Attacking AES using Deep Learning

Amin Sarihi
New Mexico State university
Las Cruces, New Mexico
sarihi@nmsu.edu

Hunter Stuckey
New Mexico State university
Las Cruces, New Mexico
hss127@nmsu.edu

ABSTRACT

With the emerge of machine learning, encryption algorithms are not as safe as before. AES, which is considered as the standard in the industry, has been compromised by deep learning. A secret key can be inferred by analyzing the side channel information obtained from hardware executing an AES encryption algorithm. This imposes a real threat to information that is meant to be kept as secret as possible. The goal of this project is to extract the AES's secret key byte by byte form power traces obtained form Chipwhisperer using deep learning. Our results show that we need at least X and X traces for training and testing the classifier, respectively.

KEYWORDS

AES, Side-channel Attack, Deep Learning

1 INTRODUCTION

AES is a sound cryptography algorithm, and it has been proven safe. It is impossible to test all the $2^{keysize}$ possible keys in practice. On the other hand, side-channel information can be used as a shortcut to guess the right key. Side-channels are information leaked from the hardware implementation such as power trace, timing information, and electromagnetic waves [3]. An attacker can utilize this information to extract information about the encryption algorithm, which in turn would take a serious toll on the data security. Statistical approaches such as differential power analysis [4] and correlation power analysis [1] have been successful in recovering the key before; however, deep-learning attacks need fewer power traces to infer the secret key. Moreover, the attacker doesn't deal with statistical information when using this method [3].

There is a significant difference when one decides to choose between either deep learning or machine learning. Figure 1 shows a sample of a power trace obtained from an XMEGA128 device. The shape of the trace changes as the plaintext and the key changes. In case we utilize classifiers such as random forest and SVM, we need to figure out where our points of interest are, which we call POI hereafter. It could i.e., start from sample 1000 to 2000 of each trace. On the other hand, we can apply convolutional filters to traces, and it will automatically extract the POI[3]. Our supervised machine learning method consists of feeding the traces to a CNN with a variety of different properties to recover the key eventually.

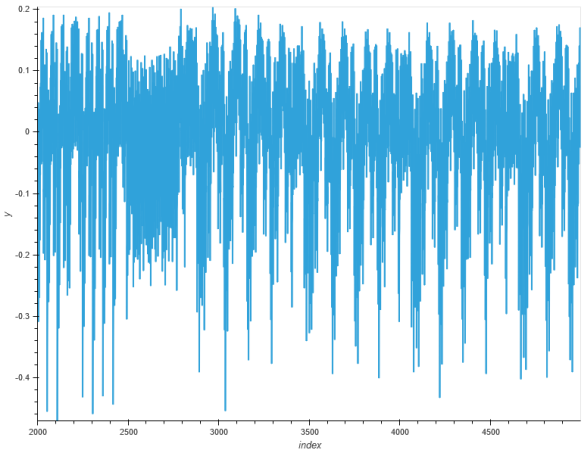

Figure 1: Power trace obtained from XMEGA128 running AES

2 BACKGROUND

2.1 AES

AES is a subset of the Rijndael block cipher proposed by two cryptographers, Vincent Rijmen and Joan Daemen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192, and 256 bits [2]. In this project, we will use the 128-bit version. The first two stages of AES contain the most information for recovering the key. These two stages are called AddRoundKey and SubByte. In the AddRoundKey, the plaintext is X-ored with the secret key. Next, in the SubByte stage, the first byte of the xor result is fed to a Sbox, and the data is substituted. Sbox is typically used to obscure the relationship between the key and the ciphertext. Figure 2 shows the first two stages of the AES Algorithm.

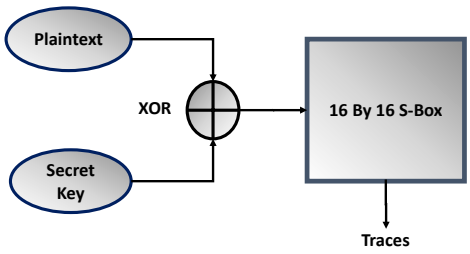

Figure 2: The traces are the the output of the S-Box stage

2.2 Side-channel attacks

Side channel attacks are done based on the hardware leakage information rather than the algorithm itself[6]. The most trivial example would be trying to guess what number you're dialling based on the tone sounds. In the hardware security field, logging and analyzing this information is of utmost importance to the researchers.

With exploiting the computation power of GPUs, analyzing the side channel has become easier than before and new opportunities are created to get more accurate and faster results.

3 OUR METHOD

Our method is based on capturing the S-Box output traces. Due to measurement variances, the captured traces could be desynchronized. In other words, they are not time-aligned. By using CNN we can skip the problem owing to the fact that we will use convolutional filters. Otherwise, we should have used some techniques to make align them. Figure 4 and figure 5 show 10 samples of power traces and our hardware setup to capture the trace which are the input to our CNN.

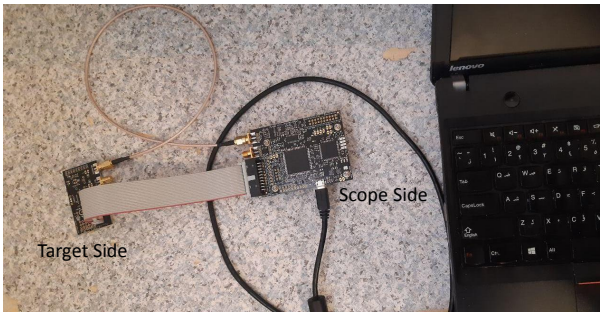

Figure 3: Chipwhisperer [5] setup for capturing power traces

```
6.054687500000000000e-02
-2.646484375000000000e-01
-1.494140625000000000e-01
-1.523437500000000000e-01
1.953125000000000000e-03
-3.925781250000000000e-01
-2.275390625000000000e-01
-2.080078125000000000e-01
-3.320312500000000000e-02
-4.257812500000000000e-01
```

Figure 4: 10 samples of a single trace

In AES128, the key and the plain text are 128 bits, which is 16 Bytes. In our approach, we will recover the key byte by byte. So each power trace corresponds to a specific key and plaintext pair, which are generated randomly. We capture the traces with know key and plaintext pairs and train our CNN. Then, we'll use the trained model to test some traces in which the plaintext is known, but the key is unknown. The computer and the board communicate through a VirtualBox image run on Oracle VM VirtualBox. This image contains the necessary compilers to program the board and capture the traces. The required steps to do this has been put in a jupyter notebook file in the repository. After we save the traces which are numpy arrays, we can store them in separate files to use them in a python code.

REFERENCES

[1] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*. Springer, 16–29.

[2] Joan Daemen and Vincent Rijmen. 1999. AES proposal: Rijndael. (1999).

[3] Sunghyun Jin, Suhri Kim, HeeSeok Kim, and Seokhie Hong. 2020. Recent advances in deep learning-based side-channel analysis. *ETRI Journal* (2020).

[4] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual International Cryptology Conference*. Springer, 388–397.

[5] Colin O'Flynn and Zhizhang David Chen. 2014. Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 243–260.

[6] Yuval Yarom and Katrina Falkner. 2014. FLUSH+ RELOAD: a high resolution, low noise, L3 cache side-channel attack. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 719–732.