Deep-Lock: Secure Authorization for Deep Neural Networks

NOTE: An extended version of this work is published in the ACM Journal on Emerging Technologies in Computing Systems [2].

Manaar Alam
alam.manaar@gmail.com
Indian Institute of Technology Kharagpur
INDIA

Sayandeep Saha
sayandeep.iitkgp@gmail.com
Indian Institute of Technology Kharagpur
INDIA

Debdeep Mukhopadhyay
debdeep.mukhopadhyay@gmail.com
Indian Institute of Technology Kharagpur
INDIA

Sandip Kundu
kundu@umass.edu
University of Massachusetts Amherst
USA

ABSTRACT
Trained Deep Neural Network (DNN) models are considered valuable Intellectual Properties (IP) in several business models. Prevention of IP theft and unauthorized usage of such DNN models has been raised as of significant concern by industry. In this paper, we address the problem of preventing unauthorized usage of DNN models by proposing a generic and lightweight key-based model-locking scheme, which ensures that a locked model functions correctly only upon applying the correct secret key. The proposed scheme, known as Deep-Lock, utilizes S-Boxes with good security properties to encrypt each parameter of a trained DNN model with secret keys generated from a master key via a key scheduling algorithm. The resulting dense network of encrypted weights is found robust against model fine-tuning attacks. Finally, Deep-Lock does not require any intervention in the structure and training of the DNN models, making it applicable for all existing software and hardware implementations of DNN.

KEYWORDS
Deep Neural Network, Obfuscation, Intellectual Property Protection, Substitution Box

1 INTRODUCTION
The remarkable success of Deep Neural Networks (DNN) has created numerous opportunities for commercial applications. Constructing accurate DNN models for targeted applications is non-trivial, as it demands domain expertise and powerful computing resources. On the other hand, the datasets used for training are deemed one of the most valuable assets of modern businesses. Consequently, modern business models consider trained DNNs as Intellectual Property (IP) cores. The presence of such IP cores both in embedded platforms (Google Coral, Intel NCS) and cloud-based Machine-Learning-as-a-Service (MLaaS) (Amazon AWS, BigML) creates multiple practical security concerns. One such potent threat is the stealing decision boundary of the IP Core (also called Model Stealing), which has been addressed previously by several work [6, 8]. Businesses may suffer economic losses, as well as losses of brand value due to such incidents.

In this work, we address another threat model, which has been proposed recently in [3]. This is concerned with the model piracy of DNN models that may also result in significant economic losses for the original model owner. While model stealing may also lead to illegal usage, the problem of model piracy is more fundamental. An adversary may obtain/access a model from several other unwanted sources and continue using it as a black box (without bothering stealing its decision boundary). Unless there is some mechanism implemented for authorized access, this threat can never be mitigated. While this threat is applicable both in the context of MLaaS and embedded platforms, it is more severe in the latter case as the model can be easily extracted as a piece of software and distributed among multiple parties.

Although there exist techniques for watermarking a given DNN model for establishing model ownership [1, 4, 5], they cannot prevent its illegal usage. Hence, an explicit locking mechanism for DNN is in demand, which will prevent unauthorized usage by making the model malfunction. Previous research in this area has mainly focused on obfuscating model structures [7]. However, a more critical component of a DNN IP is the trained parameters. Most industrial applications typically use previously published DNN architectures, which have demonstrated high modeling capabilities but use different parameters depending on applications. Observing this, we present Deep-Lock, which enables a secret key-based locking for DNN parameters. Unauthorized access, without the knowledge of the legitimate key, makes the model unusable by severely degrading the model accuracy. Deep-Lock utilizes S-Boxes with good cryptographic properties to lock each trained parameter of a DNN model, which is found to be a lightweight approach for model locking, without causing any significant degradation in the inference time and model size. The secret keys required for locking are generated from a master key via a key schedule. The master key is only needed to be stored within the device. Moreover, the locking mechanism works on trained models and has no interaction with the training phase, making it generic and scalable for different model types. We have verified our claims over several industry-scale DNN models. We also considered improving the locked model accuracy by model fine-tuning, where a manifest dataset1 is used for improving the model accuracy even for wrong key values. It was found that model fine-tuning fails to improve a model’s accuracy to a practically significant value.

Deep-Lock bears certain similarities with a key-dependent hardware-assisted DNN IP protection scheme (HPNN) proposed in [3]. However, HPNN requires changes to be made in the model structure,

1The manifest dataset is a set of labeled data that is apparent to the user, though not necessarily a subset of original training dataset, but resembles sufficiently to the original training dataset.
which is not there in Deep-Lock, easing its usability. Furthermore, Deep-Lock uses established cryptographic constructs for locking, which is lacking in HPNN. It is worth noting that Deep-Lock differs significantly from state-of-the-art software product key schemes, where a key is verified before giving access to a locked software. The security of such schemes depends on the verification step, and there exist several well-known techniques to bypass this verification check and gain control over the entire product. Furthermore, once the verification step is passed, the unlocked model remains in the memory from where it can be stolen without much hindrance. On the other hand, Deep-Lock embeds the verification mechanism inside the DNN and mandates verification for every query made to the network. This never leaves the unlocked model in the memory, making the direct steal of the model challenging.

The rest of the paper is organized as follows: Section 2 discusses the adversarial threat model considered in this work. Section 3 presents the proposed methodology in details. The evaluation of performance and security of the proposed model is demonstrated in Section 4 and Section 5, respectively. Finally, Section 6 concludes the work.

2 THREAT MODEL
The primary objective of this work is that a DNN model owner will provide services only to authorized customers who have acquired a license for model usage by paying an amount. To achieve the objective, the model owner encrypts the DNN model with a secret key. An adversary aims to use the encrypted DNN model accurately without paying a license fee to the model owner. In this scenario, we consider an adversary who has white-box access to the encrypted model, i.e., she knows model structure as well as all encrypted parameter values. The only information she does not know is the secret key used to encrypt the model. We also assume that the adversary has a manifest dataset, which she can use to query the model with a key guess to obtain an output. If the adversary uses encrypted parameters in the stolen DNN architecture, it produces wrong input-output mapping as she does not have a legitimate secret key.

3 PROPOSED METHODOLOGY
The proposed key-based authorization scheme Deep-Lock operates in two modes – offline mode and online mode. In the offline mode, a DNN model owner trains and locks a model, and in the online mode, the DNN model is deployed for practical usage. The overview of all the operations performed in the offline mode is as follows:

- First, a DNN model owner acquires a massive set of data related to a particular task and expends a considerable amount of money, experts knowledge, and computational resources to label the data and train a model. The model owner can exercise any popularly used learning technique for fine-tuning model parameters to obtain the most accurate model.
- The DNN model owner then selects a master key and uses a key-scheduling algorithm to generate different secret keys corresponding to each trained parameter. The parameters are then encrypted with an S-Box operation using the derived secret keys. The original model parameters are replaced with these encrypted values before deploying it for practical usage. The lock operation is shown in Algorithm 1.

4 PERFORMANCE EVALUATION
The method is evaluated on Intel Xeon CPU E5-2690 v4 @ 2.60GHz with 56 cores. The Convolutional Neural Network (CNN) structures and the number of trainable parameters for each evaluation dataset are shown in Table 1.

4.1 Accuracy of Locked Models
We have locked each model mentioned in Table 1 with a master key using Deep-Lock. Without loss of generality, we have considered
A key-based protection scheme Deep-Lock provides security for DNN models. Deep-Lock uses AES SBox and AES key-scheduling algorithm to lock the models. Figure 1a shows the original classification accuracies of the trained CNN models, the classification accuracies of the locked models for correct master key, the average accuracies of the locked models over 100 randomly selected incorrect keys. We can observe that Deep-Lock does not compromise the accuracy of the trained models. We can also observe that, for a wrong key, the average accuracy is not significant for MNIST and Fashion-MNIST models for classifying a single input. We can also observe that the overhead is not significant for MNIST and Fashion-MNIST. However, we obtain a comparatively higher overhead for the CIFAR-10 dataset. The figures are based on the sequential execution of software implementations of the DNN models. However, these can be significantly improved with high level of parallelization in dedicated hardware accelerators.

### Table 1: Datasets and Models

| Dataset          | Network Structure | Number of Parameters |
|------------------|-------------------|----------------------|
| MNIST            | C, MF, C, MF, FC, FC | 95,166               |
| Fashion-MNIST    | C, MF, C, MF, FC, FC | 180,438              |
| CIFAR-10         | C, MF, C, MF, FC, FC | 1,250,858            |

C: Convolution Layer, MF: MaxPool Layer, FC: Fully Connected Layer

![Figure 1: Classification accuracy of the original trained model, locked model with a correct key input, and locked model with a wrong key guess](image)

![Figure 2: Validation accuracies for each DNN model over training iterations during model fine-tuning](image)

In this paper, we propose a lightweight, generic, key-based DNN IP protection scheme Deep-Lock using an S-Box and key-scheduling algorithms to defend against unauthorized usage of stolen DNN models. The method ensures that only an authorized user with a correct master key can accurately use the locked DNN model, and a wrong master key will provide random classification. Deep-Lock does not modify any structural details of a DNN model, making it scalable for all existing software and hardware DNN implementations without adversely affecting performance. We evaluated Deep-Lock for various DNN architectures and datasets. We have also demonstrated its robustness against model fine-tuning attack.

### REFERENCES

1. Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th USENIX Security Symposium (USENIX Security 18). 1615–1631.
2. Manaar Alam, Sayandeep Saha, Debdeep Mukhopadhyay, and Sandip Kundu. 2022. NN-Lock: A Lightweight Authorization to Prevent IP Threats of Deep Learning Models. ACM Journal on Emerging Technologies in Computing Systems (JETC) 18, 3 (2022), 1–19.
3. Abhishek Chakraborty, Ankit Mondal, and Ankur Srivastava. 2020. Hardware-Assisted Intellectual Property Protection of Deep Learning Models. In Proceedings of the 57th Annual Design Automation Conference 2020.
4. Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. DeepSigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 485–497.
5. Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In 2018 IEEE/ACM International Conference on Computer-Aided Design (CCAD). IEEE, 1–8.
6. Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In 25th USENIX Security Symposium (USENIX Security 16). 601–618.
7. Hui Xu, Yuxin Su, Zirui Zhao, Yangfan Zhou, Michael R Lyu, and Irwin King. 2018. DeepCollusion: Securing the structure of convolutional neural networks via knowledge distillation. arXiv preprint arXiv:1806.10313 (2018).
8. Honggang Yu, KaiChen Yang, Teng Zhang, Yun-Yun Tsai, Tzuang-Yi Ho, and Yier Jin. 2020. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In Proceedings of Network and Distributed Systems Security Symposium.