DPA ATTACK ON AES USING SCA AND NEURAL NETWORKS

José Ricardo Rosales Castañeda, ITC A01709449, Tecnológico de Monterrey Campus Querétaro.

Abstract- **Differential Power Analysis (DPA) is a side-channel attack that exploits power consumption variations to infer cryptographic keys. In this work we investigated the vulnerability of the AES encryption algorithm against DPA attacks enhanced with neural network models. We used supervised learning, using Convolutional Neural Networks (CNN). Our model got to infer the encryption key in the testing set using only 298 traces. This study aims to show the growing threat of machine learning on side channel attacks.**

1. INTRODUCTION

The Advanced Encryption Standard (AES) is one of the most used symmetric encryption algorithms and is considered to be secure against the most common cryptographic attacks. However real-world implementations of AES on hardware or embedded systems, may leak some information through side channels, such as power consumption, making them vulnerable to side-channel attacks (SCA).

Differential Power Analysis (DPA) is a well known type of SCA attack, using statistical differences in power traces of some cryptographic operations such as encryption. Traditional DPA attacks rely on simple statistical models and require a large number of traces to infer the key.

Deep Learning techniques such as Convolutional Neural Networks (CNN), have shown the capacity of learning complex patterns in power traces and could potentially reduce the number of traces needed to successfully perform this type of attack.

In this paper, we present a study on applying a CNN model to perform a DPA attack on AES using the ASCAD dataset, which contains the traces from an Atmel ATmega8515 8-bit microcontroller.

This paper describes the attack methodology, the CNN structure, training setup and experimental results.

2. RELATED WORK

WIP

3. DATASET AND SETUP

For this project we used the **ASCAD (AES Side-Channel Attack Dataset)** which contains power consumption traces collected during the encryption process of AES. The traces are splitted into **profiling** and **attack** subsets, and are stored in ASCAD.h5 file, containing the

corresponding plaintexts, key bytes and metadata. This dataset is synced and noise free.

We loaded the data using the **h5py** library, getting the profiling traces and attack traces and classifying it as follows:

- Profiling set: Used for train and val
  - Traces: power traces.
  - Labels: labels of the traces.
- Attack set: Used to evaluate model
  - Traces: power traces.
  - Labels: labels of the traces.
  - Plaintexts: metadata for evaluation.
  - keys: metadata for evaluation.

To understand better the structure of the dataset we visualized the first five traces from the profiling dataset. As seen in the figure below each trace corresponds to the power consumption of AES encryption and consists of 700 time samples.

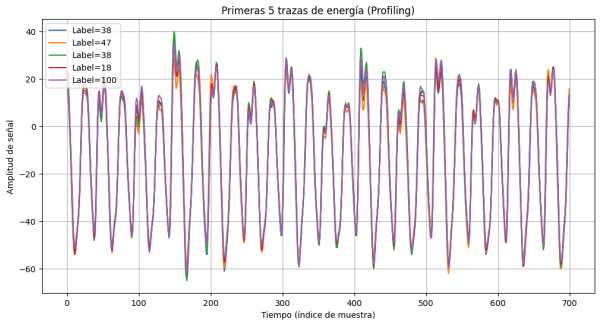

Figure 1: First five traces from profiling set

In figure 1 we can clearly see the structure of the data and how it behaves over time.

To ensure the model generalizes correctly, we split the profiling set into training and validation, corresponding each of 90% and 10% respectively, of the profiling set. The traces were scaled and reshaped to fit the input for the neural network. Additionally the labels were one-hot encoded into 256 classes, from 0 to 255.

4. METHODOLOGY

4.1. MODEL ARCHITECTURE

For this project we implemented a simple CNN architecture. CNNs characteristics make them a good fit for SCA tasks due to their ability to learn local patterns in time series data, such as power traces.

The model architecture is a simplification of the model VGG16. Our simplified model consists of:

- Input layer:
  - 1D array of 700 time series per race
- Convolutional layers:
  - Each layer has {32, 64} filters, kernel size of 11 and ReLu for activation.
- Drop layers:
  - Prevent overfitting
- Flatten layer:
  - Translates the output of the convolutional layers to a 1D vector for the dense layers.
- Output layer:
  - Output layer of 256 units and softmax activation, corresponding to each of the 256 values of the AES key bytes.

See diagram below:

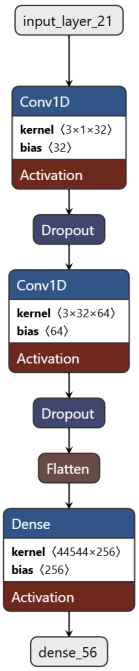

Figure 2: CNN architecture

4.2. TRAINING PARAMS

- Loss function: Categorical cross entropy
- Optimizer: adam
- Metrics: Accuracy
- Training setup:
  - Batch size: 200
  - Epochs 10

Model was trained on profiling set and evaluated on attack set using guess entropy to evaluate models performance

4.3. IMPLEMENTATION

WIP

4.4. EVALUATION

To evaluate the performance of the model, we used the traces from the attack set and computed the rank of the key over time.

This metric indicates the position of the correct key against the model's predictions. When a key gets to rank 0 means that the model inferred correctly the key

5. RESULTS

The accuracy obtained by the trained model looks as follows

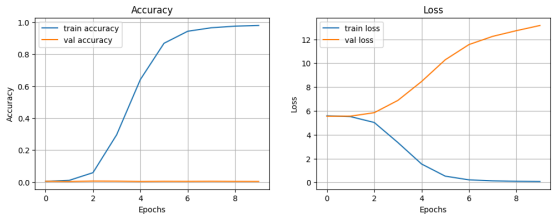

Figure 3: Train and validation Accuracy

From this plot we could infer the model had a poor performance in trying to perform the attack, but for DPA attacks the standard metric used for evaluating the attack performance is the Rank or guess entropy. The plot below shows the rank evolution across multiple traces.

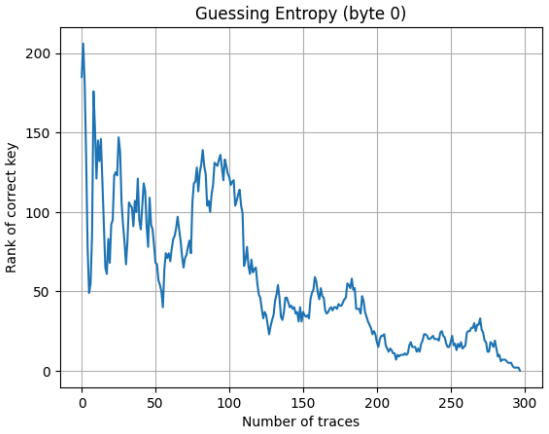

Figure 4: Rank of correct key

This result indicates that the model is capable of recovering the correct key with a small number of traces. Non AI-enhanced DPA attacks usually take between 1,000 and 10,000 traces in a similar setup.

6. CONCLUSION AND FUTURE WORK

WIP

REFERENCES