NN-Lock: A Lightweight Authorization to Prevent IP Threats of Deep Learning Models

Manaar Alam

Secured Embedded Architecture Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Joint work with *Sayandeep Saha*, *Debdeep Mukhopadhyay* and *Sandip Kundu*

Deep Learning Popularity

- Deep Neural Network (DNN) has widespread commercial applications across industries
  - Can solve real-world tasks in which humans excel
  - US$ 37.98 billion industry by 2027[1]

[1] https://www.globenewswire.com/news-release/2020/08/13/2078051/0/en/Deep-Learning-Market-Reach-to-USD-37-98-Billion-By-2027-Industry-Share-Growth-Trends-Revenue-Analysis-Top-Leaders-Baumer-Optronic-GmbH-JAI-A-S-MVTec-Software-GmbH-Tordivel-AS-ISRA-.html

Deep Learning Popularity and Copyright

- Deep Neural Network (DNN) has widespread commercial applications across industries
  - Can solve real-world tasks in which humans excel
  - US$ 37.98 billion industry by 2027[1]
- Optimised DNN models are valuable storehouses of **Intellectual Property** (IP)
  - Huge monetary investments related to
    - Collecting labeled training dataset
    - Obtaining substantial domain expertise
    - Resources for optimised training
  - Data labeling is sometimes complicated, sensitive, and error-prone for domains like medical data analysis

[1] https://www.globenewswire.com/news-release/2020/08/13/2078051/0/en/Deep-Learning-Market-Reach-to-USD-37-98-Billion-By-2027-Industry-Share-Growth-Trends-Revenue-Analysis-Top-Leaders-Baumer-Optronic-GmbH-JAI-A-S-MVTec-Software-GmbH-Tordivel-AS-ISRA-.html

Deep Learning as a Service

Cloud-based MLaaS

Amazon AWS, BigML

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

Adversary tries to steal classified and highly efficient decision boundaries of DNN models

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

Adversary tries to steal classified and highly efficient decision boundaries of DNN models

- Adversary avoids the cost of expensive model training
- Immense economic loss for the service provider

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

Adversary tries to steal classified and highly efficient decision boundaries of DNN models

- Adversary avoids the cost of expensive model training
- Immense economic loss for the service provider

Model stealing is possible even in a black-box scenario through prediction API[1][2]

[1] F. Tramer et. al, “Stealing Machine Learning Models via Prediction APIs” — Usenix Security 2016
[2] T. Orekondy et. al, “Knockoff Nets: Stealing Functionality of Black-Box Models” — CVPR 2019

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

Adversary tries to steal classified and highly efficient decision boundaries of DNN models

- Adversary avoids the cost of expensive model training
- Immense economic loss for the service provider

Edge Device

Intel NCS, Google Coral

Model stealing is possible even in a black-box scenario through prediction API[1][2]

[1] F. Tramer et. al, “Stealing Machine Learning Models via Prediction APIs” — Usenix Security 2016
[2] T. Orekondy et. al, “Knockoff Nets: Stealing Functionality of Black-Box Models” — CVPR 2019

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

Adversary tries to steal classified and highly efficient decision boundaries of DNN models

- Adversary avoids the cost of expensive model training
- Immense economic loss for the service provider

Model stealing is possible even in a black-box scenario through prediction API[1][2]

Training in the cloud

Edge Device

Intel NCS, Google Coral

[1] F. Tramer et. al, “Stealing Machine Learning Models via Prediction APIs” — Usenix Security 2016
[2] T. Orekondy et. al, “Knockoff Nets: Stealing Functionality of Black-Box Models” — CVPR 2019

Image Credit: Google Images

Deep Learning as a Service and IP Threat

Cloud-based MLaaS

Amazon AWS, BigML

Model Stealing Attacks

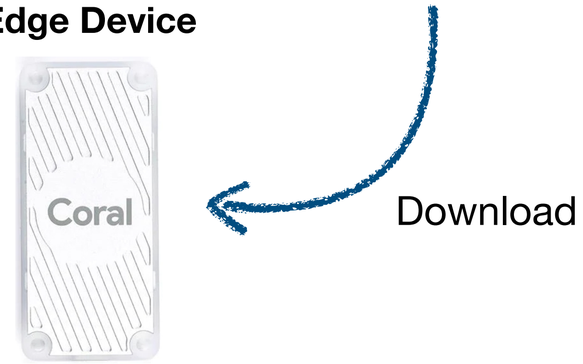
Adversary tries to steal classified and highly efficient decision boundaries of DNN models

- Adversary avoids the cost of expensive model training
- Immense economic loss for the service provider

Model stealing is possible even in a black-box scenario through prediction API[1][2]

Training in the cloud

Edge Device



Intel NCS, Google Coral

Protection of DNN IPs is more concerning as an adversary can have **physical access** to the downloaded proprietary DNN model

[1] F. Tramer et. al, “Stealing Machine Learning Models via Prediction APIs” — Usenix Security 2016
[2] T. Orekondy et. al, “Knockoff Nets: Stealing Functionality of Black-Box Models” — CVPR 2019

Image Credit: Google Images

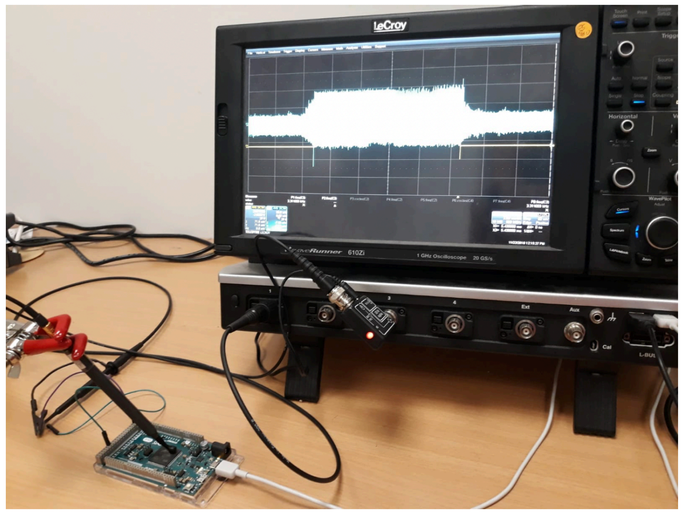
Model Extraction from Edge Devices

- A black-box DNN model in an edge device can be entirely reverse-engineered by observing electromagnetic (EM) side-channel[1][2][3]

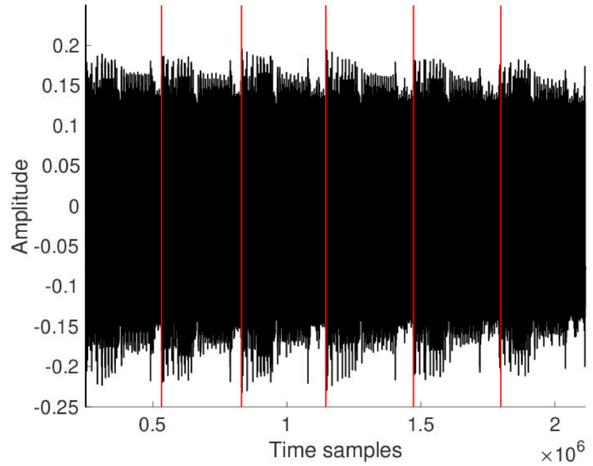
[1] L. Wei et. al, “I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators” — ACSAC 2018
[2] L. Batina et. al, “CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel” — Usenix Security 2019
[3] H. Yu et. al, “DeepEM: Deep neural networks model recovery through em side-channel information leakage” — HOST 2020

Model Extraction from Edge Devices

- A black-box DNN model in an edge device can be entirely reverse-engineered by observing electromagnetic (EM) side-channel[1][2][3]



Visual Inspection



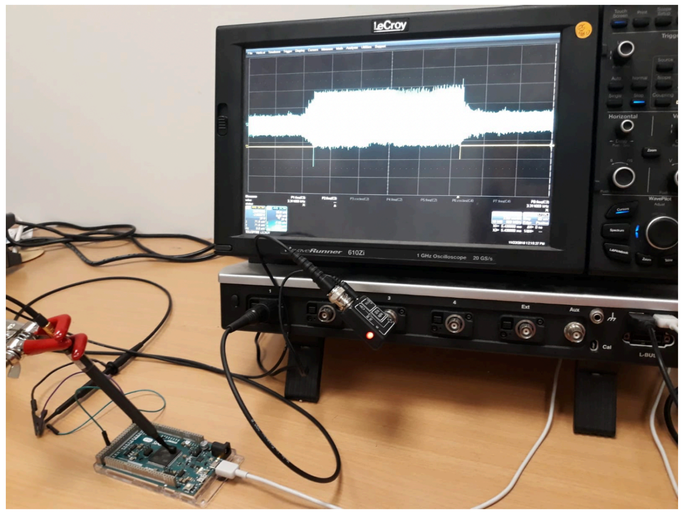
One hidden layer
6 neurons

Image Credit [2]

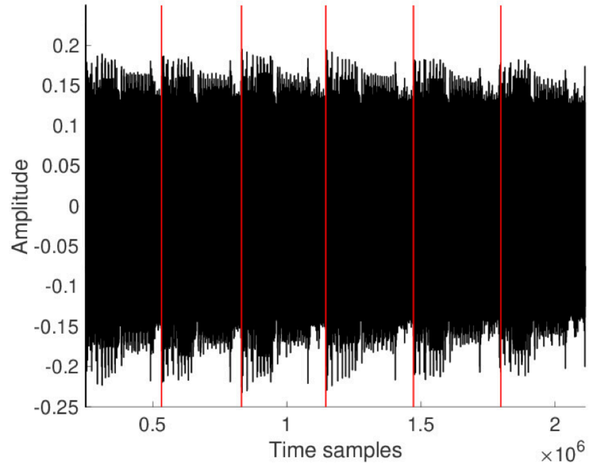
[1] L. Wei et. al, “I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators” — ACSAC 2018
[2] L. Batina et. al, “CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel” — Usenix Security 2019
[3] H. Yu et. al, “DeepEM: Deep neural networks model recovery through em side-channel information leakage” — HOST 2020

Model Extraction from Edge Devices

- A black-box DNN model in an edge device can be entirely reverse-engineered by observing electromagnetic (EM) side-channel[1][2][3]



Visual Inspection



One hidden layer
6 neurons

Image Credit [2]

Adversary can **extract network structure and parameters** or **redistribute the black-box model** as a form of illegal usage (or **model piracy**)

[1] L. Wei et. al, “I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators” — ACSAC 2018
[2] L. Batina et. al, “CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel” — Usenix Security 2019
[3] H. Yu et. al, “DeepEM: Deep neural networks model recovery through em side-channel information leakage” — HOST 2020

Protection against Model Extraction

- Model Watermarking
  - Embedding identification information into a given DNN model for establishing ownership[1][2][3]
  - Cannot prevent its illegal usage as an adversary can privately use the model

[1] Y. Adi et. al, “Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring” — Usenix Security 2018
[2] J. Guo et. al, “Watermarking deep neural networks for embedded systems” — ICCAD 2018
[3] B. D. Rouhani et. al, “DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks” — ASPLOS 2019

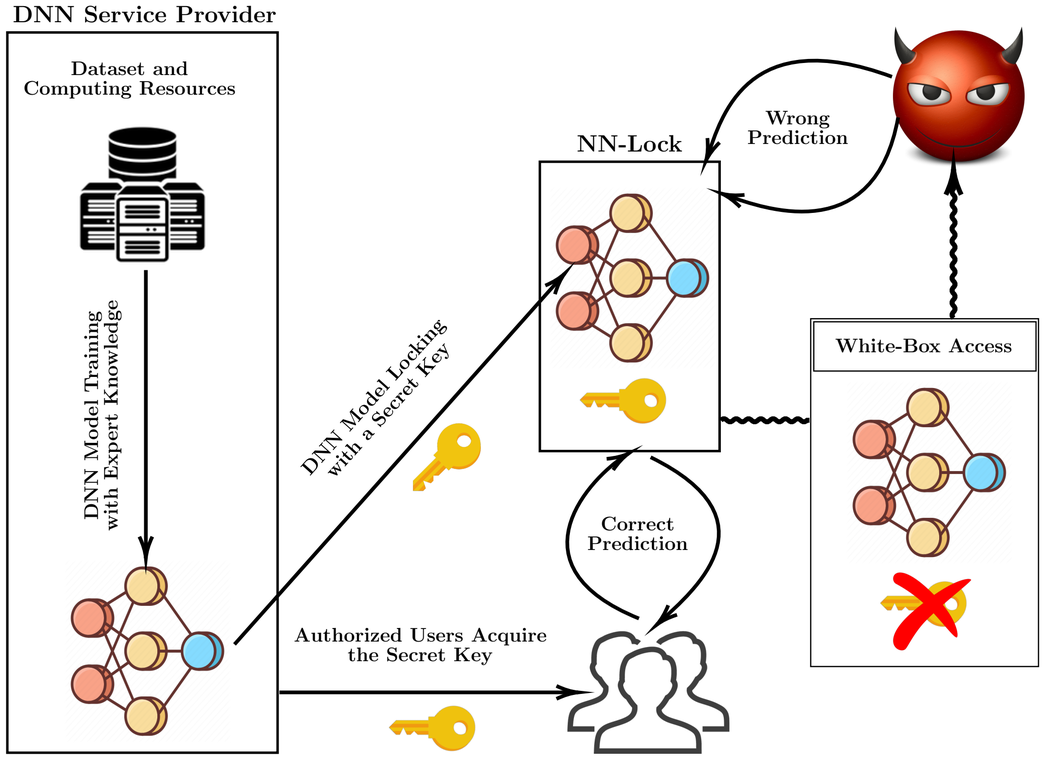
Protection against Model Extraction

- Model Watermarking
  - Embedding identification information into a given DNN model for establishing ownership[1][2][3]
  - Cannot prevent its illegal usage as an adversary can privately use the model
- Model Obfuscation
  - Obfuscate the model structures[4]
  - A more critical component of a DNN IP is the trained parameters
  - Most industrial applications typically use previously published DNN architectures but use different parameters depending on applications

[1] Y. Adi et. al, “Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring” — Usenix Security 2018
[2] J. Guo et. al, “Watermarking deep neural networks for embedded systems” — ICCAD 2018
[3] B. D. Rouhani et. al, “DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks” — ASPLOS 2019
[4] H. Xu et. al, “DeepObfuscation: Securing the Structure of Convolutional Neural Networks via Knowledge Distillation” — arXiv 2018

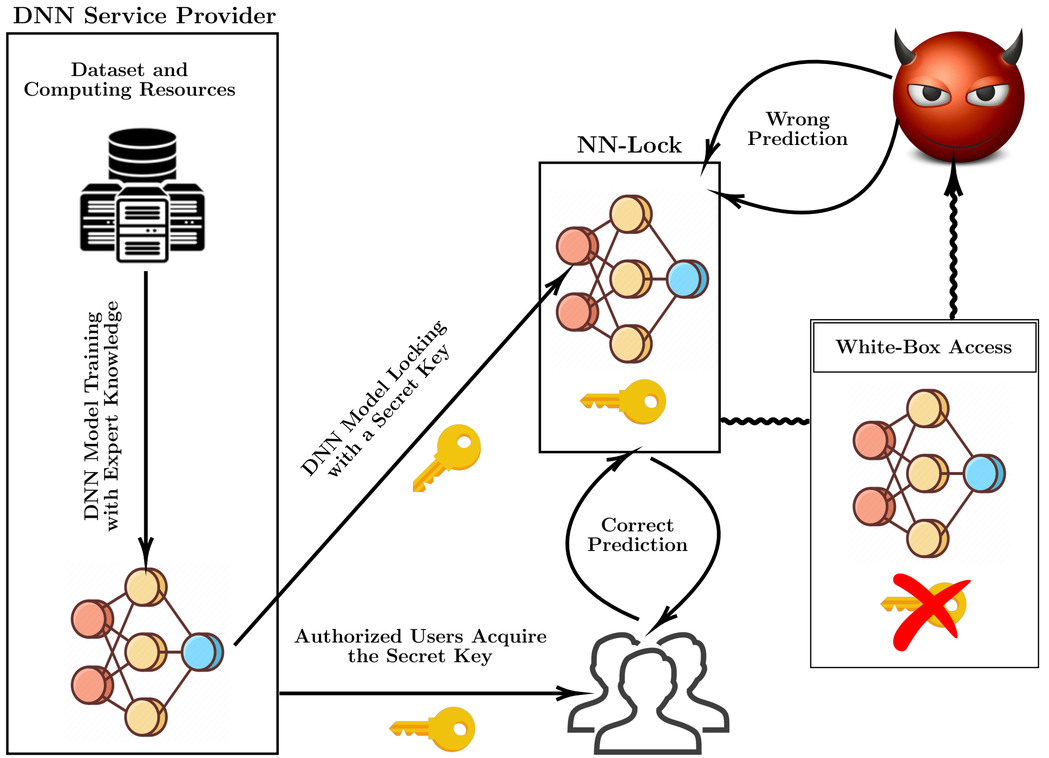
Idea of Authentication

- The main idea is to obfuscate DNN parameters in a secured manner using a secret key
- An adversary cannot make the model work correctly in the absence of a legitimate key even after obtaining obfuscated parameters
- A wrong key results in a significantly degraded model accuracy, which prevents the adversary from achieving any economic gain from the stolen model



Idea of Authentication

- The main idea is to obfuscate DNN parameters in a secured manner using a secret key
- An adversary cannot make the model work correctly in the absence of a legitimate key even after obtaining obfuscated parameters
- A wrong key results in a significantly degraded model accuracy, which prevents the adversary from achieving any economic gain from the stolen model



We address security issues in DNN IPs while they are distributed to a trusted valid user through edge devices

Capability of an Adversary

- The adversary has **white-box** access to the encrypted model
  - Knows the model structure and all encrypted parameter values
  - The only information the adversary does not know is the secret key used to encrypt the model
- The adversary has access to a **manifest dataset**
  - Can be used to query the extracted model with a key guess to obtain an output and compare the corresponding annotation
- The adversary has **adequate DNN expertise and computational resources**
  - Does not have the dataset for obtaining exceptional accuracy, like the model intended for stealing

NN-Lock: Overview

- **Model Training**
  - A DNN model owner can exercise any popularly used learning technique for fine-tuning model parameters or use a previously trained model

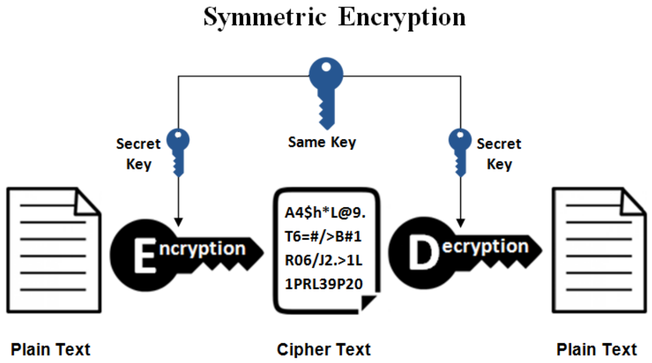
NN-Lock: Overview

- **Model Training**
  - A DNN model owner can exercise any popularly used learning technique for fine-tuning model parameters or use a previously trained model
- **Lock Parameters**
  - NN-Lock selects a master key and uses a **key-scheduling algorithm** to generate different secret keys corresponding to each trained parameter
  - The parameters are encrypted with an **S-Box operation** using the derived secret keys
  - The original model parameters are replaced with encrypted values before deploying it for practical usage

NN-Lock: Overview

- **Model Training**
  - A DNN model owner can exercise any popularly used learning technique for fine-tuning model parameters or use a previously trained model
- **Lock Parameters**
  - NN-Lock selects a master key and uses a **key-scheduling algorithm** to generate different secret keys corresponding to each trained parameter
  - The parameters are encrypted with an **S-Box operation** using the derived secret keys
  - The original model parameters are replaced with encrypted values before deploying it for practical usage
- **Distribute Model**
  - DNN model owner distributes the locked model to authorized customers
  - The secret key is embedded into a Trusted Platform Module (TPM) of the edge device

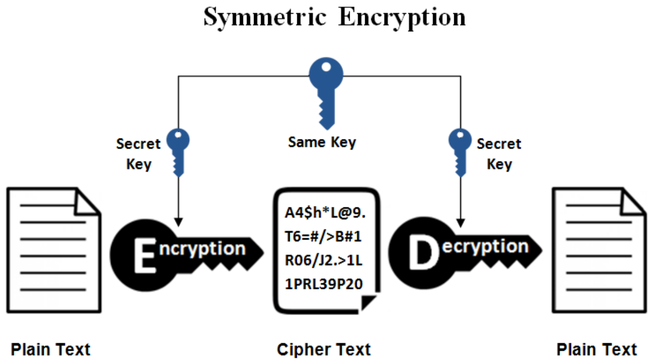
Basics of S-Box Operation



- In cryptography, an S-box (substitution-box) is a basic component of symmetric key algorithms which performs substitution
  - A non-linear Boolean mapping used to obscure the relationship between a secret key and ciphertexts.
- A good S-box provides good **confusion** property.
  - Each bit of the encrypted value should depend on several parts of the secret key.
  - This property ensures that if a single bit in the secret key is changed, most or all bits in the encrypted values will be affected (known as *Avalanche Effect*).

Image Credit: Google Images

Basics of S-Box Operation



- In cryptography, an S-box (substitution-box) is a basic component of symmetric key algorithms which performs substitution
  - A non-linear Boolean mapping used to obscure the relationship between a secret key and ciphertexts.
- A good S-box provides good **confusion** property.
  - Each bit of the encrypted value should depend on several parts of the secret key.
  - This property ensures that if a single bit in the secret key is changed, most or all bits in the encrypted values will be affected (known as *Avalanche Effect*).

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

AES S-box

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

AES Inverse S-box

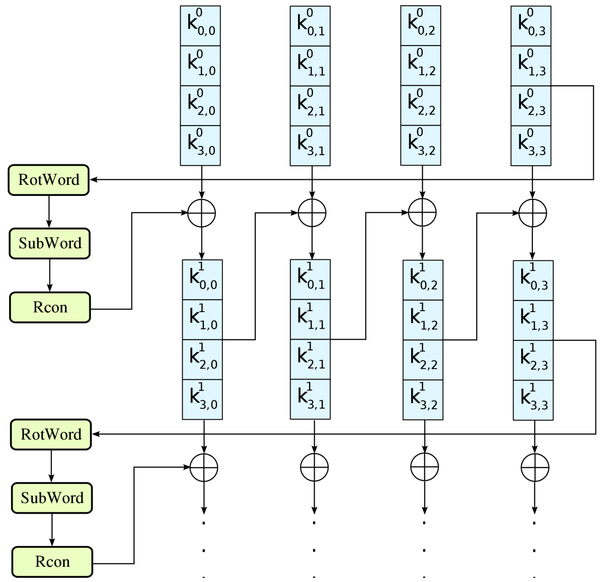
Image Credit: Google Images

Basics of Key Scheduling Operation

- In cryptography, block ciphers operate in multiple rounds. Key-scheduling algorithm takes a master key as input and generates different secret keys for each round
  - Reduces the space required for storing multiple keys
  - Using same key in each round has security vulnerabilities

Basics of Key Scheduling Operation

- In cryptography, block ciphers operate in multiple rounds. Key-scheduling algorithm takes a master key as input and generates different secret keys for each round
  - Reduces the space required for storing multiple keys
  - Using same key in each round has security vulnerabilities



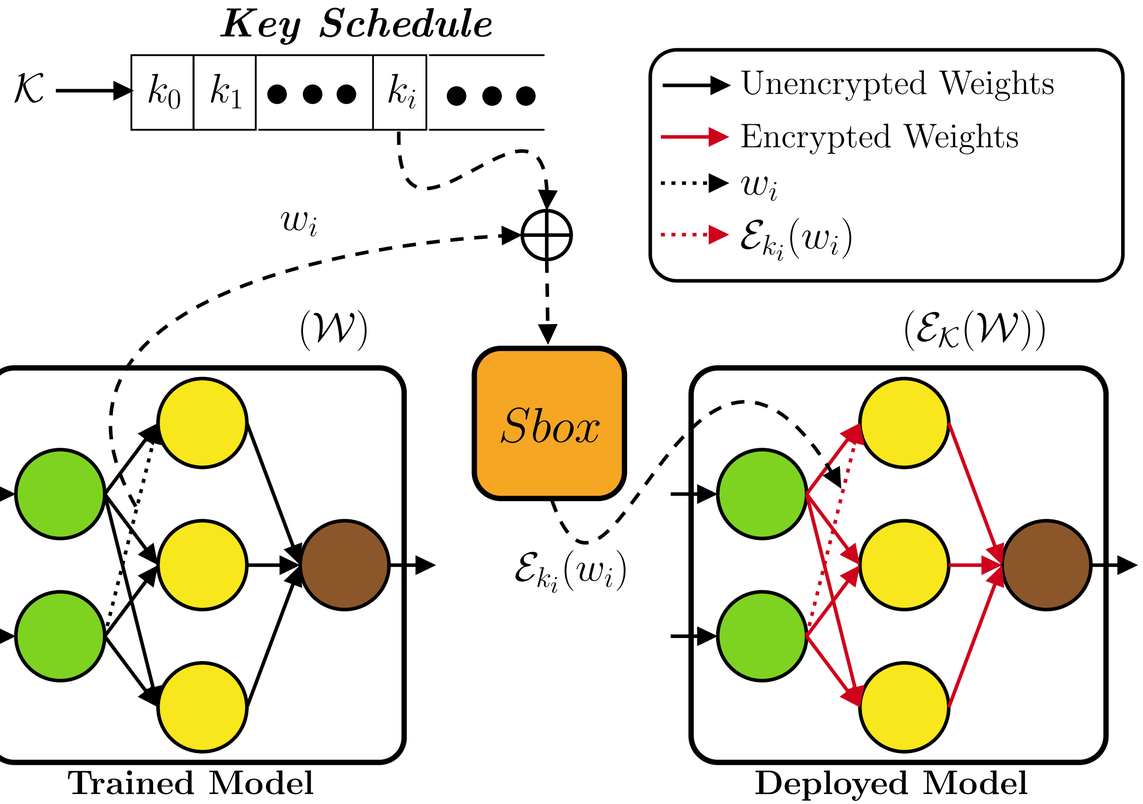
AES Key Schedule

Image Credit: Google Images

NN-Lock: The Lock Operation

- DNN models used in edge devices are trained in the cloud and downloaded after converting it into a model file with appropriate quantization for performance advantages

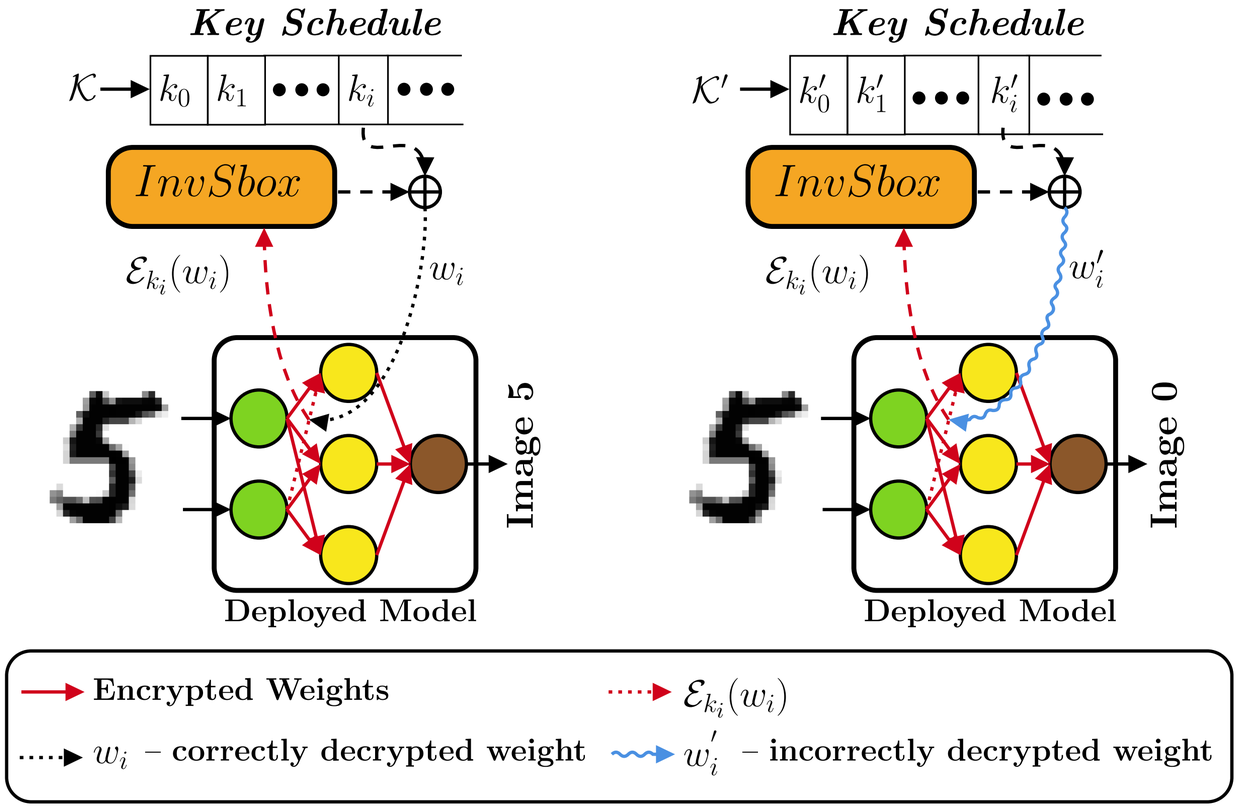
- Widely used quantization technique is full integer quantization, where all the model parameters are converted into 8-bit integers



NN-Lock: Runtime Prediction

- The locked model requires a master key during each query, which is obtained from the TPM of the edge device

- For a strong key-scheduling algorithm like AES, a single-bit difference in the master key will affect a significant portion of each round keys

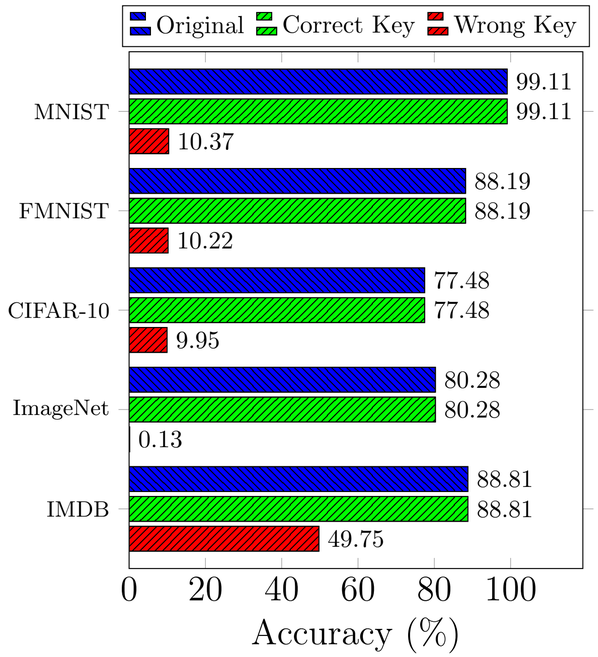


NN-Lock: Performance Evaluation

DATASETS AND MODELS

Dataset	# Class	Network Structure	# Parameters
MNIST	10	C, MP, C, MP, FC, FC	86,166
Fashion MNIST	10	C, MP, C, MP, FC, FC	180,438
CIFAR-10	10	C, C, MP, C, C, MP, FC, FC	1,250,858
ImageNet	1000	Inception V3 [22]	23,851,784
IMDB	2	E, C, MP, FC	350,751

C: Convolution Layer, **MP:** MaxPool Layer, **FC:** Fully Connected Layer, **E:** Embedding Layer



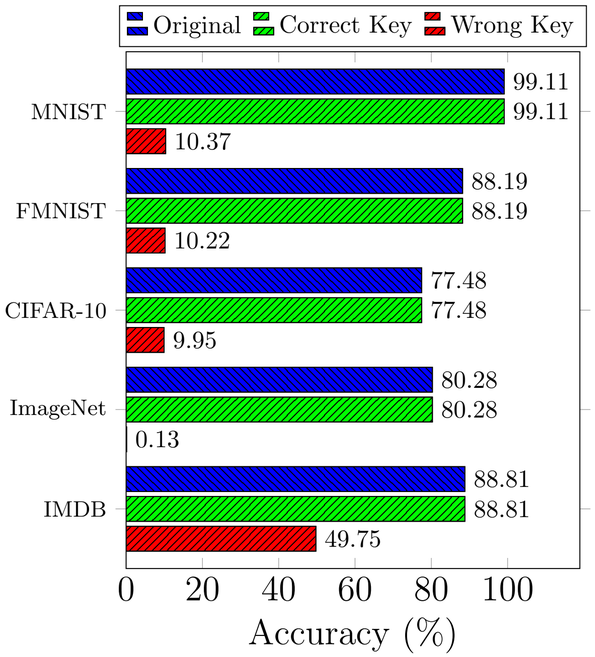
For wrong keys models work as random classifiers

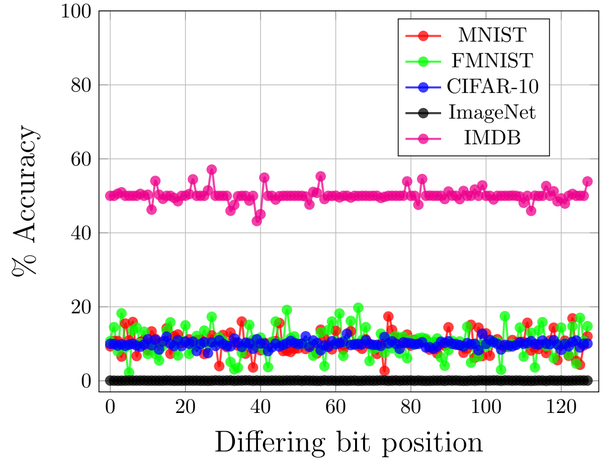
NN-Lock: Performance Evaluation

DATASETS AND MODELS

Dataset	# Class	Network Structure	# Parameters
MNIST	10	C, MP, C, MP, FC, FC	86,166
Fashion MNIST	10	C, MP, C, MP, FC, FC	180,438
CIFAR-10	10	C, C, MP, C, C, MP, FC, FC	1,250,858
ImageNet	1000	Inception V3 [22]	23,851,784
IMDB	2	E, C, MP, FC	350,751

C: Convolution Layer, **MP:** MaxPool Layer, **FC:** Fully Connected Layer, **E:** Embedding Layer





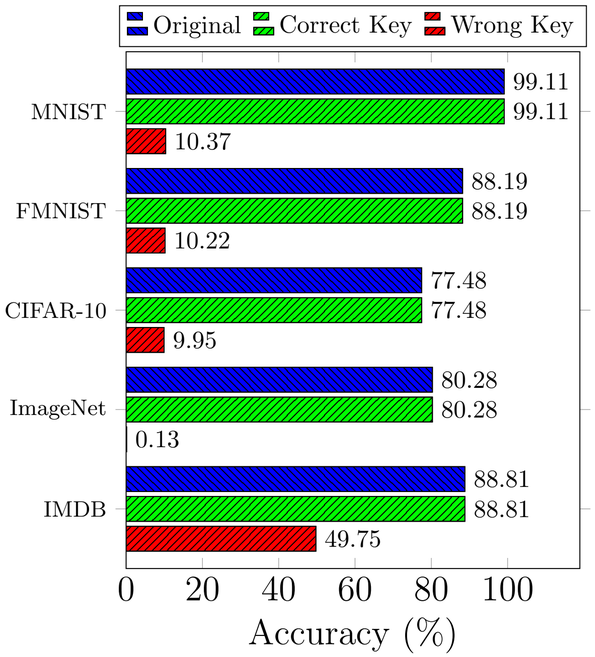
For wrong keys models work as random classifiers

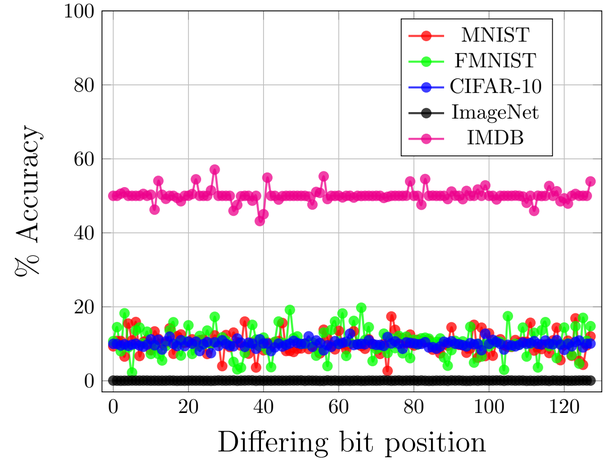
NN-Lock: Performance Evaluation

DATASETS AND MODELS

Dataset	# Class	Network Structure	# Parameters
MNIST	10	C, MP, C, MP, FC, FC	86,166
Fashion MNIST	10	C, MP, C, MP, FC, FC	180,438
CIFAR-10	10	C, C, MP, C, C, MP, FC, FC	1,250,858
ImageNet	1000	Inception V3 [22]	23,851,784
IMDB	2	E, C, MP, FC	350,751

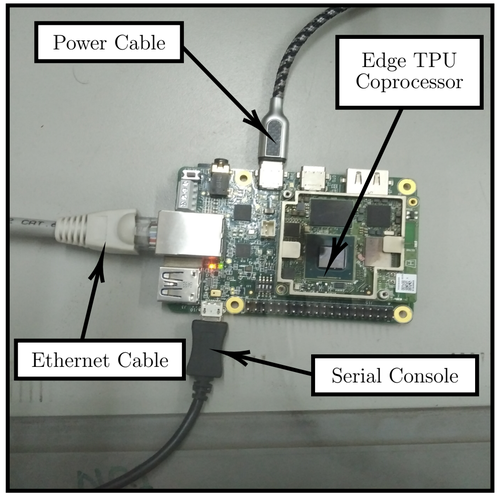
C: Convolution Layer, **MP:** MaxPool Layer, **FC:** Fully Connected Layer, **E:** Embedding Layer





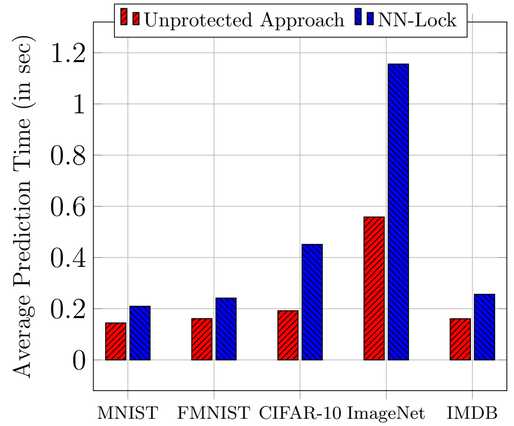
For wrong keys models work as random classifiers

Google Coral Dev Board



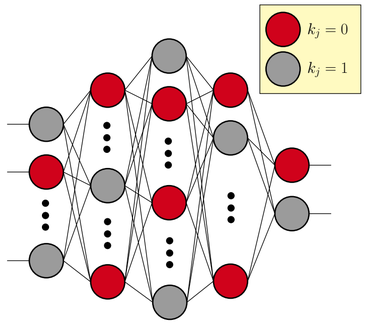
Experimental Setup

Inference Time

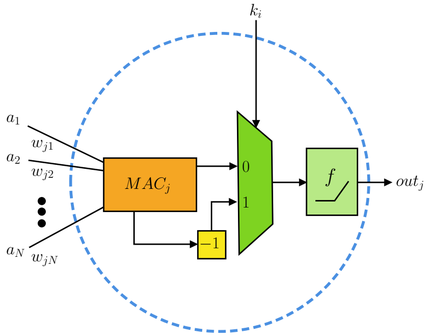


NN-Lock: Security Evaluation

Brief Overview of HPNN[1]



HPNN Framework



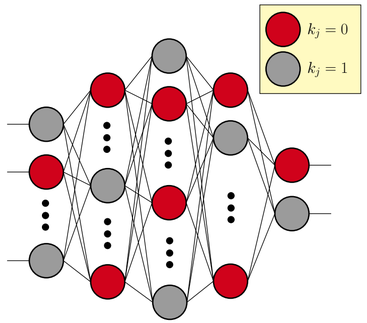
Obfuscation of Neurons

$$out_j = f((-1)^{k_j} MAC_j)$$

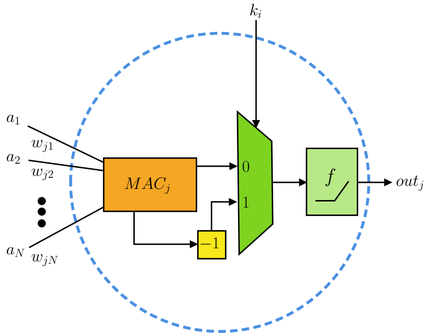
[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

Brief Overview of HPNN[1] and Security Analysis



HPNN Framework



Obfuscation of Neurons

$$out_j = f((-1)^{k_j} MAC_j)$$

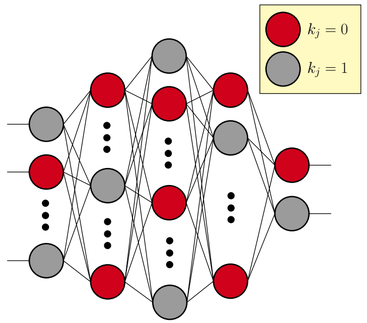
Approximation Attack

- A brute force search on the entire key space is impractical for a large key size.
  - Adversary can attempt a divide and conquer strategy to extract the key
  - The correct operation of a single neuron in a locked DNN does not ensure the correct operation of the complete DNN.
  - The adversary needs to unlock a combination of neurons simultaneously
  - A wrong key containing a significant sized sub-part of the correct key may provide better accuracy than a random guess

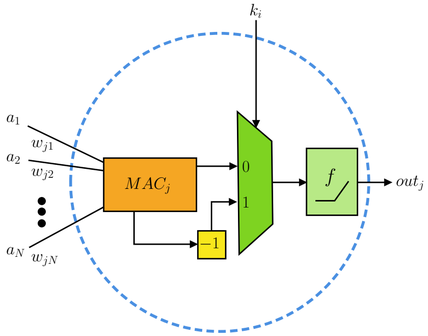
[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

Brief Overview of HPNN[1] and Security Analysis



HPNN Framework



Obfuscation of Neurons

$$out_j = f((-1)^{k_j} MAC_j)$$

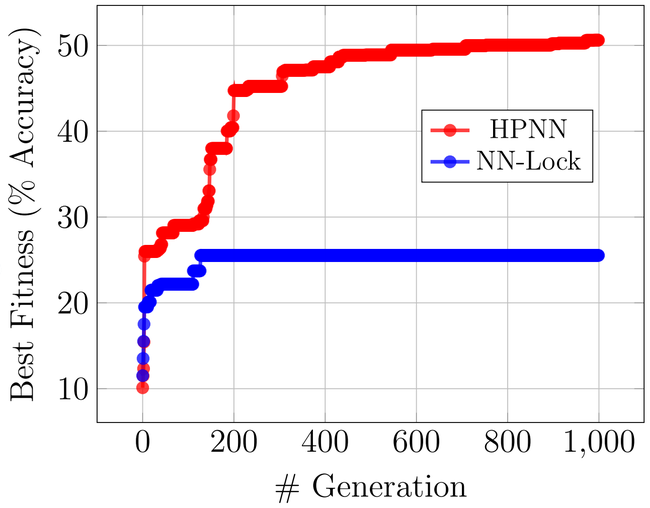
Approximation Attack

- Evolutionary algorithms often perform well when finding an approximate solution to a problem
- In NN-Lock, instead of locking each neuron, we encrypt all the trained parameters
  - A neuron is connected with multiple such parameters
  - The adversary needs to decrypt most or all the trained parameters connected to a neuron simultaneously to get the correct output from a single neuron
  - The use of cryptographic constructs to encrypt trained parameters and the use of key scheduling algorithm makes NN-Lock more robust against this type of approximation attack

[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

Approximation Attack

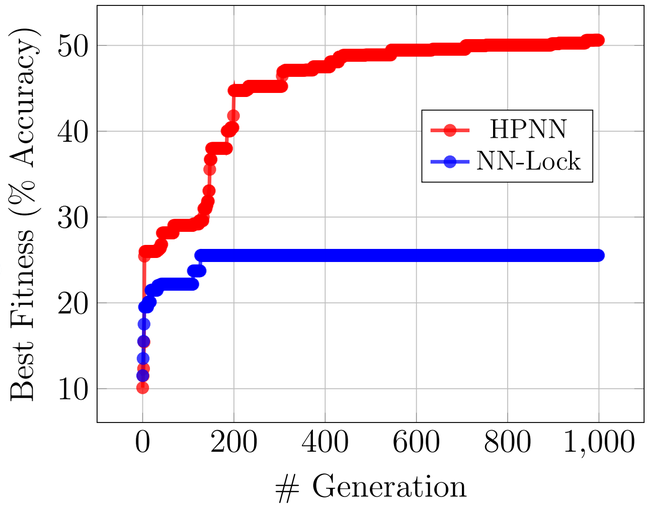


MNIST Dataset
DNN with
(784-100-100-46-10) architecture

[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

Approximation Attack



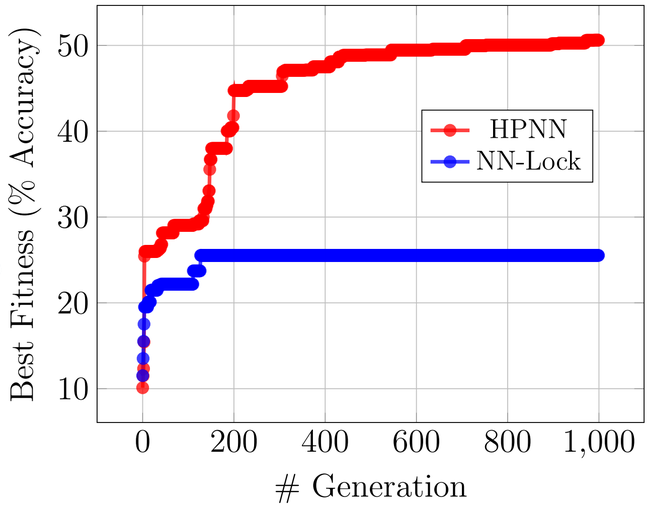
MNIST Dataset
DNN with
(784-100-100-46-10) architecture

We achieved better
security with 128-bit key.
HPNN had 256-bit key

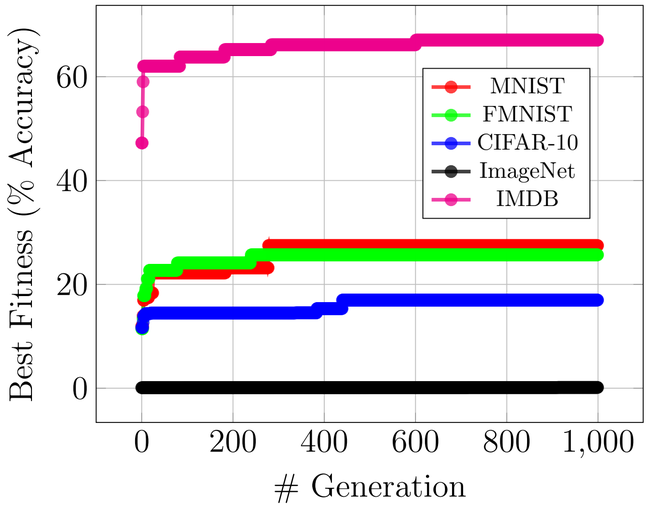
[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

Approximation Attack



MNIST Dataset
DNN with
(784-100-100-46-10) architecture



We achieved better
security with 128-bit key.
HPNN had 256-bit key

[1] A. Chakraborty et. al, “Hardware-Assisted Intellectual Property Protection of Deep Learning Models” — DAC 2020

NN-Lock: Security Evaluation

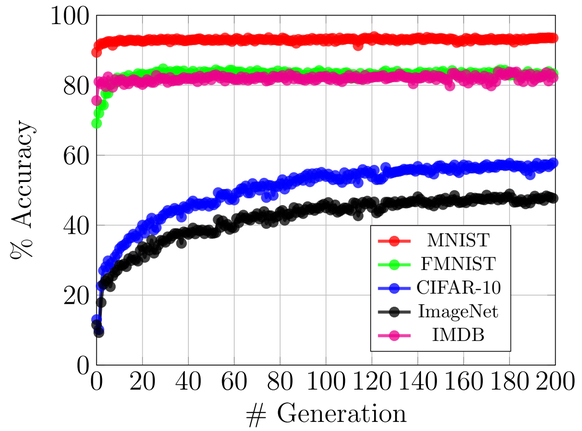
Model Fine-Tuning Attack

- Objective is to use the locked parameter of a DNN model and retrain them using the **manifest dataset** (10% of the original dataset) to achieve a comparative performance of the original model.

NN-Lock: Security Evaluation

Model Fine-Tuning Attack

- Objective is to use the locked parameter of a DNN model and retrain them using the **manifest dataset** (10% of the original dataset) to achieve a comparative performance of the original model.

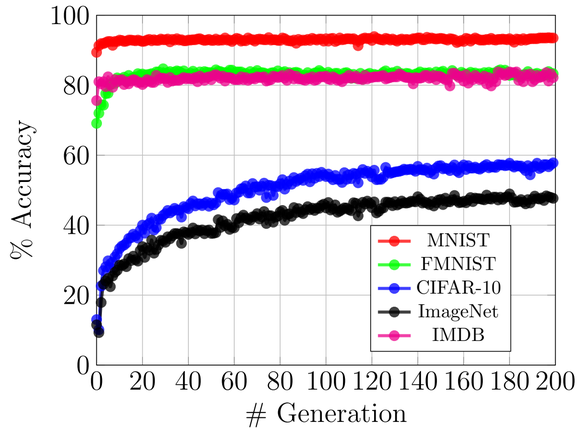


Trained with manifest dataset with random initialisation

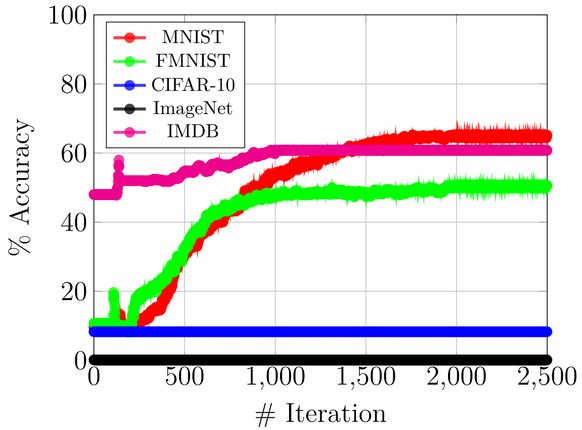
NN-Lock: Security Evaluation

Model Fine-Tuning Attack

- Objective is to use the locked parameter of a DNN model and retrain them using the **manifest dataset** (10% of the original dataset) to achieve a comparative performance of the original model.



Trained with manifest dataset with random initialisation

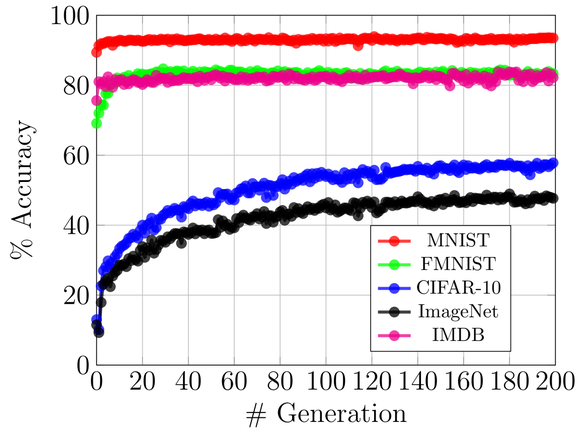


Trained with manifest dataset on model encrypted with NN-Lock

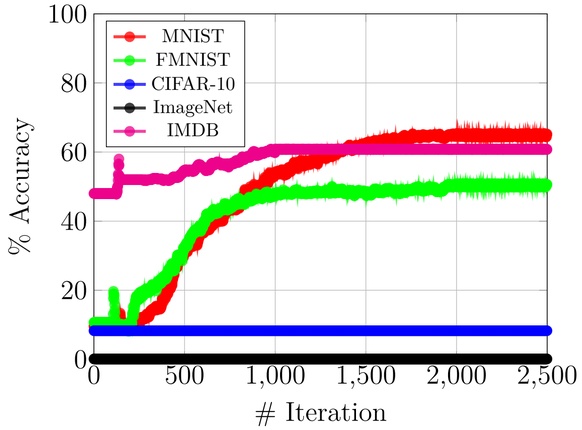
NN-Lock: Security Evaluation

Model Fine-Tuning Attack

- Objective is to use the locked parameter of a DNN model and retrain them using the **manifest dataset** (10% of the original dataset) to achieve a comparative performance of the original model.



Trained with manifest dataset with random initialization



Trained with manifest dataset on model encrypted with NN-Lock

HPNN reported 82.43% accuracy for FMNIST and 78.53% accuracy for CIFAR-10 with similar size dataset

NN-Lock: Security Evaluation

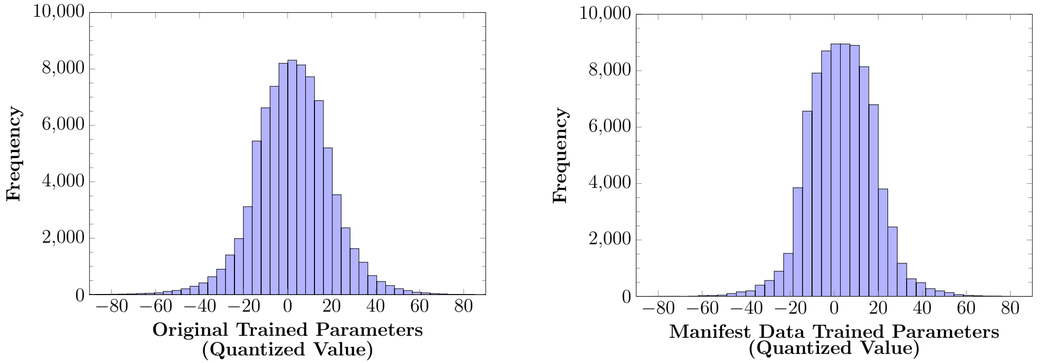
Key-Distinguishing Attack

- Parameters of a trained DNN model for different random initialization produce similar distributions though having different individual values

NN-Lock: Security Evaluation

Key-Distinguishing Attack

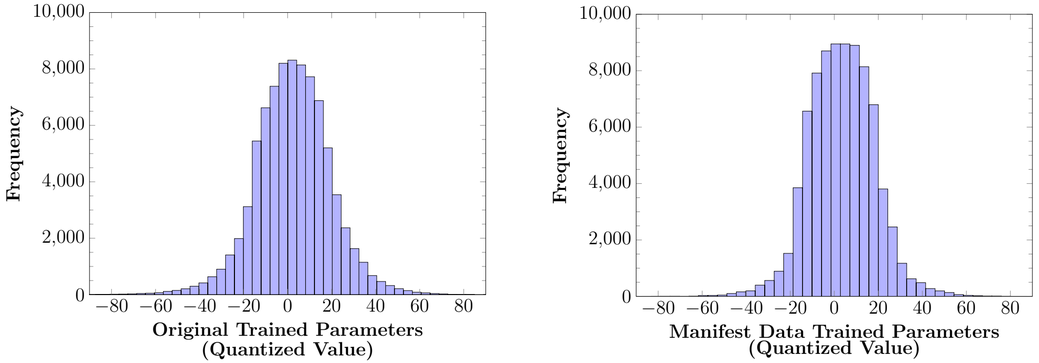
- Parameters of a trained DNN model for different random initialization produce similar distributions though having different individual values



NN-Lock: Security Evaluation

Key-Distinguishing Attack

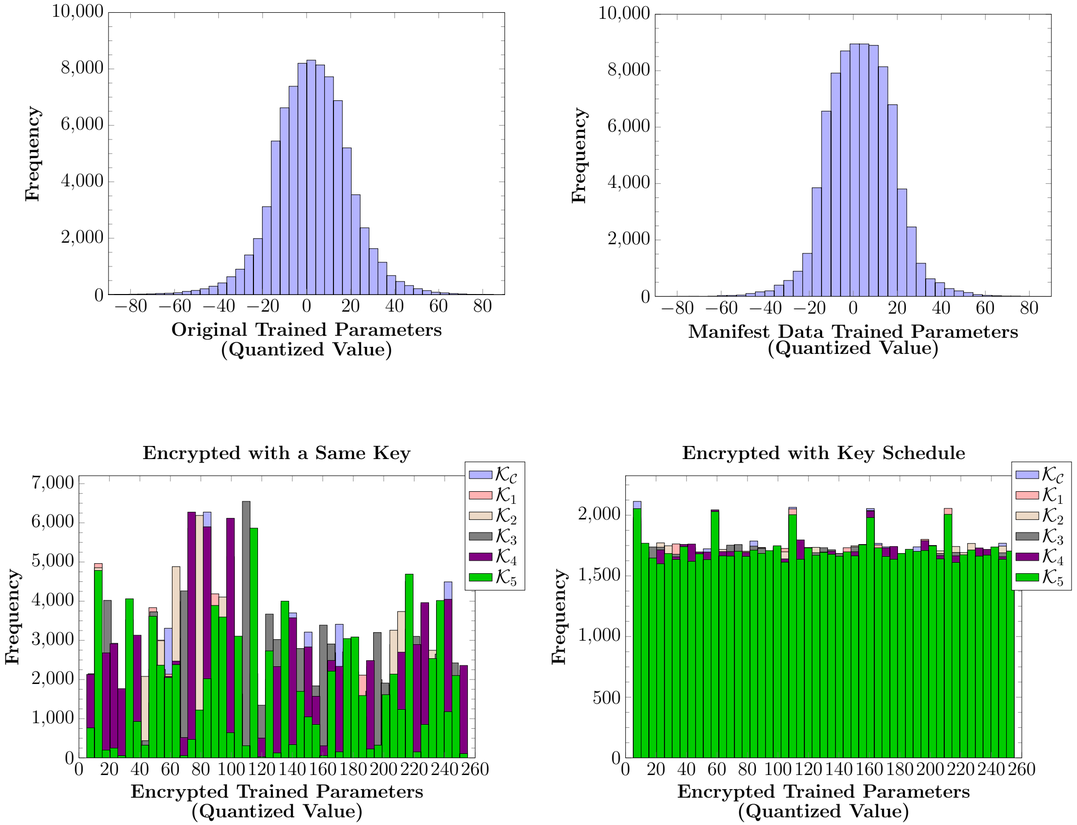
- Parameters of a trained DNN model for different random initialization produce similar distributions though having different individual values
- Frequency distribution similar to the unencrypted trained parameters can act as a distinguisher between different keys in this scenario



NN-Lock: Security Evaluation

Key-Distinguishing Attack

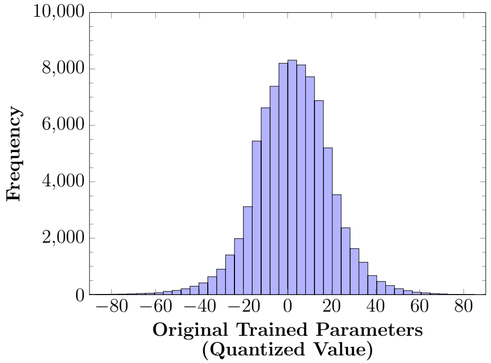
- Parameters of a trained DNN model for different random initialization produce similar distributions though having different individual values
- Frequency distribution similar to the unencrypted trained parameters can act as a distinguisher between different keys in this scenario

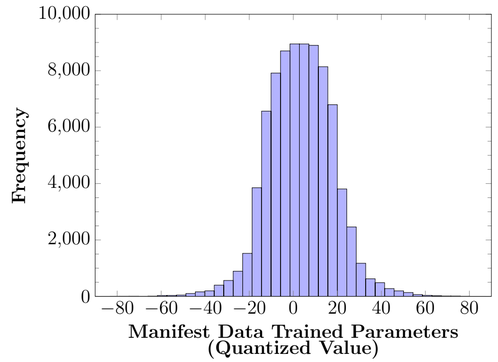


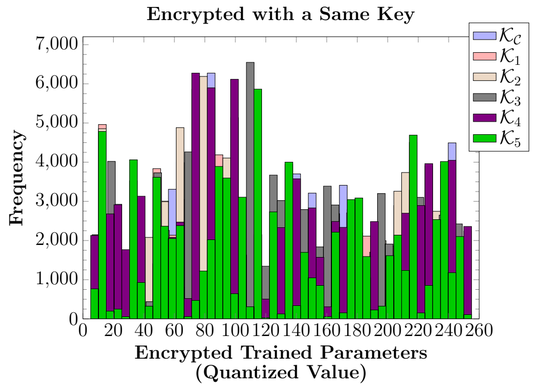
NN-Lock: Security Evaluation

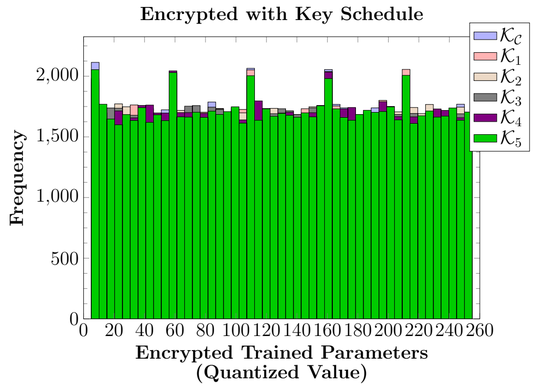
Key-Distinguishing Attack

- Parameters of a trained DNN model for different random initialization produce similar distributions though having different individual values
- Frequency distribution similar to the unencrypted trained parameters can act as a distinguisher between different keys in this scenario









The encryptions of two close parameters in the unencrypted domain with different keys disperse significantly in the encrypted domain.

NN-Lock: Security Evaluation

SAT/SMT-based Attack

- NN-Lock has a resemblance with so-called logic-locking schemes
- SAT/SMT-based techniques are quite popular for breaking logic locking constructions
- Success of such techniques largely depends on the problem's encoding in the underlying theory of the solver
- Designing SAT/SMT attack is still an active research topic in the context of complex DNN architectures[1]
- NN-Lock eventually results in a dense network of S-Boxes
  - Such a dense network of S-Boxes typically results in SAT-hard instances[2]

[1] G. Katz et. al, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks" — CAV 2017
[2] A. Saha et. al, "LoPher: SAT-Hardened Logic Embedding on Block Ciphers" — DAC 2020

Further Improvements and Future Works

- In this work, we address security issues in DNN IPs while they are distributed to a trusted valid user through edge devices
  - The trusted user can itself engage in model piracy by distributing the secret key
  - We are working on to use a *Physical Unclonable Function* (PUF) as a replacement of the secret key
    - PUF is “digital fingerprint” of a device which uniquely identifies it
- We have encrypted all the trained parameters of a DNN model, which increases the inference time for a large network 2x times
  - We are working on to use different *pruning algorithm* to remove redundant parameters from a DNN model without affecting accuracy
  - We believe the encryption of less number of parameters will decrease the overhead significantly

Thank You