HHS Public Access

Author manuscript

IEEE Consum Commun Netw Conf. Author manuscript; available in PMC 2022 March 11.

Published in final edited form as:

IEEE Consum Commun Netw Conf. 2021 January ; 2021: . doi:10.1109/ccnc49032.2021.9369515.

DLWIoT: Deep Learning-based Watermarking for Authorized IoT Onboarding

Spyridon Mastorakis,
Dept. of Computer Science, University of Nebraska Omaha

Xin Zhong,
Dept. of Computer Science, University of Nebraska Omaha

Pei-Chi Huang,
Dept. of Computer Science, University of Nebraska Omaha

Reza Tourani
Dept. of Computer Science, Saint Louis University

Abstract

The onboarding of IoT devices by authorized users constitutes both a challenge and a necessity in a world, where the number of IoT devices and the tampering attacks against them continuously increase. Commonly used onboarding techniques today include the use of QR codes, pin codes, or serial numbers. These techniques typically do not protect against unauthorized device access–a QR code is physically printed on the device, while a pin code may be included in the device packaging. As a result, any entity that has physical access to a device can onboard it onto their network and, potentially, tamper it (*e.g.*, install malware on the device). To address this problem, in this paper, we present a framework, called Deep Learning-based Watermarking for authorized IoT onboarding (*DLWIoT*), featuring a robust and fully automated image watermarking scheme based on deep neural networks. *DLWIoT* embeds user credentials into carrier images (*e.g.*, QR codes printed on IoT devices), thus enables IoT onboarding only by authorized users. Our experimental results demonstrate the feasibility of *DLWIoT*, indicating that authorized users can onboard IoT devices with *DLWIoT* within 2.5–3sec.

Keywords

Internet of Things (IoT); IoT onboarding; deep learning; watermarking

I. Introduction

In a world, where the number of IoT devices rapidly increases every year, the onboarding of such devices has been a challenge, especially when security becomes a requirement. One of the common techniques for IoT onboarding today is the use of Quick Response (QR) codes [1] (other popular out-of-band IoT onboarding techniques include the use pin codes or

smastorakis@unomaha.edu .

serial numbers [2]). A user scans the QR code of an IoT device with his/her mobile phone and this QR code is translated to a url, which allows the user to communicate with a server typically located on the cloud. Through this process, an IoT device acquires the necessary configuration and credentials, so that it onboards the user IoT network (*i.e.*, becomes an operational part of this network) [3], [4].

However, such onboarding techniques typically do not protect against unauthorized device use, since, for example, a QR code is printed on the device or a pin code may be inside the retail packaging of the device. As a result, unauthorized users or any entity that has physical access to the IoT device can onboard it onto their personal network and, potentially, tamper the device (*e.g.*, install malicious software on it).

To protect against unauthorized IoT onboarding, in this paper, we present Deep Learning-based Watermarking for authorized IoT onboarding (*DLWIoT*). *DLWIoT* is a trusted third-party service that interacts with the users and manufacturers for secure onboarding of IoT devices through an image watermarking scheme [5] based on deep Neural Networks (NNs). In our design, the users contact the *DLWIoT* service after purchasing their IoT devices. *DLWIoT* embeds a watermark instructed by the user (*e.g.*, user credentials) covertly onto a carrier image (*e.g.*, QR code). The marked image (the watermark embedded into the carrier image–*e.g.*, a QR code with the user credentials embedded to it) is printed on the device. When the user receives the "watermarked" IoT device, this image will be used for the on-boarding of the device onto the user network. Specifically, the user will take a picture of the image with his/her mobile phone, the watermark will be extracted, and the device will be onboarded. Attackers will not be able to onboard and tamper the device even if they have physical access to it, assuming that they do not have access to the credentials used for the watermark creation.

A. Motivation and Contribution

Why using deep NNs for image watermarking in *DLWIoT:* Among the goals of an image watermarking system is robustness–the watermark must survive even after distortion or quality degradation of the marked image [5]. This is especially needed when a marked image is printed on a device and/or the embedded watermark is extracted from images captured by a mobile phone. In our use-case, in addition to attackers potentially tampering IoT devices, the attackers can further attempt to distort the marked images printed on the devices in order to prevent legitimate users from onboarding their devices. *DLWIoT* can utilize any carrier image for IoT onboarding, but, in this paper, we primarily focus on QR codes given their simplicity and widespread use as the means to onboard IoT devices today. More specifically, as illustrated in Fig. 1, legacy QR codes, even when error correction techniques are applied, can tolerate up to limited amounts of distortion–typically up to 15% [6] and at most up to 33% [7].

In *DLWIoT*, a deep NN for watermarking can provide enhanced robustness during the process of embedding a watermark into a marked image and extracting the watermark from camera resamples of a marked image. As a result, marked images (*e.g.*, QR codes) created through *DLWIoT* can tolerate severe distortions (up to 85-90%). More specifically, the deep NN of *DLWIoT* will dynamically learn the rules of watermark embedding and extraction,

resulting in a fully automated and optimized watermarking system that is able to deal with different degrees of distortion and image quality degradation (*e.g.*, lighting variations, compression, cropping, lens distortion) [8]. On the other hand, conventional watermarking schemes are based on fixed watermark embedding and extraction algorithms, thus being able to tolerate only certain types of distortions.

How is *DLWIoT* different than setting a password for each IoT device: *DLWIoT* not only supports the embedding of any binary-encoded image into a marked image, but it can also embed information, such as the user's voice (*e.g.*, a recorded word or phrase). In addition, *DLWIoT* can embed various biometrics, including users' fingerprints, iris scans, and gestures. Utilizing biometrics provides stronger security guarantees compared to password-based mechanisms [9] due to advanced social engineering and dictionary attacks against password-based systems as well as weak and predictable password selection. To showcase the *DLWIoT* design and capabilities, in this paper, we study the use-case of fingerprint scans to represent user credentials due to its popularity and hardware availability on user devices. We evaluate the embedding of fingerprint scans into marked images in Section IV.

Contribution: Our contribution is two-fold: (i) we present a novel scheme for image watermarking based on deep NNs that provides fully automated watermark embedding and extraction of enhanced robustness (Section III); and (ii) we take advantage of this scheme for the design of *DLWIoT*, a framework for authorized IoT onboarding, (Section II) and we evaluate the feasibility and tradeoffs of *DLWIoT* (Section IV). Our experimental results indicate that *DLWIoT* is able to onboard IoT devices for authorized users in 2.5 – 3*sec*.

II. *DLWIoT* Threat Model and Design

In this section, we first discuss the *DLWIoT* threat model and assumptions, and we then present the design of the *DLWIoT* architecture.

A. Threat Model and Assumptions

As we mentioned earlier, common practices for onboarding an IoT device include scanning a QR code or using the device's serial number (or pin code) that is printed on the device. The potential threats from an unauthorized user with physical access to the device include: *(a)* onboarding the device into the unauthorized user's network aiming to tamper with the device (*e.g.*, installing malware such as Mirai [10]); and *(b)* adding distortion or running a subtle cropping attack to prevent the legitimate user from successfully onboarding the device. In this paper, we assume that the *DLWIoT* service and IoT device manufacturers are trustworthy. This is a fair assumption since neither the *DLWIoT* service nor manufacturers gain benefit by interfering with the onboarding process.

B. DLWIoT *Design*

Figure 2 illustrates the design of *DLWIoT* service. In the context of *DLWIoT*, we assume that a user buys an IoT device online (*e.g.*, through a manufacturer's website or an online shopping website). Once the user has completed her purchase, she will be redirected to the

DLWIoT, which is a trusted third-party service similar to a certificate authority. Through a secure connection (*e.g.*, SSL/TLS [11]), the user (*e.g.*, user A in our example) uploads her credential that she would like to use for the watermark creation and a carrier image to the *DLWIoT* server. Alternatively, the user may select a carrier image (*e.g.*, a QR code) among the ones offered by the *DLWIoT* server.

Once the user credentials are received by the *DLWIoT* server, our embedding deep NN, running on the same or a different *DLWIoT* server (illustrated as running on the same server for simplicity in Figure 2) will create a marked image by embedding the user credentials (*e.g.*, fingerprint scan, recorded voice, image, cryptographic key)[1] into the carrier image. The *DLWIoT* service securely transmits the marked image to the manufacturer server and, subsequently, the production line. Finally, the watermarked image will be printed on the IoT device before delivering it to the user.

Upon receiving the watermarked IoT device, the user (user B in Figure 2) takes a picture of the marked image on the device. Subsequently, the user securely sends the picture and the credentials embedded in the marked image (*e.g.*, fingerprint scan) to the *DLWIoT* server (illustrated for simplicity as the same server for the watermark creation in Figure 2). The *DLWIoT* server extracts the embedded credentials from the marked image and compares them against the user-provided credentials. Upon successful verification, the server sends the contact information (*e.g.*, IP address, TCP/UDP port, public key) of the onboarding server to the user.

III. Deep Learning-based Image Watermarking

In this section, we present a deep learning-based image watermarking scheme, which is the core of *DLWIoT.* To facilitate our IoT onboarding scenario, the deep NN is specially trained for an image watermarking with three main merits: (i) to obtain an automated system without requiring domain knowledge, we exploit the fitting ability of deep NNs in learning image watermarking algorithms; (ii) we propose a deep learning architecture suitable to image watermarking that trains in an unsupervised manner to reduce human intervention; and (iii) the proposed scheme achieves robustness without any prior knowledge or adversarial examples of possible attacks.

A. Overall Architecture

An image watermarking scheme often consists of watermark embedding and extracting stages, and each stage can be decomposed into several steps in typical methods. The watermark embedding stage aims to insert a watermark into a cover image. The first step is to project a cover image into one of its feature spaces in spatial, frequency, or other domains. The obtained feature space is then modified to carry the watermark. To create a marked image, the modified feature space is projected back into the cover image space. Inversely,

[1]Note that storing user biometrics on the *DLWIoT* server may come with certain privacy concerns that should be considered. However, we would like to note that approaches to provide secure and privacy-preserving biometric storage and identification as a service on the cloud have been recently proposed [12], [13]. Such approaches can be utilized to complement the *DLWIoT* design. To alleviate privacy concerns related to storing user biometrics on the *DLWIoT* server, *DLWIoT* allows users to utilize a wide variety of other credentials (*e.g.*, any image, any recorded word or phrase).

watermark extraction is to project the marked image to the same feature space and then separate the watermark information. The watermark can be transformed or encoded based on different target applications. An image watermarking scheme often highlights its fidelity (*i.e.*, high similarity between the marked and the cover image) and robustness (*i.e.*, keeping the integrity of the watermark when there are noise and/or attacks applied to the marked image).

The idea of the proposed scheme is to develop a deep learning model to learn and generalize image watermarking algorithms. As shown in Figure 3, given two input spaces of watermark images and cover images, W and C, we first fit a function that encodes W to its encoded space W_f with NN μ_{θ_1} parameterized by θ_1. Then, an embedder function that inserts W_f into (a domain of) C is fit by NN σ_{θ_2} parameterized by θ_2. The obtained space after embedding for the marked image is named as M. To handle possible distortions, an NN τ_{θ_5} parameterized by θ_5 is introduced to fit a function coverting M to its transformed space T. During the transformation, τ_{θ_5} preserves information about W_f while rejecting all irrelevant noise on M, and hence providing robustness to the proposed scheme. Finally, the inverse watermark reconstruction functions are fit by two NNs, φ_{θ_3} and γ_{θ_4} with trainable parameters θ_3 and θ_4, that extract W_f from T and decode W from W_f respectively. Note that the convolutional NNs applied in the proposed scheme not only fit the processes of feature extraction and feature space modification performed in traditional watermarking schemes, but also optimize these processes dynamically.

B. Scheme Objective

The entire architecture is trained as a single deep NN with several loss terms designed for image watermarking. Given the data samples $w_i \in W$, $i = 1, 2, 3, \ldots$ and $c_i \in C$, $i = 1, 2, 3, \ldots$, the proposed scheme can be trained in an unsupervised manner. There are two inputs w_i and c_i, and two outputs m_i and w_i^* for the proposed deep NN. For the output w_i^*, an extraction loss that minimizes the difference between w_i^* and w_i is computed to ensure full extraction of the watermark. For the output m_i, a fidelity loss that minimizes the difference between m_i and c_i is computed to enable watermark invisibility. For the output m_i, we also compute an information loss that forces m_i to contain the information of w_i. To achieve this, we maximize the correlation between a feature map of w_f^i and a feature map of m_i. Denoting the parameters to be learned as $\vartheta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$, the loss function $L(\vartheta)$ of the proposed scheme can be expressed as:

$$L(\vartheta) = \lambda_1 \|w_i^* - w_i\|_1 + \lambda_2 \|m_i - c_i\|_1 + \lambda_3 \psi(m_i, w_f^i), \tag{1}$$

where λ_i, $i = 1, 2, 3$ is the weight factor and ψ is a function computing the correlation given as:

$$\begin{aligned} &\psi(m_i, w_f^i) \\ &= \frac{1}{2}(\|g(f_1(w_f^i)), g(f_1(m_i))\|_1 + \|g(f_2(w_f^i)), g(f_2(m_i))\|_1), \end{aligned} \tag{2}$$

where g denotes the Gram matrix that contains all possible inner products. By minimizing the distance between the Gram matrices of the feature maps produced by f_1 and f_2, we maximize their correlation. To extract the feature maps of m_i and w_f^i, the intermediate results (f_1 and f_2 of the "*" convolution block as shown in Figures 4 and 5) of two layers are applied (further explained in Section III-C).

In Eq. 1, each two of the fidelity loss, information loss, and extraction loss terms can be a trade-off for image watermarking–for example, minimizing the fidelity loss term to zero means that m_i is identical to c_i. However, in this case, there is no embedded information in m_i, thus the extraction of w_i will fail. To allow some imperfectness of the loss terms, the mean absolute error (*i.e.*, the L1 norm) is selected to highlight the overall performance rather than a few outliers.

With regularization, the proposed scheme objective is represented as $L(\vartheta) + \lambda_4 P$, where P is the penalty term to achieve robustness as in Eq. 6, and λ_4 is the weight controlling the strength of the regularization term. The deep NN needs to learn the parameter ϑ^* that minimizes $L(\vartheta) + \lambda_4 P$:

$$\vartheta^* = \arg\min_{\vartheta}[L(\vartheta) + \lambda_4 P]. \tag{3}$$

In the backpropagation during training, the term $\lambda_1 \| w_i^* - w_i \|_1$ is applied by the components of the architecture in their weight updates, while only μ_{θ_1} and σ_{θ_2} apply terms $\lambda_2 \| m_i - c_i \|_1$ and $\lambda_3 \psi(m_i, w_f^i)$ to their weight updates. This enables μ_{θ_1} and σ_{θ_2} to encode and embed the information in a way that φ_{θ_3} and γ_{θ_4} are able to extract and decode the watermark.

C. Design of Component NNs

This subsection describes the design of the component NNs μ_{θ_1}, σ_{θ_2}, φ_{θ_3}, γ_{θ_4} and τ_{θ_5} in more detail. The overall design is modularized and is illustrated in Figure 4. If we single out two pairs (μ_{θ_1}, γ_{θ_4}) and (σ_{θ_2}, φ_{θ_3}), we can find that each pair is conceptually symmetrical.

1) *The Encoder μ_{θ_1} and the Decoder γ_{θ_4} NNs:* Taking the samples w_j, $i = 1, 2, 3, \ldots$ from the input space W, the encoder NN μ_{θ_1} learns an encoding function that converts W to its feature space W_f. Inversely, the decoder NN γ_{θ_4} learns a decoding function from W_f to W with samples w_f^{i*}, $i = 1, 2, 3, \ldots$. μ_{θ_1} increases a 32 × 32 × 1 watermark image to 32 × 32 × 24 and 32 × 32 × 48, and γ_{θ_4} successively decreases the 32 × 32 × 48 feature space to a 32 × 32 × 1 watermark image. The reason to train this channel-wise increment is two-fold. First, it produces a 128 × 128 × 3 w_f^i that has the same width and height as the cover image, so that we can concatenate a feature map of w_f^i and c_i along their channel dimension. Each of w_f^i and c_i will contribute equally to the 128 × 128 × 6 concatenated matrix used in the embedder NN σ_{θ_2}, thus, we are evenly weighing the watermark and the cover image. Second, the increment in the latent space W_f introduces redundancy, decomposition, and perceivable randomness to W, which helps with robustness.

2) *The Embedder σ_{θ_2} and the Extractor φ_{θ_3} NNs:* The embedder NN σ_{θ_2} applies the convolution block f to extract a 128 × 128 × 3 to-be-embedded feature map of w_f^i that is concatenated along the channel dimension with the cover image. Directly applying c_i, while only applying a feature map of w_f^i, helps c_i to dominate the appearance. The 128 × 128 × 6 concatenation is fed into another convolution block to produce m_i. The extractor NN φ_{θ_3} inverses the process by two successive convolution blocks.

To capture various scales of features for image watermarking, the inception residual block [14] is applied. All the convolution blocks in Figure 4 have the structure shown in Figure 5, where F_w, F_d, and F_c respectively denote the height, width, and the channel of the block input. In the case of the "*" convolution block f of Figure 4, the annotated intermediate results f_1 and f_2 of Figure 5 are applied in Eq. 2. Specifically, block f extracts features not only from w_f^i, but also from m_i. The annotated $F_w \times F_d \times 96$ and $F_w \times F_d \times F_c$ feature maps are the intermediate results f_1 and f_2 respectively.

3) *The Invariance Layer τ_{θ_5}:* This is the key component to provide robustness in the proposed image watermarking scheme. Using a fully-connected layer, τ_{θ_5} learns a transformation from space M to an over-complete space T, where the neurons are activated sparsely. The idea is to redundantly project the most important information from M into T and to deactivate the neural connections of the areas on M irrelevant to the watermark, thus preserve the watermark even if there is noise or distortion that modified a part of M. As shown in Figure 4, τ_{θ_5} converts a 3-channel instance m_i of M into an N-channel ($N \geq 3$) instance t_i of T, where N is the redundant parameter. Increasing N results in increased redundancy and decomposition in T, which provides higher tolerance of the errors in M and thus enhances robustness.

Referring to the contractive autoencoder, τ_{θ_5} employs a regularization term that is obtained by the Frobenius norm of the Jacobian matrix of its outputs with regards to its inputs. Mathematically, the regularization term P is given as:

$$P = \Sigma_{i,j}\left(\frac{\partial h_j(X)}{\partial X_i}\right)^2, \tag{4}$$

where X_i denotes the i-th input and h_j the output of the j-th hidden unit of the fully connected layer. Similar to a gradient computation, the Jacobian matrix can be written as:

$$\frac{\partial h_j(X)}{\partial X_i} = \frac{\partial A(\omega_{ji}X_i)}{\partial \omega_{ji}X_i}\omega_{ji}, \tag{5}$$

where A is an activation function and ω_{ji} is the weight between h_j and X_i. We set A as the hyperbolic tangent (tanh) for strong gradients and bias avoidance [15]. Hence, P can be computed as:

$$P = \Sigma_j(1 - h_j^2)^2\Sigma_i(\omega_{ji}^T)^2. \tag{6}$$

If the value of P is minimized to zero, all weights ω in τ_{θ_5} will be zero, so that the output of τ_{θ_5} will be always zero no matter how we change the inputs X. Thus, minimizing P alone will cause the rejection to all the information from the inputs m_i. Therefore, we place P as a regularization term in the total loss function to teach τ_{θ_5} to preserve useful information related to the loss terms of image watermarking, while rejecting all other noise and irrelevant information. In this way, we achieve robustness without prior knowledge of possible attacks.

IV. Experimental Evaluation

In this section, we evaluate our *DLWIoT* framework by presenting experimental results on: (i) the training and testing of the deep NN; (ii) the overhead that *DLWIoT* introduces to the IoT onboarding process; and (iii) the robustness of *DLWIoT*'s watermarking scheme.

A. Experimental Setup

Deep NN deployment, training, and testing: The deep NN is trained and tested on four NVIDIA TITAN Xp (12GB) GPUs. The watermarking scheme is trained as a single deep NN using the ImageNet dataset [16] for cover images and the binary version of the CIFAR dataset [17] for watermarks, to introduce a large scope of instances to the proposed scheme. 10,000 images from each dataset are separated as the validation set. The testing is performed on 10,000 image samples from the Microsoft COCO dataset [18] as the cover image, and 10,000 images of the testing division of the binary CIFAR as the watermark. In Section IV-B1, we present results on: (i) training and testing of the deep NN; and (ii) the *Peak Signal-to-Noise Ratio (PSNR)* and *Bit-Error-Rate (BER)*, which are respectively used to quantitatively evaluate the fidelity of the marked image and the quality of the watermark extraction in the testing process. The PSNR is defined as:

$$PSNR = 10\log_{10}\left(\frac{(max(c_i)^2}{MSE(c_i, m_i)}\right) \tag{7}$$

where *MSE* is the mean squared error. The BER is computed as the percentage of error bits on the binarization of the watermark extraction w_i^*. Finally, in Section IV-B3, we present results on the robustness in terms of BER for varying marked image distortion percentages in comparison with QR codes.

Mobile application and IoT onboarding: We used a QR code as the carrier image having a stream of bits (*e.g.*, a user key, a password, a secret image, a user's fingerprint scan) as the watermark. The marked image is another QR code. We have developed a prototype Android application, so that users can take pictures of marked images (QR codes). These pictures are sent over WiFi to the GPU, where the watermarks are extracted and the user credentials are verified. If the verification is successful, the IoT device receives a url that will take it to a server for onboarding. In cases that the user's fingerprint scan has been embedded as the watermark, the user device needs to be equipped with a fingerprint scanner. The user will be asked to scan their fingerprint, since this fingerprint scan will be sent to the server, which will verify that it matches the scan embedded into the QR code. In Section IV-B2, we present results on the following metrics:

- *End-to-end delay for onboarding:* This is measured as the time elapsed between the moment that a user takes a picture of a marked image and sends it to the watermark extraction deep NN until the moment that the IoT device is onboarded.
- *Processing time for watermark embedding and extraction:* The processing time needed by the deep NNs to embed a watermark into a cover image and to extract a watermark from a marked image after it is requested by a user.
- *Utilization of resources during deep NN operation:* The GPU and memory usage during watermark embedding and extraction.
- *Accuracy of biometrics-based* DLWIoT: The *DLWIoT* accuracy when the users' fingerprint scans are used for onboarding given that the scans embedded into the QR code and the scans sent to the server may be similar, but not identical.

B. Experimental Results

1) NN training and testing: Figure 6a illustrates the NN loss function $L(\vartheta) + \lambda_4 P$ during 200 epochs. During the training and validation, the value of $L(\vartheta) + \lambda_4 P$ converges below 0.03, indicating a proper fitting.

In testing, the BER is zero, indicating that the original and the extracted watermarks are identical. The testing PSNR is $39.72 dB$, indicating a high fidelity of marked images, so that the hidden information cannot be identified by human vision.

2) IoT onboarding: We present our results in Table I.

End-to-end onboarding delay: This delay consists of the network delay to/from the GPU that runs the watermark extraction deep NN ($54 msec$ roundtrip delay), the extraction deep NN processing time, the processing by the user mobile phone, the network delay to the onboarding server ($52 msec$ roundtrip delay), and the processing by the onboarding server. Our results indicate that the end-to-end onboarding delay is roughly $2.5 sec$. Even in cases of longer roundtrip delays (*e.g.*, distant clouds with roundtrip delays of $150 – 200 msec$), the onboarding delay is not expected to exceed $3 sec$.

Processing delay for watermark embedding and extraction: Our results indicate that embedding by the deep NN lasts about $1.12 sec$. The extraction process lasts $1.62 sec$, requiring about $1.5x$ more processing time than embedding.

Resource utilization during deep NN operation: The water-mark embedding and extraction processes utilize a single GPU up to 26% and 41% respectively. The memory consumption is $7.47 GB$ for embedding and $11.21 GB$ for extraction.

Accuracy of biometrics-based *DLWIoT*: The accuracy of *DLWIoT* reaches 99%. The onboarding and processing delays and the resource utilization results reported above still hold.

3) DLWIoT *Watermark Robustness:* In Figure 6b, we present the BER for varying marked image distortion percentages for *DLWIoT* and legacy QR codes. The results demonstrate that legacy QR codes can tolerate distortions up to 1/3 of the image through the application of Error Correction Coding (ECC). However, *DLWIoT* can tolerate distortions of up to 2/3 of the marked image without ECC, while with ECC, *DLWIoT* can tolerate distortions up to 85-90%.

V. Related Work

Related Work on IoT Onboarding:

The selection of on-boarding techniques depends on the design of the security architecture (*e.g.*, distributed, centralized). Out-of-band techniques include the use of QR code and/or pre-defined passwords by users [2]. Onboarding in centralized architectures often relies on pre-established trust relations and utilize protocols, such as Extensible Authentication Protocol (EAP) [19], for authentication. In distributed architectures, devices do not rely on pre-established trust relations–onboarding results in credentials being created for security in subsequent communication. To this end, peer IoT devices can perform a Diffie-Hellman type of handshake to agree on a common secret [20]. Protocols such as IKEv2 [21] and TLS [11] allow peers to exchange keys and establish security associations without the need for a connection to a trusted server. Furthermore, approaches to secure and automate IoT onboarding through trusted hardware have been proposed [22]. In this paper, we focused on a centralized architecture through a deep learning-based watermarking scheme for IoT onboarding.

Related Work on Deep Learning for IoT:

The potential of deep learning in the context of IoT has been discussed in prior work [23], [24]. Deep learning has been used to detect tampered IoT devices [25], while work has also been done on running deep learning on IoT devices [26]. In this paper, we focused on deep learning to enable the onboarding of IoT devices by authorized users through a watermarking scheme.

Related Work on Deep Learning for Image Watermarking:

Although still at its infancy, incorporating deep learning into image watermarking has attracted increased attention in recent years. Zhong *et al.* [27] investigated a general-purpose deep learning-based watermarking scheme without considering the embedding of user biometrics to cover images. Kandi *et al.* [28] used two deep autoencoders to indicate bits 1 and 0 respectively for a non-blind binary watermark extraction. By embedding via adversarial images and extracting through the first layer of a deep NN, Vedran *et al.* [29] developed a single-bit watermarking scheme. In scenarios where a master share is sent separately from the image, Fierro-Radilla *et al.* [30] linked the watermark with features from the cover image extracted by convolutional NNs to create the master share. Due to the fragility of deep NNs [31], robustness is a challenge, since noise or modification on the marked image can destroy the trained models. Mun *et al.* [32] proposed to solve this issue by enumerating adversarial examples during training. In this paper, we achieved robustness

without adversarial examples of potential attacks and tolerate noise on the marked images without requiring any information from the cover images.

VI. Conclusion and Future Work

In this paper, we presented *DLWIoT*, a framework for IoT onboarding through image watermarking. In *DLWIoT*, user credentials can be covertly embedded into an image (*e.g.*, a QR code), which can be printed on an IoT device and can be used only by an authorized user to onboard the device. In its core, *DLWIoT* features a novel deep learning-based scheme for image watermarking that offers robustness against distortion and quality degradation of marked images.

While *DLWIoT* is off to a promising start, the current framework supports the embedding and extraction of the credentials of a single user. As a result, an IoT device can have a single owner, who can onboard it. In our future work, we plan to enable the embedding and extraction of the credentials of a group of users, so that IoT devices can be onboarded by a group of authorized users. We also plan to tackle changes of device ownership–for instance, cases of device re-selling, where the device owner sells the device to another individual.

Acknowledgements

This work is partially supported by a pilot award from the Center for Research in Human Movement Variability and the NIH (P20GM109090), the National Science Foundation under award 2016714, a planning award from the Collaboration Initiative of the University of Nebraska system, and the Nebraska Tobacco Settlement Biomedical Research Development Funds.

References

[1]. “Automatic identification and data capture techniques - QR code 2005 bar code symbology specification,” International Organization for Standardization, Standard, 2008.

[2]. Latvala S, Sethi M, and Aura T, “Evaluation of out-of-band channels for iot security,” SN Computer Science, vol. 1, no. 1, p. 18, 2020.

[3]. Nour B, Mastorakis S, and Mtibaa A, “Compute-less networking: Perspectives, challenges, and opportunities,” IEEE Network, 2020.

[4]. Mastorakis S, Mtibaa A, Lee J, and Misra S, “Icedge: When edge computing meets information-centric networking,” IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4203–4217, 2020.

[5]. Cox I, Miller M, Bloom J, Fridrich J, and Kalker T, Digital watermarking and steganography. Morgan kaufmann, 2007.

[6]. “QR code.com–Error Correction Feature,” 8 2020, [Online; accessed 19. August 2020]. [Online]. Available: https://www.qrcode.com/en/about/error_correction.html

[7]. Stark MM, QR Codes: The Technical Guide. AK Peters, Ltd., 2013.

[8]. Pramila A, Keskinarkaus A, and Seppänen T, “Camera based water-mark extraction-problems and examples,” in Proceedings of the finnish signal processing symposium. Citeseer, 2007.

[9]. Uludag U et al. , “Biometric cryptosystems: issues and challenges,” Proceedings of the IEEE, vol. 92, no. 6, pp. 948–960, 2004.

[10]. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Durumeric Z, Halderman JA, Invernizzi L, Kallitsis M et al., “Understanding the mirai botnet,” in 26th {USENIX} security symposium ({USENIX} Security 17), 2017, pp. 1093–1110.

[11]. Dierks T, “The transport layer security (tls) protocol version 1.2,” 2008.

[12]. Haghighat M, Zonouz S, and Abdel-Mottaleb M, “Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification,” Expert Systems with Applications, vol. 42, no. 21, pp. 7905–7916, 2015.

[13]. Barra S et al. , “Cloud-based biometrics (biometrics as a service) for smart cities, nations, and beyond,” IEEE Cloud Computing, vol. 5, no. 5, pp. 92–100, 2018.

[14]. Szegedy C, Ioffe S, Vanhoucke V, and Alemi AA, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[15]. LeCun YA et al., “Efficient backprop,” in Neural networks: Tricks of the trade. Springer, 2012, pp. 9–48.

[16]. Russakovsky O et al. , “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.

[17]. Krizhevsky A, Hinton G et al. , “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep, 2009.

[18]. Lin T-Y et al., “Microsoft coco: Common objects in context,” in European conference on computer vision. Springer, 2014.

[19]. Vollbrecht JR et al., “Extensible authentication protocol (eap),” 2004.

[20]. Diffie W and Hellman M, “New directions in cryptography,” IEEE transactions on Information Theory, vol. 22, no. 6, pp. 644–654, 1976.

[21]. Kaufman C, “Internet key exchange (ikev2) protocol,” 2005.

[22]. “Intel secure device onboard.” [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/infographics/intel-sdo-infographic.pdf

[23]. Li H, Ota K, and Dong M, “Learning iot in edge: Deep learning for the internet of things with edge computing,” IEEE Network, vol. 32, no. 1, pp. 96–101, 2018.

[24]. Al-Garadi MA, Mohamed A, Al-Ali A, Du X, and Guizani M, “A survey of machine and deep learning methods for internet of things (iot) security,” arXiv preprint arXiv:1807.11023, 2018.

[25]. Ferdowsi A and Saad W, “Deep learning-based dynamic watermarking for secure signal authentication in the internet of things,” in 2018 IEEE International Conference on Communications (ICC). IEEE, 2018.

[26]. Tang J, Sun D, Liu S, and Gaudiot J-L, “Enabling deep learning on iot devices,” Computer, vol. 50, no. 10, pp. 92–96, 2017.

[27]. Zhong X, Huang P-C, Mastorakis S, and Shih FY, “An automated and robust image watermarking scheme based on deep neural networks,” arXiv preprint arXiv:2007.02460, 2020.

[28]. Kandi H, Mishra D, and Gorthi SRS, “Exploring the learning capabilities of convolutional neural networks for robust image watermarking,” Computers & Security, vol. 65, pp. 247–268, 2017.

[29]. Vukotić V, Chappelier V, and Furon T, “Are deep neural networks good for blind image watermarking?” in 2018 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, 2018, pp. 1–7.

[30]. Fierro-Radilla A et al., “A robust image zero-watermarking using convolutional neural networks,” in 2019 7th International Workshop on Biometrics and Forensics (IWBF). IEEE, 2019, pp. 1–5.

[31]. Papernot N et al., “The limitations of deep learning in adversarial settings,” in 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016, pp. 372–387.

[32]. Mun S-M, Nam S-H, Jang H, Kim D, and Lee H-K, “Finding robust domain from attacks: A learning framework for blind watermarking,” Neurocomputing, vol. 337, pp. 191–202, 2019.

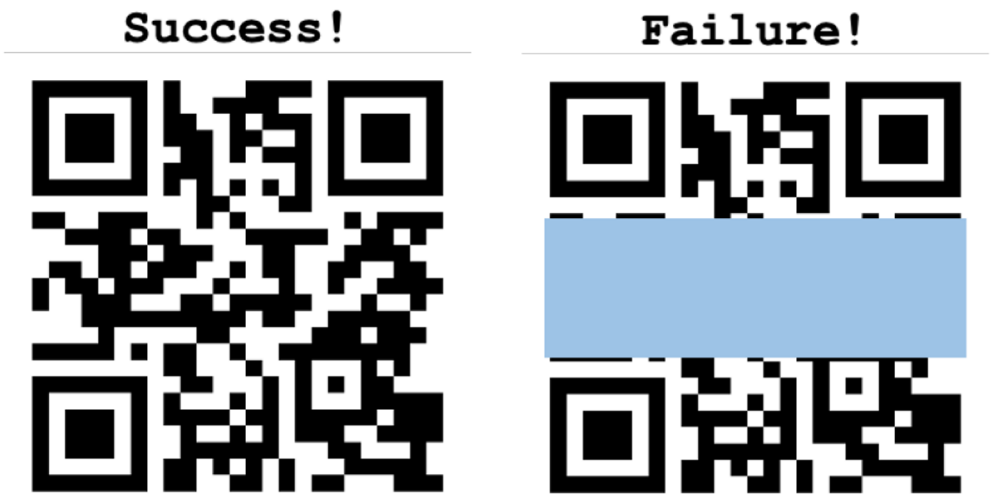

Fig. 1: Example of a successful and failed QR code scanning.

(a) QR code without distortion

(b) QR code with distortion of ≥ 33%

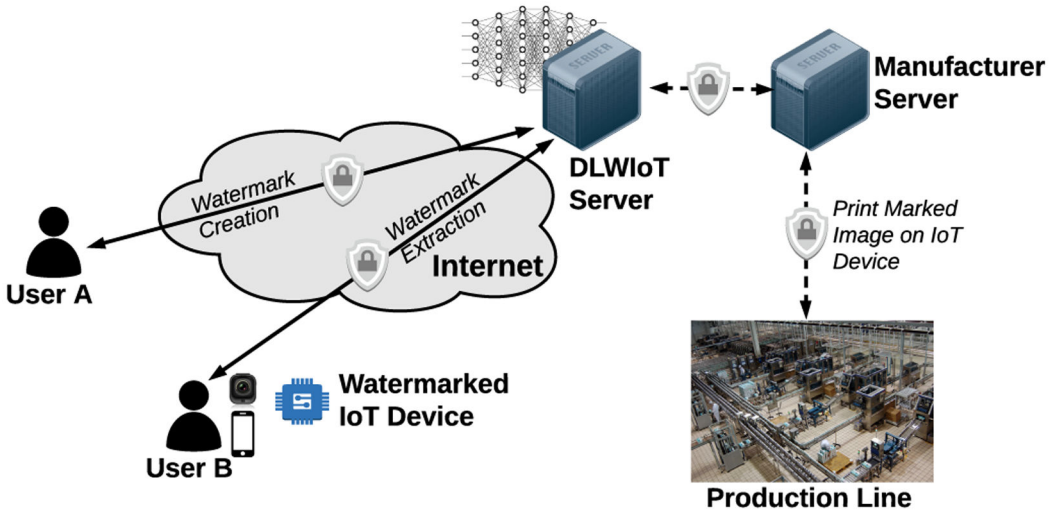

Fig. 2:
DLWIoT design and example workflow.

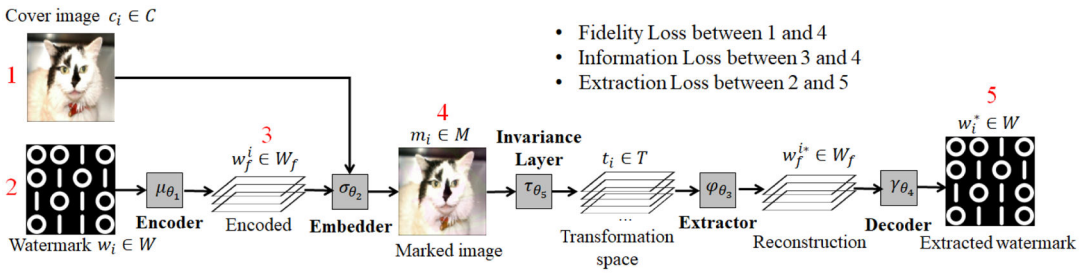

Fig. 3:
Overall architecture of the watermarking scheme.

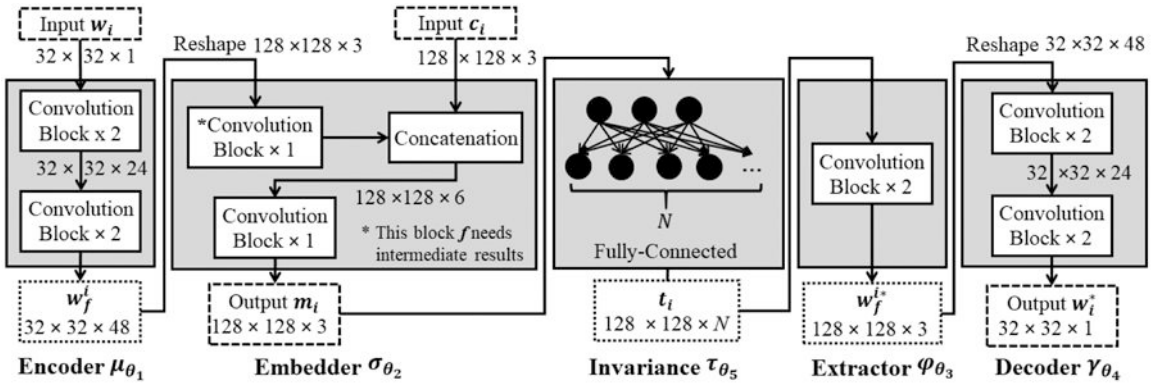

Fig. 4:

Detailed illustration of the components in the proposed watermarking scheme; Encoder μ_{θ_1}, Embedder σ_{θ_2}, the invariance layer τ_{θ_5}, Extractor γ_{θ_4}, and Decoder φ_{θ_3}.

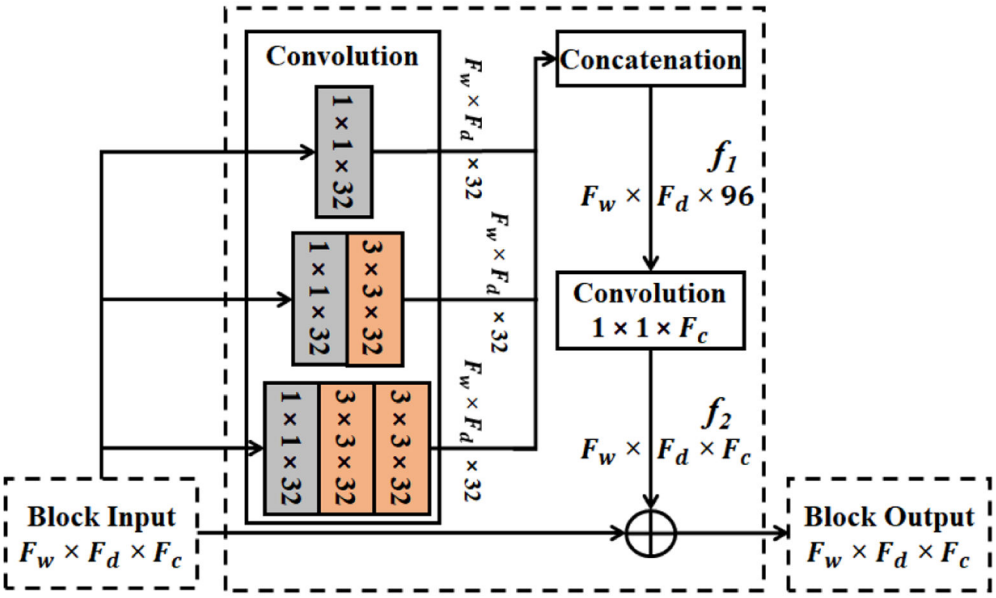

Fig. 5:
Design of a convolution block.

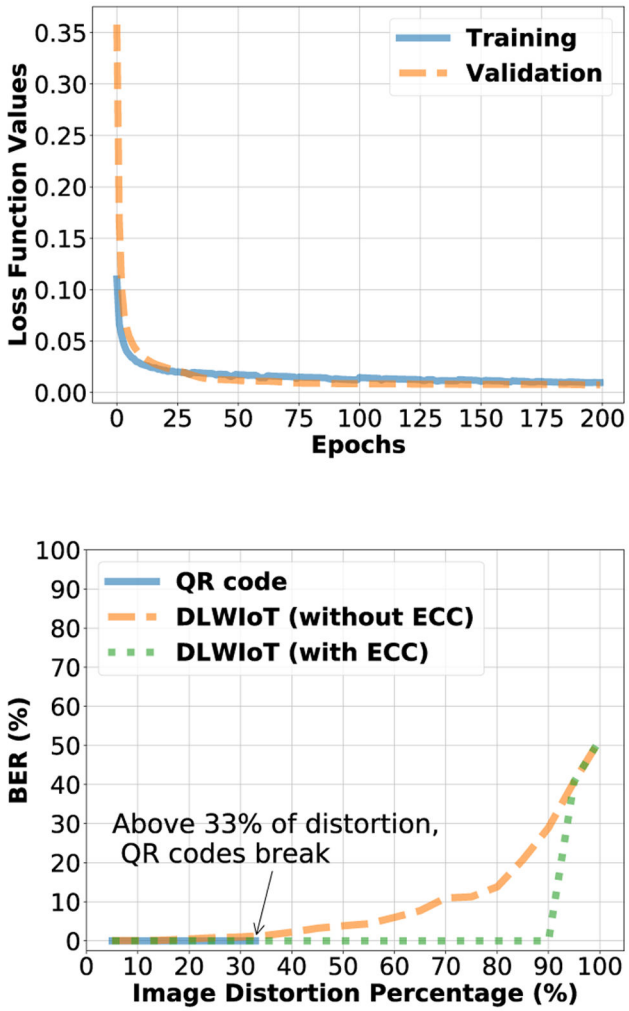

Fig. 6: Deep NN training and watermark robustness results.

(a) Training results: the NN loss function during 200 epochs.

(b) BER for varying image distortion percentages.

TABLE I:

Experimental IoT onboarding results (average and standard deviation)

End-to-end onboarding delay (sec)	Deep NN processing time (sec)		GPU utilization (%)		Memory usage (GBs)	
	Embedding	Extraction	Embedding	Extraction	Embedding	Extraction
2.53 ± 0.32	1.12 ± 0.11	1.62 ± 0.21	0-26	0-41	7.47	11.21