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 rise 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 can recover the first key byte with a probability of

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 short-cut to guess the right key. Side-channels are information leaked from the hardware implementation such as power trace, timing information, and electromagnetic waves [5]. 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 [6] and correlation power analysis [2] 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 [5]. 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, since each part of the trace is correlated with the hamming weight that portion[4].

Using Deep learning algorithms provides us with a some advantages such:

- Points of interest selection are no longer necessary; however, we need to make sure that we feed the least possible traces to the DNN to lessen the training time.
- Convolutional layers can extract features independently of their position in the data.[5]

In general, we train an MLP (multi-layer perceptron) and a CNN (convolutional neural network) with our data which will be explained in chapter 3. The goal of the project is to feed the trained DNN with a set of unseen traces and try to recover the first byte of the whole 16-byte key.

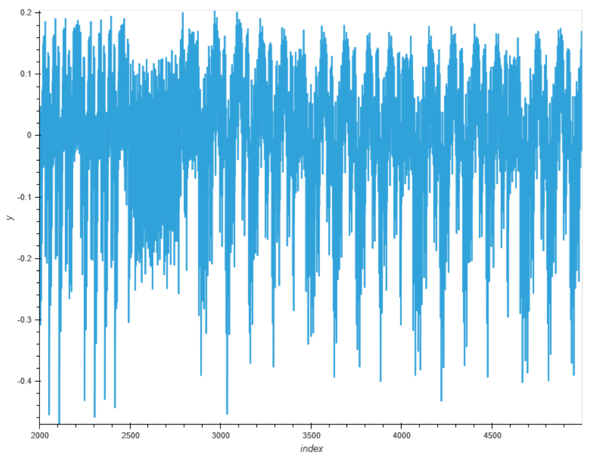


Figure 1: Power trace obtained from XMEGA128 running AES

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 [3]. 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 output of xor result is fed to aSbox, and the data is substituted. A full animated steps of AES can be found in [9]. 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.

This is where the algorithm's leakage point lays. In other words, the traces captured at this point contain the most useful information to exploit. There are other leakage points such as AES Inv SBox output which is not within the scope of this project.

2.2 Side-channel attacks

Side channel attacks are done based on the hardware leakage information rather than the algorithm itself[8]. The most trivial example would be trying to guess what number you're dialling on your phone based on the tone sounds. In the AES context, electromagnetic emissions and power comnsumption traces are used

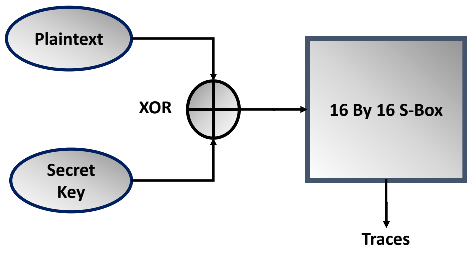


Figure 2: The traces are the the output of the S-Box stage

for analysis. With the advent of GPUs and open-source software libraries for machine learning such as Tensorflorw[1], training and testing process of DNN (deep neural networks) has become fatser than ever. Since brute forcing AES with computationally impossible, researchers have utilized side channels to recover the key. Our attack on AES is categorized as profiling side channel attacks which are actually the most powerful ones. They assume that the adversary acquires a copy of the final target to precisely tweak all the parameters to tune into best model to attack. **??**

3 OUR METHOD

Our method is based on capturing the S-Box output traces using Chipwhisperer. Our setup is shown in figure 3. Our target board is an XMEGA128, which is running tinyAES**??**. Each captures a set of metadata, such as the used keys and plaintexts. The steps include capturing N traces with t samples each. During the capture stage, we fix the key-value and obtain different waveforms with different plaintexts. Further, we will label each trace with its first stage SBox output. Figure 4 shows ten samples of a power trace, which is the input to our DNN. Due to measurement variances, the captured traces could be misaligned. We can feed the traces directly to an MLP (multi-layer perceptron) provided that they are time-aligned. Otherwise, we need to use a convolutional network with a convolutional filter at its input.

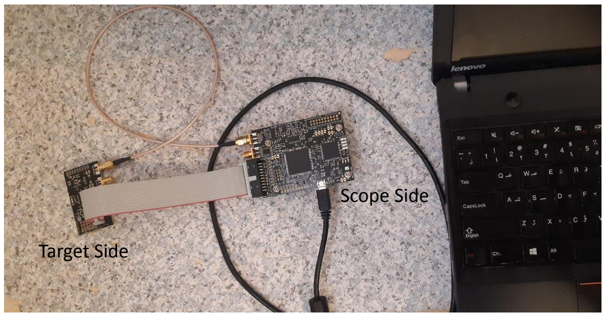
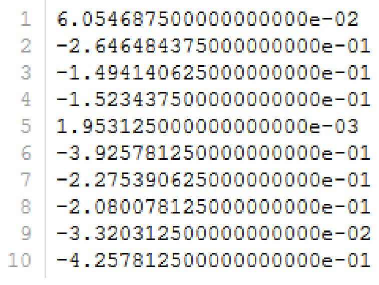


Figure 3: Chipwhisperer [7] setup for capturing power traces

Since we want to use as few data as possible to break the key, first, we start with the first 500 samples of the data and cross-verify our attack with correlation power analysis (CPA)**??** The DNN architectures we use are depicted in figures 5 and 6, respectively. MLP and CNN are for training and testing data.



```
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

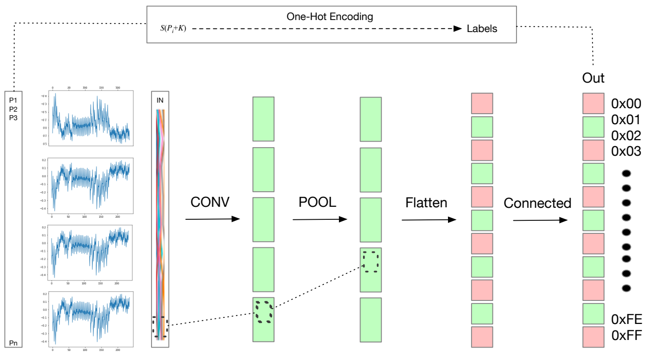


Figure 5: CNN architecture used in our method

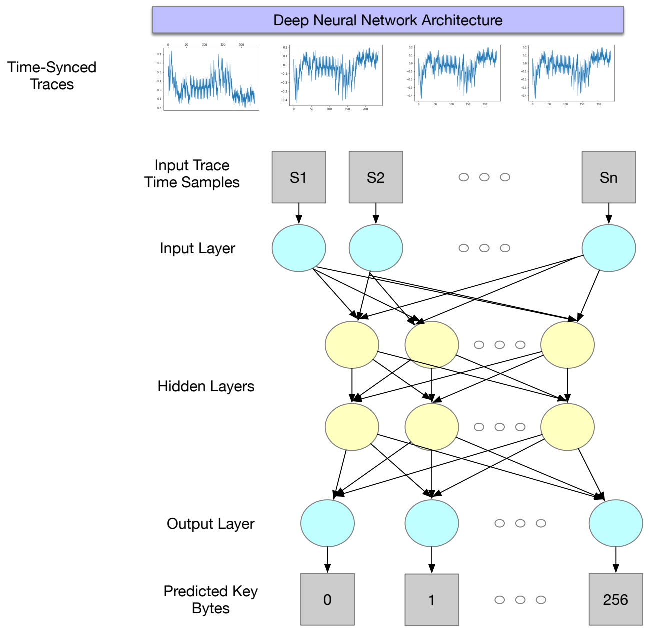


Figure 6: MLP architecture used in our method

In AES128, the key and the plaintext are 128 bits, which is 16 Bytes. We are targeting the first bute of the key in this project. For the rest 15, we need to train 15 other model which correspond to each key byte separately. Each power trace corresponds to a specific key and a random plaintext pair. 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 libraries and compilers to program the scope and the target. The required steps to do this have been put in a jupyter notebook file in the repository. After we save the traces which are in NumPy arrays format, we can store them in separate files to use them in our python code. The codes and the dataset have been uploaded to a Github repository**??**.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] 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.

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

[4] Jasper van Woudenberg Guilherme Perin, Baris Ege. 2018. Lowering the bar:deep learning forside-channel analysis. (2018).

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

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

[7] 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.

[8] 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.

[9] Enrique Zabala. 2008. The Rijndael Animation. http://www.formaestudio.com/rijndaelinspector/.