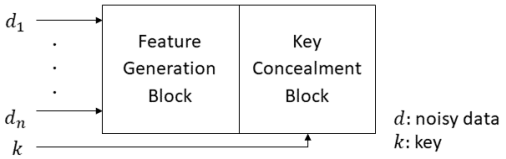


FIGURE 1. Conceptual diagram of key concealment.

1) KEY CONCEALMENT

In Figure 1, d_i is noisy data where $1 \leq i \leq n$ and n is the number of noisy data types. We use two noisy data—image data and the IMEI number—; therefore, n is 2. Although the length of the IMEI number is 60 bits, it is used as noisy data by adding 4 random bits to it. In Figure 1, the feature generation block represents neural networks for generating features of the noisy data $d_1, \ldots, d_n$. In the key concealment block, a key k is bound to the features generated in the previous block and concealed in the neural networks by training. The key concealment network is trained using noisy data d_i to conceal k. Once the training phase is completed, the network and its parameters are stored for use in the key reproduction stage.

2) KEY REPRODUCTION

The key reproduction stage reproduces k concealed in the trained DNNs if the noisy data d_i is proper where $1 \leq i \leq n$ (Figure 2). The feature generation and key reproduction blocks are the same as those of the trained key concealment network, except for the step that binds k to features.

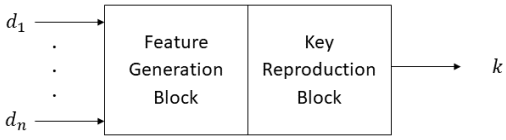


FIGURE 2. Conceptual diagram of key reproduction.

In Section III-B, an instantiation of our DNNs-based key concealment scheme is presented to demonstrate the feasibility of this novel approach.

B. INSTANTIATION

Here, we present a detailed instance for the DNNs-based key concealment scheme proposed in Section III-A. Our instantiation is shown in Figure 3. First, we prepare images of size 16×16 and 64-bit IMEI numbers as noisy data d_1 and d_2. Note that d_1 and d_2 are secret information; however, the DNNs and its parameters are public information.

1) KEY CONCEALMENT

The feature generation block, the first block in the key concealment stage, consists of the feature sampling part of the CNN and an IMEI binding layer. The key concealment block consists of a key binding layer and sk logistic regressions where sk is the key bit size. Let img and $imei$ represent the image and the IMEI number used as input data in the layers, respectively.

The feature generation block works as follows:

1) **Convolutional layer** Let w_{CNN} be a kernel of size $7 \times 7 \times sk$ and a stride of 2. img passes through the kernel and the ReLU function.
2) **Max pooling layer** Calculate the maximum value for each 2×2 patch of the output from the convolutional layer. Then, feature maps $\{m_i\}_{i=1}^{sk}$ are generated where the size of m_i is 8×8.
3) **IMEI binding** Let $m_i = [f_1^i, \ldots, f_{64}^i]$, $imei = p_1\, p_2 \ldots p_{64}$, and $pm_i = [pf_1^i, \ldots, pf_{64}^i]$ be an output of this layer where $1 \leq i \leq sk$. Determine $pf_j^i = (-1)^{(1-p_j)} \cdot f_j$, where the sign of f_j is changed if the j-th bit of the IMEI number is 0. Then, modified feature maps $\{pm_i\}_{i=1}^{sk}$ are generated. The IMEI number $imei$ could be bound in many different ways; we use simple sign changing, which showed excellent outcomes in the experiments described in Section IV-A.

The key concealment block works as follows:

1) **Key binding** Let $\{km_i\}_{i=1}^{sk}$ be an output of this layer. Compute $km_i = [(-1)^{(1-k_i)} \cdot pf_1^i, \ldots, (-1)^{(1-k_i)} \cdot pf_{64}^i]$ for each $1 \leq i \leq sk$. This means that the sign of every component pf_j^i in pm_i is changed if k_i is 0 for $1 \leq i \leq sk$ and $1 \leq j \leq 64$.
2) **Logistic regression** Logistic regressions $\{LR_i\}_{i=1}^{sk}$ are generated. Each logistic regression has weight values w_{LR}^i where $1 \leq i \leq sk$, a sigmoid function, and cross-entropy. The feature map km_i from the key binding

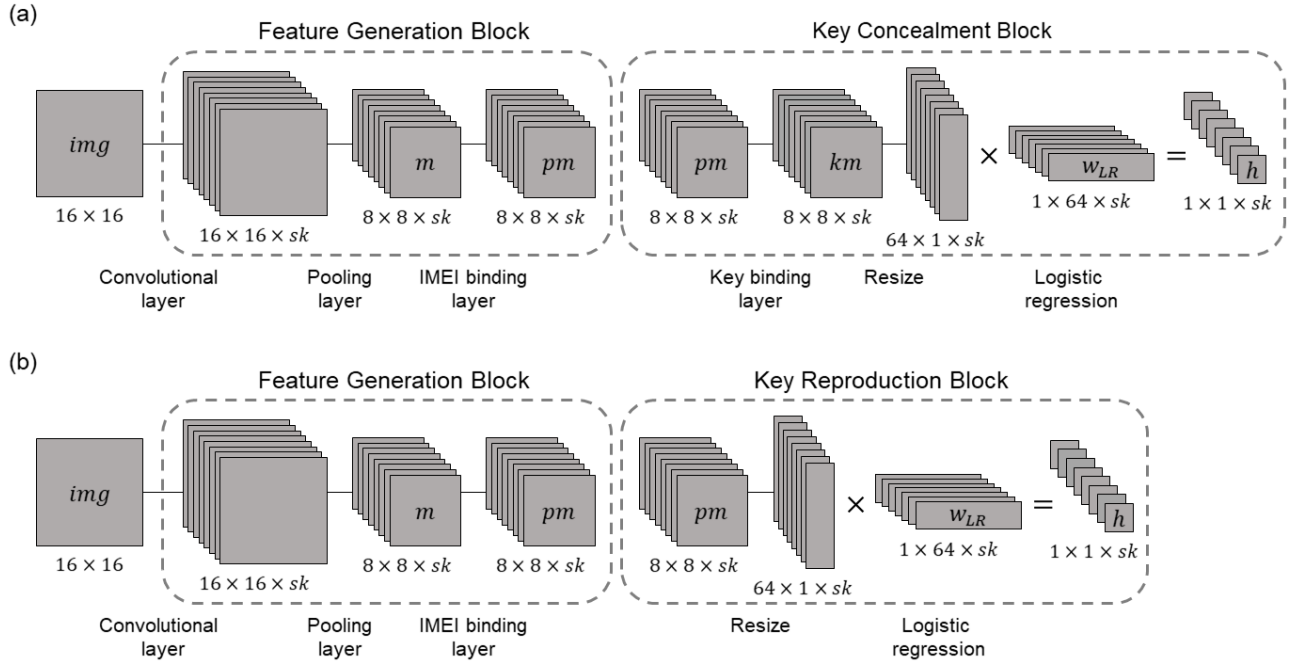
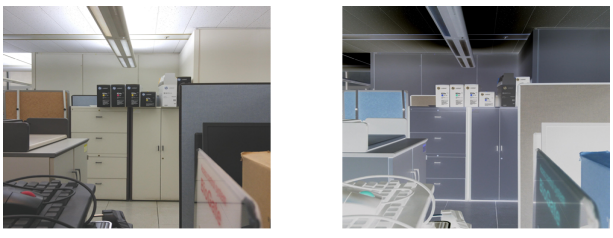


FIGURE 3. Graphical description of our instantiation: (a) key concealment and (b) key reproduction.

layer is an input of the logistic regression LR_i. Let h_i be an output of $km_i \times w^i_{LR}$ and y_i be an output of $sigmoid(h_i)$.

Now, we explain the two types of training used to conceal a key in the DNNs.

- **Positive training** is the training step with the label 1. The weight $\{w^i_{LR}\}^{sk}_{i=0}$ in the logistic regression are changed to reduce the difference between y_i and the label. The data employed for positive training are as follows:
  - Authentic images: images d_1 collected from IoT devices (see (a) in Figure 4).
  - Authentic IMEI numbers: IMEI numbers d_2 with 4 random bits added.
  - Authentic key: key k.
- **negative training** is the training step with the label 0 in sk logistic regressions, and it is very important in the security of our example. If inappropriate data are employed in this process, a serious security problem could occur (see Section IV). The data employed for negative training are as follows:
  - Synthetic images: images generated by converting authentic images to binary representations and then reversing each bit (see (b) in Figure 4).
  - Synthetic IMEI numbers: modified values by reversing all bits of the authentic IMEI numbers d_2.
  - Synthetic Key: modified value by reversing all bits of the authentic key.



(a) (b)

FIGURE 4. Images used in (a) positive training and (b) negative training.

2) KEY REPRODUCTION

The key reproduction stage includes the feature generation and key reproduction blocks. The sk logistic regression part of the key reproduction block predicts a key using features $\{pm_i\}^{sk}_{i=1}$ that are generated from the feature generation block and optimized weight values w^i_{LR} in the logistic regression. Let y_i be the result of $sigmoid(pm_i \times w^i_{LR})$ for $1 \leq i \leq sk$. Then, y_i has a value between 0 and 1. We apply a key determination rule whereby if $y_i > 0.5$, the i-th bit of a key k' is 1; otherwise, it is 0. Finally, a predicted iikey k' is reproduced. If a proper image and an IMEI number are produced, k' and k are the same (see Algorithm 1).

Here, we describe how the concealed key k could be reproduced for a proper image and an IMEI number. Note that, we change the sign of every element pf^i_j of the feature pm_i when the i-th bit of k is 0. Additionally, all the features in $\{km_i\}^{sk}_{i=1}$ are trained using label 1 in positive training. Assuming that w_{CNN} and w^i_{LR} where $1 \leq i \leq sk$ are optimized, let pm^c_i and pm^r_i denote the i-th feature from the feature generation block that are generated in the key concealment

Algorithm 1 Key Reproduction

input : img', $imei' = p'_1\, p'_2 \ldots p'_{64}$
output: k'

Restore w_{CNN}, w^i_{LR}
/* feature generation block */
$img_c = ConvolutionalLayer(img)$
$m = MaxPoolingLayer(img_c)$
for $i = 0$ **to** sk **do**
  for $j = 0$ **to** 64 **do**
    if $p'_j = 0$ **then**
      $pf^i_j = -f^i_j$
    else
      $pf^i_j = f^i_j$
/* key reproduction block */
for $i = 0$ **to** sk **do**
  $h_i = pm_i \times w^i_{LR}$
  $y_i = sigmoid(h_i)$
  if $y_i > 0.5$ **then**
    $k'_i = 1$
  else
    $k'_i = 0$
return k'

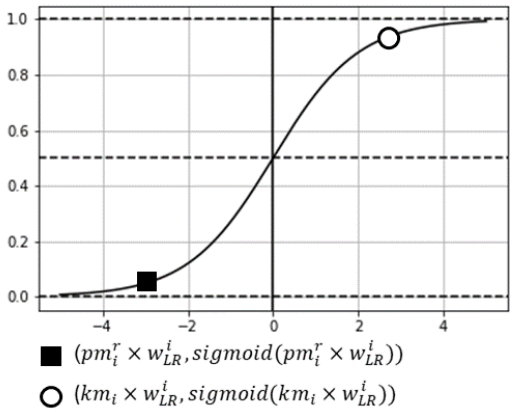


FIGURE 5. Outputs of sigmoid function for $k_i = 0$.

and key reproduction stages, respectively, and k_i be i-th bit of the k where $1 \le i \le sk$.

- $k_i = 0$: the sign of every component of pm^c_i is changed so that $km_i = -pm^c_i$ and $sigmoid(km_i \times w^i_{LR}) \approx 1$. However, as the step for changing the sign of the feature map pm^r_i is not included, the absolute values of $pm^r_i \times w^i_{LR}$ and $km_i \times w^i_{LR}$ are approximately identical, but the signs are opposite. Accordingly, $sigmoid(pm^r_i \times w^i_{LR})$ is near to 0. Following the key determination rule, the reproduced bit of the key is 0 (see Figure 5).
- $k_i = 1$: opposed to 0, the signs of every component of pm^c_i are not altered; accordingly, $km_i = pm^c_i$ and $sigmoid(km_i \times w^i_{LR}) \approx 1$. In key reproduction, the absolute values of $pm^r_i \times w^i_{LR}$ and $km_i \times w^i_{LR}$ are identical. Furthermore, their signs are also almost identical. Therefore, $sigmoid(pm^r_i \times w^i_{LR})$ is close to 1. Following the key determination rule, the reproduced bit of the key is 1 (see Figure 6).

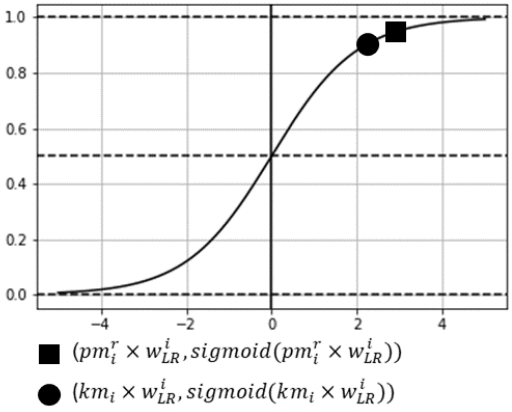


FIGURE 6. Outputs of sigmoid function for $k_i = 1$.

C. USE-CASE SCENARIO

In this section, we show the manner in which the DNNs-based key concealment scheme could be applied in an IoT environment. The example IoT environment contains a server and IoT devices. We assume that the IoT devices used in our scenario were produced with the server's public key in the manufacturing process. The detailed use-case scenario is as follows:

1) The IoT devices are installed at specific places.
2) The noisy data collected from the IoT devices are encrypted with the server's public key and transmitted to the server.
3) The server decrypts the encrypted noisy data from each of the IoT devices and trains a key with the two noisy data. The training method follows the one detailed in Section III-B1. After the training is complete, the architecture and parameters of the DNN are transmitted to each IoT device.
4) The IoT devices can reproduce a concealed key using the architecture and parameters of the DNN if valid noisy data are produced. The reproduction method is described in Section III-B2.

Although adversaries obtain the architecture and parameters of the DNN in 3), they cannot reproduce the concealed key with the data. Because we employ the steady training to prevent an adversary reproduce the concealed key. A detail explanation is in Sec IV-B.

Remark: In this scenario, the server generates the cryptographic keys for the IoT devices. However, this means that the server could be the single point of failure. If the server is attacked, it would put the IoT environment at a serious risk, because the server generates all the keys used for the IoT devices. If the learning process does not need to be delegated

to the server, the cryptographic keys are generated by the IoT devices, and therefore the server is not the single point of failure.

To show that the scenario can be applied to the real world, we run the key concealment scheme on IoT environment and measure the execution time. We only measure the key reproduction time which are performed on IoT devices, not a server. Samsung Galaxy S20 is employed for an IoT device. The execution time of the key reproduction is 0.003, 0.007 and 0.016 seconds for each key size—64, 128, and 256—.

IV. ANALYSIS

A. CORRECTNESS

In this section, we show that the DNNs-based key concealment scheme can reliably reproduce a concealed key regardless of the key size and does not reproduce a concealed key when improper data are entered. In this experiment, we employ the following data:

- proper images: almost equal images as the images employed for concealing the key k.
- improper images: images dissimilar to the images used to conceal a key. They are collected randomly in Pixabay [9].
- proper IMEI numbers: IMEI numbers employed to conceal the key k.
- improper IMEI numbers: a set of arbitrary bit strings.

Each set contains 1,000 data. The DNNs-based key concealment scheme is implemented using TensorFlow [10]. And 3000 images, 3000 IMEI numbers and one key are used for training the key concealment network. The experimental results are presented in Table 1. We use the false negative ratio (FNR) and the false positive rate (FPR) for measuring the correctness. In our experiments, the FNR represents the probability of not reproducing a concealed key for proper data. Additionally, the FPR shows the probability of reproducing a concealed key in the case of improper data. In case 1 presented in Table 1, the FNR is 0%,, thereby implying that concealed key is reproduced reliable when proper images and IMEI numbers are used. Furthermore, the FNR for each key size—64, 128, and 256—shows a same value, thereby implying that our instantiation can stably reproduce the concealed key regardless of the key size. In cases 2,3 and 4, the FPR is 0%. It shows that our instantiation does not reproduce the concealed key with improper data. In the experiments of [5], the FNR of case 1 was 0.1%, 0.6%, and 0.2% for 64, 128 and 256 key, respectively. And the FPR of case 3 was 4.1%, 2%, and 0.8%. For the other cases, the FPR was all zero. However, we obtain better results by initializing weight values of feature extraction in a normal distribution; in [5], the weight values are initialized in a uniform distribution.

TABLE 1. Correctness.

	input data			key size		
	image	IMEI		64	128	256
case1	Proper images	Proper IMEI	FNR	0.0	0.0	0.0
case2	Proper images	Improper IMEI		0.0	0.0	0.0
case3	Improper images	Proper IMEI	FPR	0.0	0.0	0.0
case4	Improper images	Improper IMEI		0.0	0.0	0.0

FNR(%): False Negative Ratio
FPR(%): False Positive Ratio

B. ROBUSTNESS

In Section IV-A, we showed that the instantiation conceals a key and stably reproduces it for appropriate data. Here, we present two security evaluation methods for verifying the robustness of the scheme and our experimental results. As described earlier, the secret and public data of our scheme are as follows:

- secret data: proper noisy data employed for reproducing the concealed key during key reproduction.
- public data: key reproduction mechanism including building blocks of DNNs and its optimized parameters.

The two potential attack mechanisms are proposed for guessing the concealed key using the public data.

1) An adversary can examine the possibility of key exposure from the parameters in the key reproduction stage.
2) An adversary can use random inputs to gather a considerable amount of results from the key reproduction stage, and then employ statistical methods to reveal the information of the concealed key.

Based on the two above-mentioned attack mechanisms, we present two security evaluation methods.

1) WEIGHTS AGGREGATE EVALUATION

In this analysis, an attacker investigates the optimized parameters in the key reproduction stage and tries to identify information of the concealed key. In the instantiation described in Section III-B, the public parameters are w_{CNN} and w_{LR}. w_{CNN} is not related to the key; however, the weight of the key concealment block, w_{LR}, is closely related to the concealed key. Therefore, the adversary can exploit the weight values. Let $w^i_{LR} = (w^i_1, w^i_2, \ldots, w^i_{64})$ be the weight value related to the i-th key bit reproduction in (b) in Figure 3 for $1 \leq i \leq sk$. Now, we simply aggregate all the components $(w^i_1, w^i_2, \ldots, w^i_{64})$ in each weight value in $\{w^i_{LR}\}^{sk}_{i=1}$, and obtain $\{weightAgg_i\}^{sk}_{i=1}$ as the result.

$$
\begin{aligned}
weightAgg_1 &= (w^1_1 + w^1_2 + \ldots + w^1_{64}) \\
weightAgg_2 &= (w^2_1 + w^2_2 + \ldots + w^2_{64}) \\
&\ldots \\
weightAgg_{sk} &= (w^{sk}_1 + w^{sk}_2 + \ldots + w^{sk}_{64})
\end{aligned}
$$

Finally, the adversary determines the key bits as follows (also refer to Algorithm 2):

- the i-th bit of the guessed key is determined as 1 if $weightAgg_i > 0$
- the otherwise, the i-th bit of the guessed key is determined as 0

Algorithm 2 Weight Values Aggregate Evaluation

input : $\{w_{LR}^{i}\}_{i=1}^{sk}$
output: An guessed key k'

for $i = 0$ **to** sk **do**
  $weightAgg = 0$
  for $j = 0$ **to** 64 **do**
    $weightAgg = weightAgg + w_j^i$
  end
  if $weightAgg > 0$ **then**
    $k_i' = 1$
  else
    $k_i' = 0$
  end
end
return k'

We evaluate our instantiation with the images and IMEI numbers to validate its correctness. We compare two cases in which steady negative training or unsteady negative training is employed using the Pearson correlation. The factors of a steady negative training are described in Section III-B. Additionally, we show that if someone does not use the factors of steady negative training, the information of the concealed key can be partially exposed from the analysis of Algorithms 2. The unsteady negative training data are generated carelessly as follows:

- Factors of an unsteady negative training
  - Authentic images: images identical to the ones employed for positive training.
  - Synthetic IMEI numbers: values reversing each bit of the authentic IMEI numbers employed for positive training.
  - Authentic key: a key identical to the one employed for positive training

The Appendix A details the results of the experiment for the concealed key size 64. In the Table 2, the aggregate of weights $weightAgg_i$, guessed key bit and Pearson correlation coefficient are presented for both steady and unsteady negative training. In the case of steady negative training, the Pearson correlation coefficient is approximately 0.09 between the concealed key and guessed key. It implies that the guessed key is analogous to the randomly generated bit strings, and therefore, there is no exposure of information regarding the concealed key. In the case of unsteady negative training, however, the Pearson correlation coefficient is 0.84. This means that the guessed key contains a significant amount of information regarding the concealed key.

2) STATISTICAL EVALUATION

Fredrikson *et al.* [11] presented an inversion attack that infers training data by analyzing the outputs of machine learning algorithms. Furthermore, they succeeded in regenerating face images using APIs of facial recognition service. In the instantiation we presented, there is a potential risk that an adversary can infer the training data including images, IMEI numbers and a key using the inversion attack. There are two ways of extracting the concealed key using the inferred data. First, by inferring the concealed key directly from the key reproduction outputs. Second, by inferring the training images and IMEI numbers using the inversion attack and extracting the key with key reproduction. In this evaluation, we focus on the first scenario and regard the second scenario as an open issue.

Based on [11], in the second evaluation, we assume that an adversary uses the outputs of the key reproduction stage to statistically extract a concealed key. The adversary can randomly select the input data and perform key reproduction to produce the bit strings. Assuming that they collect numerous random input data and execute key reproduction several times, they can construct a set of keys. From this set, they can count the number of 0 and 1 values at the i-th bit of the entire key, with $zeroNum$ and $oneNum$ as the results. Then they can set a key determination rule whereby if $zeroNum > oneNum$, the i-th key bit is conjectured to be 0; otherwise, it is conjectured to be 1. This statistical analysis is described in algorithm 3.

Algorithm 3 Statistical Evaluation

input : Random images and IMEI numbers
output: An inferred key k'

$\{zeroNum_i\}_{i=1}^{sk} = 0$
$\{oneNum_i\}_{i=1}^{sk} = 0$
while $i < iteration$ **do**
  $candKey = keyReproduction(img_i, imei_i)$
  for $i = 0$ **to** sk **do**
    if $candKey_i = 0$ **then**
      $zeroNum_i = zeroNum_i + 1$
    else
      $oneNum_i = oneNum_i + 1$
    end
  end
end
for $i = 0$ **to** sk **do**
  if $zeroNum_i > oneNum_i$ **then**
    $k_i' = 0$
  else
    $k_i' = 1$
  end
end
return k'

Once again, the instantiation from Section III-B is analyzed using the images and IMEI numbers to confirm its correctness. We now compare the two cases using the Pearson correlation and show a potential risk of key exposure. The

unsteady negative training data are generated carelessly as follows:

- Factors of an unsteady negative training
  - Authentic images: images identical to the ones employed for positive training.
  - Synthetic IMEI numbers: bit strings generated randomly
  - Authentic key: a key same as the one employed for positive training

The Appendix B details the results of the experiment for the concealed key size 64. In the experiment, we assume that the attacker has 1000 arbitrary images and IMEI numbers. In the Table 3, the number of 0 *zeroNum*, number of 1 *oneNum*, guessed key bit and Pearson correlation coefficient are presented for both steady and unsteady negative training. In steady negative training, the Pearson correlation coefficient is 0.1 between the concealed key and guessed key. This implies that the guessed key is analogous to the randomly generated bit strings, such there is no potential risk of key exposure. In unsteady negative training, however, the Pearson correlation coefficient is 0.58. This implies that the guessed key includes a significant amount of information of the concealed key.

V. OPEN ISSUES

To the best of our knowledge, this is the first attempt to conceal a cryptographic key within a DNN. The following unsolved questions remain require further research.

- **Networks architecture issues**
  - In instantiation, we employ a feature extraction of CNN for extracting feature maps from images, and a logistic regression to conceal the key. Notably, the neural network architecture can be varied depending on the noisy data used.
  - We employ the method of sign flipping to bind a key with the output. This results in the best performance in terms of correctness and robustness. However, other methods could be more efficient in some cases when instantiation is not used.
- **Noisy data issues**
  - We use images and outputs for concealing a key. In IoT environments, numerous types of noisy data could be collected from IoT devices.
  - In biometric cryptosystems, noisy data including fingerprints, faces and irises are used to generate cryptographic keys. The noisy data can also be employed in our scheme.
- **Security evaluation issues**
  - There are four attack methods that are generally used in machine learning (ML); an inversion attack [11], a poisoning attack [12], an evasion attack [13] and a model extraction attack [14]. The inversion attack (the model extraction, respectively) tries to reverse the input data (extract the model parameters, respectively) of a machine learning training model by analyzing a deployed ML model. The poisoning attack attempts to make misclassification on training phase by compromising the data collection, and the evasion attacker tries to generate input data of an ML system not to make a correct decision.
  - Firstly, the threat analysis for the model extraction is not necessary for our key reproduction, since the involved parameters are public information. Instead of this attack model, we have introduced the weight aggregate security analysis to evaluate the key leakage possibility in the weight parameters of the key reproduction.
  - In our concept, the input data for training of the key concealment is the noisy data (image and value in the example of Section III) and the key to be concealed. Our statistical security evaluation of Section IV only focuses on the key leakage possibility by analyzing the output of key reproduction for random input. Therefore, the threat analysis for the inversion attack of the noisy data still remain an open issue.
  - Because the training of the key concealment is securely performed in the given use-case scenario of Section III, and so it is unclear that the poisoning attack model should be considered. Therefore, the further study to find a poisoning attack scenario for our key concealment can be an open issue.
  - Lastly, in our key reproduction phase, the evasion attack means that an attacker attempts to obstruct the correct key extraction from the key reproduction by modifying the environment obtaining the noisy data, even if they don't know this noisy input data (this attack is a kind of the denial of service attack). In our example of Section III, an IoT device uses the outside image as its noisy input data of our key concealment and key reproduction. Therefore, the attacker may obstruct the key reproduction of the device by physically add small thing within the place around the device. Concrete attack scenario and experimental proof can be a good further study.

VI. CONCLUSION

In this paper, we propose a new DNNs-based key concealment scheme for concealing cryptographic keys within DNNs. To prove the feasibility of this approach, we present an instantiation and a use-case scenario of the proposed scheme and validate its correctness. We also present two fundamental security evaluation tools to check its robustness. Finally, we state several open issues that require further study. To the best of our knowledge, this is the first attempt at such an approach.

APPENDIX A
WEIGHT VALUES AGGREGATE EVALUATION OF UNSTEADY AND STEADY NEGATIVE TRAINING

TABLE 2. A comparison of unsteady and steady negative training in weight aggregate evaluation.

i-th bit	Unsteady training		concealed key	Steady training	
	weight agg	inferred key		inferred key	weight agg
1	0.56	1	1	0	-4.2
2	-0.58	0	0	1	0.72
3	1.12	1	1	1	0.2
4	0.62	1	1	1	2.24
5	-1.34	0	0	1	1.24
6	1.01	1	1	0	-1.59
7	0.81	1	1	1	1.52
8	-0.78	0	0	0	-3.18
9	0.84	1	1	1	0.33
10	-1.04	0	0	1	0.52
11	0.77	1	1	0	-0.14
12	-0.87	0	0	1	0.2
13	-1.37	0	0	1	0.31
14	-0.88	0	0	1	0.47
15	-0.43	0	0	0	-0.49
16	1.36	1	1	1	2.15
17	-0.71	0	0	0	-0.13
18	0.37	1	1	1	1.0
19	0.72	1	1	0	-0.54
20	1.24	1	1	0	-0.58
21	0.75	1	1	1	0.76
22	-1.01	0	0	1	1.0
23	0.17	1	0	0	-0.85
24	-0.41	0	0	0	-1.08
25	-0.17	0	1	0	-2.12
26	-0.39	0	0	1	0.37
27	-0.46	0	0	0	-1.6
28	0.9	1	1	0	-2.27
29	-1.47	0	0	1	0.5
30	-0.95	0	0	0	-1.0
31	0.7	1	1	1	1.77
32	-0.44	0	0	0	-0.02
33	-0.96	0	0	0	-0.91
34	-0.69	0	0	1	0.29
35	0.49	1	1	0	-0.62
36	0.64	1	1	0	-0.59
37	0.61	1	1	0	-0.65
38	0.49	1	1	1	0.65
39	0.63	1	1	0	-0.13
40	1.23	1	1	0	-1.74
41	0.77	1	1	1	1.89
42	-0.16	0	1	1	0.41
43	0.89	1	1	1	0.49
44	0.57	1	1	0	-3.43
45	0.23	1	1	1	0.54
46	-0.66	0	0	0	-1.56
47	-0.13	0	0	0	-0.54
48	0.37	1	1	0	-0.8
49	1.1	1	1	0	-0.87
50	-0.43	0	0	1	0.03
51	-0.86	0	0	1	0.94

i-th bit	weight agg	inferred key	concealed key	inferred key	weight agg
52	-0.36	0	0	1	0.25
53	0.01	1	0	1	0.4
54	0.96	1	1	1	0.68
55	-0.64	0	0	0	-0.39
56	-0.48	0	0	0	-0.49
57	-0.5	0	0	1	0.69
58	0.68	1	0	0	-1.09
59	0.3	1	1	1	0.76
60	-0.58	0	0	1	2.04
61	0.76	1	1	0	-0.84
62	-1.18	0	0	1	0.16
63	0.98	1	1	0	-3.04
64	-0.42	0	0	1	0.05
Correlation	0.84416			0.09379	

APPENDIX B
STATISTICAL EVALUATION OF UNSTEADY AND STEADY NEGATIVE TRAINING

TABLE 3. A comparison of unsteady and steady negative training in statistical evaluation.

i-th bit	Unsteady training			concealed key	Steady training		
	1	0	inferred key		inferred key	0	1
1	487	513	0	1	0	514	486
2	510	490	1	0	0	505	495
3	480	520	0	1	0	514	486
4	485	515	0	1	0	522	478
5	496	504	0	0	1	495	505
6	484	516	0	1	0	501	499
7	488	512	0	1	0	511	489
8	503	497	1	0	0	500	500
9	490	510	0	1	1	489	511
10	494	506	0	0	0	505	495
11	500	500	0	1	0	500	500
12	495	505	0	0	1	492	508
13	513	487	1	0	0	526	474
14	524	476	1	0	0	500	500
15	501	499	1	0	0	508	492
16	485	515	0	1	0	527	473
17	520	480	1	0	1	499	501
18	506	494	1	1	0	505	495
19	498	502	0	1	0	513	487
20	490	510	0	1	1	486	514
21	511	489	1	1	1	495	505
22	534	466	1	0	0	505	495
23	531	469	1	0	0	502	498
24	510	490	1	0	0	502	498
25	499	501	0	1	0	525	475
26	521	479	1	0	0	506	494
27	501	499	1	0	1	499	501
28	471	529	0	1	1	490	510
29	501	499	1	0	0	516	484
30	531	469	1	0	1	490	510
31	481	519	0	1	0	524	476
32	504	496	1	0	1	499	501
33	477	523	0	0	0	515	485
34	513	487	1	0	0	508	492
35	512	488	1	1	1	468	532

i-th bit	1	0	inferred key	concealed key	inferred key	0	1
36	485	515	0	1	1	491	509
37	504	496	1	1	0	536	464
38	480	520	0	1	1	499	501
39	488	512	0	1	1	495	505
40	481	519	0	1	1	487	513
41	506	494	1	1	0	507	493
42	456	544	0	1	0	504	496
43	511	489	1	1	1	499	501
44	509	491	1	1	0	519	481
45	507	493	1	1	1	494	506
46	506	494	1	0	1	490	510
47	526	474	1	0	1	480	520
48	510	490	1	1	0	520	480
49	499	501	0	1	0	512	488
50	506	494	1	0	1	497	503
51	520	480	1	0	1	487	513
52	521	479	1	0	0	518	482
53	514	486	1	0	1	476	524
54	473	527	0	1	1	488	512
55	495	505	0	0	0	507	493
56	516	484	1	0	0	503	497
57	511	489	1	0	0	533	467
58	528	472	1	0	1	481	519
59	498	502	0	1	1	497	503
60	507	493	1	0	0	503	497
61	489	511	0	1	0	506	494
62	511	489	1	0	1	498	502
63	493	507	0	1	1	491	509
64	508	492	1	0	1	496	504
Correlation	0.58134				0.10171		

REFERENCES

[1] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proc. 6th ACM Conf. Comput. Commun. Secur. CCS*, 1999, pp. 28–36.
[2] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology—EUROCRYPT)* (Lecture Notes in Computer Science), vol. 3027, 2004, pp. 523–540.
[3] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Jan. 2008.
[4] Z. Jin, A. B. J. Teoh, B.-M. Goi, and Y.-H. Tay, "Biometric cryptosystems: A new biometric key binding and its implementation for fingerprint minutiae-based representation," *Pattern Recognit.*, vol. 56, pp. 50–62, Aug. 2016.
[5] K. Taehyuk, Y. Taek-Young, and C. Dooho, "Is it possible to hide my key into deep neural network," in *Proc. Int. Workshop Inf. Secur. Appl.*, 2019, 259–272.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[7] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. ICML*, 2010, pp. 807–814.
[8] P. Harrington, *Machine Learning in Action*. Greenwich, CT, USA: Manning, 2012.
[9] H. Braxmeier, S. Steinberger, A. Thiemermann, and O. Foma. (2017). *Pixabay*. Accessed: Jan. 8, 2019. [Online]. Available: https://pixabay.com/
[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[11] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. - CCS*, 2015, pp. 1322–1333.
[12] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*. [Online]. Available: http://arxiv.org/abs/1206.6389
[13] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2013, pp. 387–402.
[14] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *Proc. 25th USENIX Secur. Symp. (USENIX) Secur.*, 2016, pp. 601–618.

TAEHYUK KIM received the B.S. degree from Incheon University, in 2016. He is currently pursuing the Ph.D. degree with the University of Science and Technology (UST), South Korea. His current research interests include security technologies of IoT, cryptography, and quantum crypto analysis.

TAEK YOUNG YOUN (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Korea University, in 2003, 2005, and 2009, respectively. From 2010 to 2020, he has worked as a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), South Korea. From 2016 to 2020, he was an Associate Professor with the University of Science and Technology (UST), South Korea. Since 2020, he has been an Assistant Professor with Dankook University, South Korea. His research interests include cryptography, information security, authentication, data privacy, and security issues in various communications.

DOOHO CHOI (Member, IEEE) received the B.S. degree in mathematics from Sungkyunkwan University, South Korea, in 1994, and the M.S. and Ph.D. degrees in mathematics from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 1996 and 2002, respectively. Since 2002, he has been a Principal Researcher with Electronics and Telecommunications Research Institute (ETRI). Since 2015, he has been a Professor with the University of Science and Technology (UST). From 2016 to 2017, he was a Visiting Research Fellow with Queens University Belfast, U.K. His main research interests include side channel analysis and its countermeasure design, quantum crypto analysis, and security technologies of IoT.