On Use of Deep Learning for Side Channel Evaluation of Black Box Hardware AES Engine

Yoo-Seung Won(✉) and Shivam Bhasin

Temasek Laboratories, Nanyang Technological University, Singapore, Singapore
{yooseung.won,sbhasin}@ntu.edu.sg

Abstract. With the increasing demand for security and privacy, there has been an increasing availability of cryptographic acclerators out of the box in modern microcontrollers, These accelerators are optimised and often black box. Thus, proper evaluation against vulnerabilities like side-channel attacks is a challenge in absence of architecture information and thus leakage model. In this paper, we show the use of deep learning based side-channel attack can overcome this challenge, allowing evaluation of black box AES hardware engine on a secure microcontroller, without the knowledge of precise leakage model information. Our results report full key recovery with only 3,000 traces under a profiling setting.

Keywords: Hardware AES engine · Side-channel analysis · Deep learning

1 Introduction

With the rise in need for security and privacy across applications, reliance on cryptography is ever growing. As a result, manufacturers integrate more and more cryptographic functions in modern microcontrollers to facilitate design of secure applications. For high performance applications, cryptographic functions are often available as in-built accelerators, accessible through an API. While these accelerators are secure in a classical setting, implementation security remains a concern. These accelerators may be used in sensitive applications requiring protection against attacks like side-channel attacks (SCA [8]) or faults attacks [13]. Thus, it must be carefully evaluated against such attacks when necessary. However, most if not all, accelerators are proprietary in nature and their architecture and related details are not available in public domain, making evaluation difficult. For instance, popular SCA like correlation power analysis is performed with a leakage model assumption [4]. If the leakage model is not precise, CPA is sub-optimal and may misguide security evaluation. The leakage model is better understood with knowledge of the architecture, which are not available in this setting.

N.-S. Vo et al. (Eds.): INISCOM 2021, LNICST 379, pp. 185–194, 2021.
https://doi.org/10.1007/978-3-030-77424-0_15

In this paper, we investigate the side-channel security of a black-box hardware AES engine on a commercial off the shelf microcontroller. The target microcontroller is recommenced for security critical applications like point of sale transactions. We demonstrate that using a deep learning based side-channel attack can allow better evaluation in the black box setting as compared to attacks like CPA where a precise leakage model is required.

The rest of the paper is organised as follows. Section 2 provides general background on SCA and deep learning for SCA. Section 3 describes the target device and evaluation platform. Section 4 reports experimental results and conclusions are drawn in Sect. 5.

2 Preliminaries

In this section, we provide background information on side-channel attacks (SCA) and use of deep learning for SCA.

2.1 Side-Channel Attacks

Side-Channel Analysis or Attacks (SCA) are a class of implementation level attacks which observe and exploit unintended physical leakages from target devices to gain information on underlying sensitive data. In context of cryptography, SCA aim at recovering the underlying secret key. The information can be observed by different channels including power consumption, electromagnetic emanation, timing, *etc.*

SCA can be widely classified as profiled and non-profiled. A profiled attack assumes a strong attacker with access to a clone device. By measuring traces corresponding to known plaintext and key, the adversary characterizes a model of the target device. On the victim device, the adversary captures only a few traces (ideally 1) with known plaintext but the key is unknown. These traces are then compared to the characterized model obtained from clone device to learn information on the secret key used by victim device. Initially, Gaussian templates [6] were used for model characterization but later machine learning and deep learning [1,11,12] were also shown to advantageous for profiled SCA.

Non-profiled attack on the contrary are directly applied on victim device, where adversary has access to plaintext or ciphertext but key is secret. Based on a leakage model like Hamming distance or Hamming weight, the adversary predicts a sensitive intermediate leakage value which depends on a part of secret kay (8-bits for AES) and known plaintext/ciphertext. The adversary test dependency of actual measurement with predicted leakage based on leakage model and all key hypothesis, using statistical tools. The correct key hypothesis is expected to show maximum dependency. In this work, we use Pearson correlation ρ as a statistical tool [4] to perform a correlation power analysis (CPA).

2.2 Deep Learning Based SCA

Recent profiled SCA have seen the application of deep neural networks [5,9]. Especially, convolution neural network (CNN) architectures are shown to be powerful for breaking countermeasures such as hardware jitter [5], shuffling [14], and masking [15]. Recent finding report CNN structures on various SCA open datasets [15] outperform the classical profiled SCA such as template attack [6].

3 Target Board and Setup

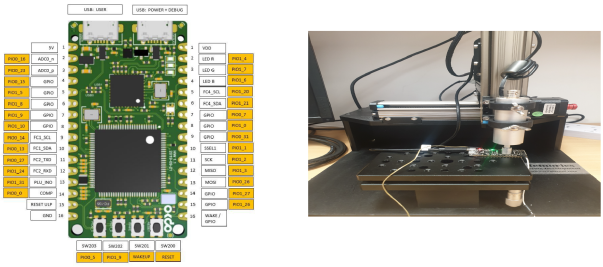
In this section, we report the target board and experimental setup. We target the Okdo E1 development board which is based on ARM Cortex-M33 chip.

The OKdo E1 development board which is an ultra-low-cost Development Board based on the NXP LPC55S69JBD100[1] dual-core Arm Cortex® M33 MCU. The intented application for this board are security sensitive like point-of-sale terminal. The security features of LPC55S69JBD100 are explained in the user manual [10]. It contains several hardware IPs such as an AES engine, a SHA engine, a random number generator, a PRINCE engine, and a key storage block that derive keys from an SRAM based Physically Unclonable Function (PUF). These IPs are accessible from the main processor as well as from a DMA engine for supporting functions like encryption and hashing. The hardware AES can be configured to operate with user defined or device specific key which is derived from the PUF. There are no security claims of the AES engine against physical attacks. The public information on specification of hardware AES engine is as follows.

- It supports key size: 128-bit, 192-bit or 256-bit key
- It supports following mode of operation: ECB, CBC, CTR, and ICB modes (ICB mode only supports to 128-bit key)
- AES functionality is combined with SHA block, referred to as SHA-AES
- When using 128-bit keys, the AES block takes 35 cycles for each block to encrypt, and additional 6 cycles for 192-bit key, and additional 12 cycles for 256-bit key.

We configure the main ARM Cortex® M33 MCU to run at the default frequency of 96 Mhz. The timing for AES-128 is determined by calling the encryption function between a LED toggle. The LED toggle then also serves as a trigger for the oscilloscope to synchronise the measurements. The side channel traces are measured on an oscilloscope via electromagnetic probe. We used a high-sensitivity low noise EM probe from Riscure [7] which has sufficient bandwidth to capture the activity at main clock frequency and clock frequency of hardware AES IP engine, since the probe is connected to a DC-powered Riscure amplifier with a frequency range of 100 kHz–2.5 GHz (Fig. 1).

[1] The chip manufacturers do not claim side-channel security for embedded AES engine. We have notified our findings to NXP PSIRT team and the details are under responsible disclosure.



(a) Okdo E1 Board [10] (b) Measurement setup

Fig. 1. Measurement setup for Okdo E1 development board.

While the trigger based on LED synchronises the traces, the trigger in itself is `18.68µs` but AES operation only takes 35 clock cycles which is about `0.35µs`. On further analysis, we found that apart from I/O manipulation, the processor also performs some key management task before the actual AES operation, causing a total execution time of `18.68µs`. The points corresponding to AES-128 operation only are thus determined by performing correlation power analysis on side-channel traces with public information like plaintext and ciphertext. The correlation peak corresponding to plaintext and ciphertext gives approximate bounds on the AES operation, allows to significantly reduce the number of samples per trace by approximately 9×. Note that, the internal architecture of the AES is not known and considered black box. Other techniques like normalised inter-class variance (NICV [2]) can also be used.

3.1 Leakage Model

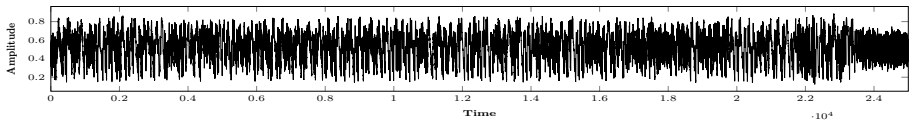
Since, the AES architecture is not known, it is hard to hypothesize the leakage model. Based on available information, we know that its a hardware architecture with 35 clock cyles. This means it is a parallel architecture which processes several bytes of the block per clock cycle. Previous works on hardware architecture target the last round with Hamming distance model *i.e.* leakage corresponding to state register being updated from last round input to output ciphertext [3]. However, given that 35 clock cycles also indicate that a complete round is not processed in every clock cycle. Thus, we assume a weak leakage corresponding to computation of last round Sbox. This is not optimal in hardware but still requires less assumption on the underlying architecture which is always computed. The model can be written as Model 1: $\text{S-box}^{-1}[ct_i \oplus k^*]$, $i = 1, ..., 16$, where S-box^{-1} indicate the inverse of AES S-box, ct_i, k^* mean the i-th byte of ciphertext and the correct key respectively.

4 Experimental Results

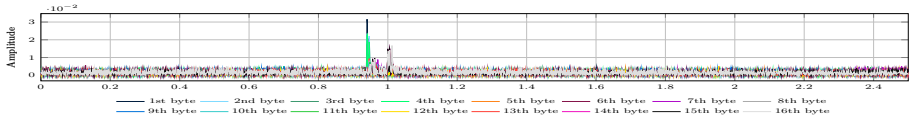
In this section, we report the results of deep learning based side channel analysis for the OKdo E1 board. We measured 500, 000 traces corresponding to fixed key and random plaintext. All the analysis in this section are based on these traces.

4.1 Locating AES Activity

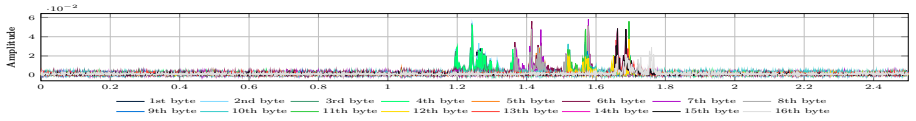
Since AES forms a small part of the triggered activity observed with toggling of the LED. We determine boundaries of the AES operation by computing correlation between traces and plaintext and ciphertext. As plaintext and ciphertext are first and last part of the computation respectively, activity corresponding to them will gives us bounds on AES activity in the trace. The result is shown in Fig. 2. The leakage of the plaintext and ciphertext is likely due to their transfer between main processor and hardware AES engine. For ciphertext, we observe leakage at 4 different instances, each time leaking 4 bytes. This indicate a 32-bit bus for data transfer between main processor and AES. The leakage of first, second, third, and fourth 32-bit words of ciphertexts was found between 12,000 and 14,000 points. The leakage of 16 plaintexts occurs at two instances respectively for first and last 8 bytes in Fig. 2b. This could be due to the loading of plaintext into hardware engine. The separate leakage of first 8 and last 8 bytes of the plaintext indicate a 64-bit architecture.



(a) A trace for hardware AES engine in LPC55S69-EVK



(b) Hamming weight leakage of 16 plaintexts



(c) Hamming weight leakage of 16 ciphertexts

Fig. 2. A trace of hardware AES engine and results for Hamming weight of plaintext and ciphertext.

4.2 Experimental Result for Deep Learning Based SCA

To perform a side-channel analysis based on deep learning, we use the state-of-the-art neural network structure of CHES 2020 [15]. More precisely, we consider the neural network structure as AES_HD structure since our main target is also hardware AES engine. As shown in Table 1, the CNN structure is quite simple.

Table 1. AES_HD architecture

Arch.	Convolution stage			MLP
	Filters	Kernel size	Pool size	
AES_HD	2	1	2	2

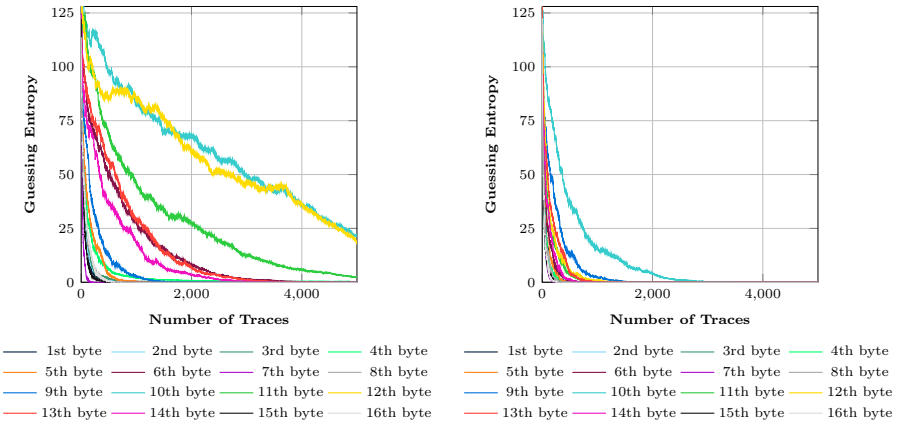
The attack is performed under a profiling setting. This means the adversary has access to a profiling or training dataset where the key and plaintext are known. The adversary then labels the dataset with the knowledge of key and plaintext and trains the deep learning architecture. Next, on the attack traces, where key is unknown, the unlabeled traces are queried against the trained model to predict the label. The predicted label for several traces are collected to determine the value of the secret key. The adversary then confirms the key with a known plaintext-ciphertext pair. In case, the attack is unable to find few key bytes, the attacker is able to brute force the remaining bytes using the known plaintext-ciphertext pair, up to a computation limit.

As stated earlier, the likely target leakage without much information on the hardware architecture can be tested with Model 1. We use Model 1 to label our training set, leading to 256 classes. Using 256 classes instead of commonly used Hamming weight of Model 1 (HW(Model 1)) will lead to an imbalanced dataset and must be avoided [11].

For Model 1, we take 45,000 traces (like [15]) as the number of profiling traces and 5,000 for the testing set. The testing set is unlabeled and queried against the trained model to predict labels which is then used for key recovery. The results are shown in Fig. 3a. It plots the guessing entropy of all the key bytes. A key byte is considered to be recovered, when the guessing entropy reaches minimum. In the current experiment, we can only recover 13 bytes of the 16 byte secret using all the 5,000 traces. As stated before, the remaining 3 bytes can be brute-forced using a known plaintext-ciphertext pair with a complexity of 2^{24}, which is easy to perform on a standard computer.

4.3 On the Power of Detailed Profiling

Now we consider a stronger attacker who has access to a bigger training dataset. For this, we used 200,000 traces for profiling. The rest of the experiments remain the same *i.e.* the unlabeled testing dataset is 5,000 traces and the labels are computed using Model 1. The results are shown in Fig. 4. As the deep learning



(a) 45,000 Training Traces (b) 200,000 Training Traces

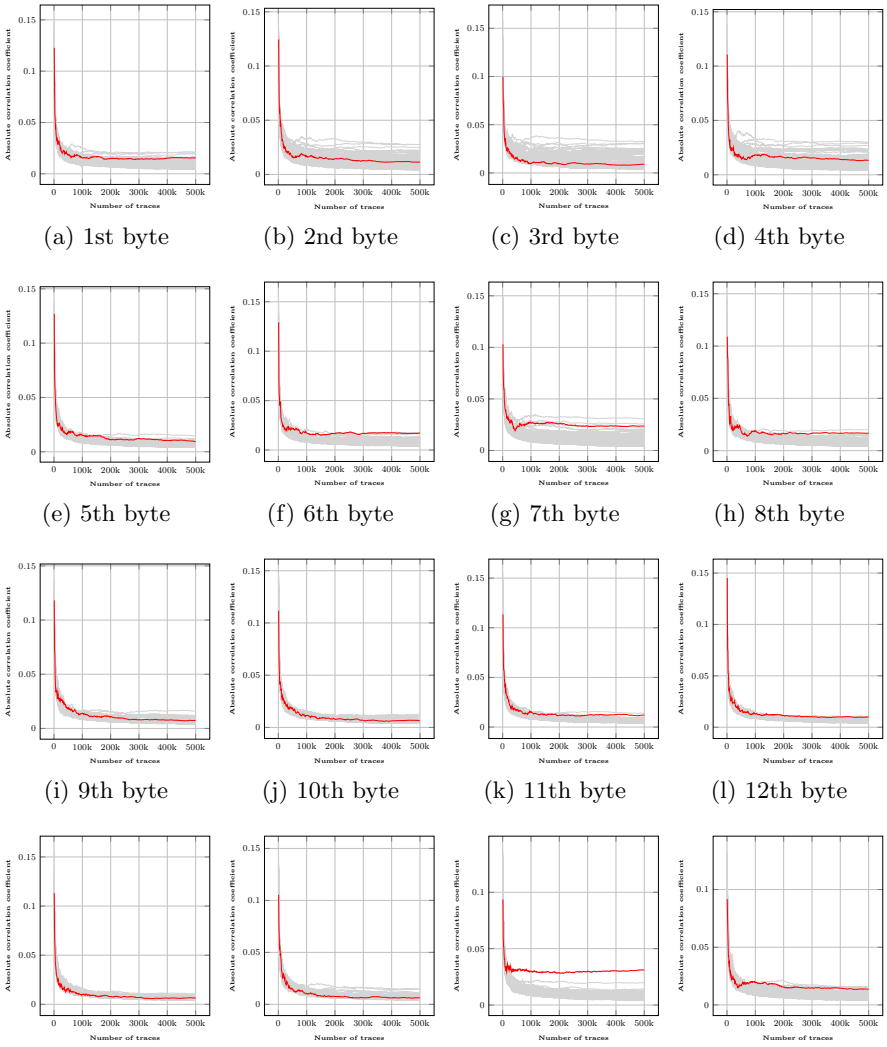
Fig. 3. Results of deep-learning based profiled SCA for 45,000 and 200,000 training dataset sizes.

model is now trained with a bigger training set, the attack in this case needs about 3,000 traces to recover the key. In fact, with less than 2,000 traces, 15 out of 16 bytes can be successfully recovered, leaving only one byte to guess. Care must be taken in choosing the training dataset size so as to not overfit the deep learning model.

4.4 Is Model 1 an Optimal Leakage Model

Finally, we verify if Model 1 actually fits well as the leakage model for the target black-box hardware engine. We performed correlation power analysis (CPA) in a known plaintext setting using all the 500,000 traces, with Model 1 as our leakage model. The results are shown in Fig. 4. The red line indicates the absolute correlation coefficient for correct key and the gray lines means the key candidates except for correct key. An attack is successful if red line stands out from all the grey lines. It can be observed that the attack is successful for only one byte (15th byte) and model Model 1 is not optimal for the given device. The brute force attack has to recover 15 bytes of the key which is beyond limit on standard computer, concluding the CPA to be unsuccessful.

Nevertheless, this highlights the power of deep learning based SCA. As shown previously, even without knowledge of the perfect model but only general information of the underlying architecture, deep learning based SCA could recover the key successfully.



(a) 1st byte (b) 2nd byte (c) 3rd byte (d) 4th byte

(e) 5th byte (f) 6th byte (g) 7th byte (h) 8th byte

(i) 9th byte (j) 10th byte (k) 11th byte (l) 12th byte

(m) 13th byte (n) 14th byte (o) 15th byte (p) 16th byte

Fig. 4. CPA trend for 16 bytes. (Color figure online)

5 Conclusion

In this paper, we analyze the side-channel security of black box hardware AES engine integrated in NXP LPC55S69JBD100 microcontroller. The microcontroller is developed for security applications like point of sale. By minimum assumption on the AES architecture and considering a commonly manipulated sensitive value as leakage, we demonstrate successful attack using deep learning based SCA. The attack requires only 3,000 traces from the victim device when

performed with a commonly known CNN architecture. We also confirmed that the leakage considered is not optimal for the architecture and could not recover the complete key using CPA. This demonstrates the advantages of using deep learning based SCA for targeting black box architecture. Further work can investigate precise leakage model and optimised CNN architecture for a worst case analysis on the underlying hardware AES.

Acknowledgement. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research.

References

1. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptographic Eng. **10**(2), 163–188 (2019). https://doi.org/10.1007/s13389-019-00220-8
2. Bhasin, S., Danger, J.L., Guilley, S., Najm, Z.: NICV: normalized inter-class variance for detection of side-channel leakage. In: 2014 International Symposium on Electromagnetic Compatibility, Tokyo, pp. 310–313. IEEE (2014)
3. Bhasin, S., Guilley, S., Heuser, A., Danger, J.-L.: From cryptography to hardware: analyzing and protecting embedded Xilinx BRAM for cryptographic applications. J. Cryptographic Eng. **3**(4), 213–225 (2013). https://doi.org/10.1007/s13389-013-0048-4
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
5. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
6. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
7. Doerr, C.: Side-channel based intrusion detection for industrial control systems. In: Critical Information Infrastructures Security: 12th International Conference, CRITIS 2017, Lucca, Italy, Revised Selected Papers, 8–13 October 2017, vol. 10707, p. 207. Springer (2018). https://doi.org/10.1007/978-3-319-99843-5_19
8. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
9. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 3–26. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_1
10. NXP Semiconductors: UM11126 LPC55S6x/LPC55S2x/LPC552x User manual Rev. 1.8 - 24 October 2019. https://www.mouser.com/pdfDocs/NXP_LPC55S6x_UM.pdf
11. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans. Cryptographic Hardw. Embedded Syst. **2019**(1), 1–29 (2019)

12. Picek, S., Samiotis, I.P., Kim, J., Heuser, A., Bhasin, S., Legay, A.: On the performance of convolutional neural networks for side-channel analysis. In: Chattopadhyay, A., Rebeiro, C., Yarom, Y. (eds.) SPACE 2018. LNCS, vol. 11348, pp. 157–176. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05072-6_10
13. Piret, G., Quisquater, J.-J.: A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45238-6_7
14. Wu, L., Picek, S.: Remove some noise: on pre-processing of side-channel measurements with autoencoders. IACR Trans. Cryptographic Hardw. Embedded Syst. **2020**(4), 389–415 (2020)
15. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptographic Hardw. Embedded Syst. **2020**(1), 1–36 (2020)