A systematic side-channel evaluation of black box AES in secure MCU : architecture recovery and retrieval of PUF based secret key

Won, Yoo-Seung; Bhasin, Shivam

2021

Won, Y. & Bhasin, S. (2021). A systematic side-channel evaluation of black box AES in secure MCU : architecture recovery and retrieval of PUF based secret key. 2021 IEEE International Symposium on Circuits and Systems (ISCAS).
<https://dx.doi.org/10.1109/ISCAS51556.2021.9401404>

<https://hdl.handle.net/10356/147151>

<https://doi.org/10.1109/ISCAS51556.2021.9401404>

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/ISCAS51556.2021.9401404>.

Downloaded on 26 Apr 2023 19:24:52 SGT

A Systematic Side-Channel Evaluation of Black box AES in secure MCU: Architecture Recovery and Retrieval of PUF based Secret Key

Yoo-Seung Won and Shivam Bhasin

Temasek Laboratories, Nanyang Technological University, Singapore

Email: {yooseung.won, sbhasin}@ntu.edu.sg

Abstract—Modern microcontrollers (MCUs) come packed with features to support rising demand of security and privacy in different applications. Features like hardware support for cryptography, trusted execution environment, memory protection, *etc* are widely available. In this paper, we take a deeper look into ARM Cortex M33 microcontroller designed to support critical applications like point of sale, smart home, smart factory, *etc*. In particular, we demonstrate architecture recovery of black box AES engine using side-channel analysis. The architecture is then exploited through side-channels to recover device intrinsic keys based on physical unclonable functions. Finally, feasibility of cross-device attacks are investigated with deep learning based side-channel attacks. Our results give a better insight into embedded AES engine available off-the shelf and allow user to design secure applications knowing such vulnerabilities at design time.

Index Terms—Hardware AES engine, Side-Channel Analysis, Deep Learning, Cross-device attack.

I. INTRODUCTION

Since the standardization by NIST [1], Advanced Encryption Standard (AES) has become a default for symmetric key encryption. It is used both in hardware and software platforms. Post standardization, a lot of research has focused on implementation security of AES. This include attacks like power or electromagnetic side-channel [2], fault attacks [3], cache-timing attacks [4], *etc*. Owing to look up table based structure, AES is specially vulnerable to cache-timing attacks which can be executed remotely, that has motivated manufacturers to integrate hardware support for AES. Performance and energy remains other motivating factors for having hardware support for AES. However, each manufacturer has a proprietary architecture of AES which usually is black-box and only available as an API. A better understanding of underlying AES architecture can help user optimize their applications.

In this paper, we take a look at off the shelf ARM Cortex-M33 microcontroller (MCU) with security cores like hardware cryptographic engines, physical unclonable functions (PUF), Trustzone *etc* integrated. The microcontroller is designed for security critical applications like point of sale and has features like hardware root of trust, trusted execution environment, memory protection, *etc*. As a lot of these features use underlying hardware AES as a building block, a better understanding of its functioning and security is important.

A. Related Works

O’Flynn and Dewar [5] reported side-channel attacks (SCA) on hardware AES engine embedded in SAML11 evaluation board. It contains the TrustZone for Cortex-M (Armv8-M) core with hardware AES accelerator. Our target the OKdo E1 development board has a similar Cortex-M core, the AES accelerator on the two chips are completely different. The AES architecture of SAM11 from Microchip studied in [5] takes 800 cycles for one encryption and the AES architecture studied in this work is on LPC55 from NXP (OKdo) takes only 35 clock cycles for encryption. While the focus of [5] was remote SCA, in this paper we rely on physical (EM) SCA to enable architecture estimation and demonstrating key recovery attacks.

B. Our Contributions

The main contributions of this work are as follows:

- We investigate security of hardware AES integrated in LPC55S69JBD100 microcontroller¹ against practical electromagnetic SCA.
- By analyzing different leakage models, we promote better understanding of underlying architecture of the AES and demonstrate successful key recovery attack when AES is configured with user defined secret key.
- Next, we demonstrate complete recovery of user-inaccessible physical unclonable function (PUF) based intrinsic device key using SCA.
- Finally, we show practical and efficient cross-device attacks on target AES, with deep learning based SCA.

II. TARGET SETUP & EVALUATION STRATEGY

In this section, we report the target board, experimental setup and evaluation process for side-channel analysis on black-box hardware AES engine available in the LPC55S69JBD100 on OKdo board.

A. Hardware Background

The OKdo E1 development board which is an ultra-low-cost Development Board based on the NXP LPC55S69JBD100

¹The chip manufacturers do not claim side-channel security for embedded AES engine. We have notified our findings to NXP PSIRT team as part of responsible disclosure.

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

dual-core Arm Cortex® M33 MCU is used as the application of point-of-sale terminal. The security features of LPC55S69JBD100 are explained in the user manual [6]. Analyzing the document, the hardware IPs consist of 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 PUF. An on-chip hardware AES engine is provided on the LPC55S69JBD100, which can either encrypt/decrypt using a user-defined key or a device intrinsic key derived from PUF. The manufacturer claims no side-channel resistance for this AES. In addition, ICB-AES mode is available which has the resistance against side-channel analysis.

The main focus of this work is study of the embedded AES engine to uncover the underlying architecture to explore further vulnerabilities. The specification of hardware AES engine is as below.

- Key size: 128-bit, 192-bit or 256-bit key
- 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 12 cycles for 256-bit key.
- The device intrinsic key derived from PUF only supports the AES-256 and is protected by countermeasures like KEYMASK against advanced side-channel like loading.

B. Experimental Details

The main ARM Cortex® M33 MCU supports clock up to 150 Mhz. We set default 96 MHz for main clock frequency. To measure the signal corresponding to AES operation in ECB mode, we call it in between a LED toggle. The LED signal via the GPIO pin can then be used as the trigger. To measure the electromagnetic emission, we used a high-sensitivity low noise EM probe from Riscure [7] which has sufficient bandwidth (100kHz – 2.5GHz) to capture the target activity. The measurement point is shown with red dotted circle in Fig. 1a. In order to capture the trace, we choose the sampling rate as 1GHz and we have used the MCUExpresso IDE v11.1.1 to flash the program to the target. Moreover, we only consider the single core scenario, even though there are multiple cores in the target.

While the trigger based on LED is $18.68 \mu s$ but AES-128 operation only takes 35 clock cycles which is about $0.35 \mu s$. We noticed that apart from I/O manipulation, the processor also performs some key management task [8] before the actual AES operation, causing a total execution time of $18.68 \mu s$. The points corresponding to AES-128 operation only are thus determined by performing correlation with side-channel traces and 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 $9\times$. Moreover, the internal architecture of the AES is not known and considered black box and estimating it is one of the objective of this work. The experiments are performed with the following approach:

- 1) Correlation analysis of SCA traces with plaintext/ciphertext is performed to identify points of interest and compress traces. Trace alignment is applied where necessary to increase signal to noise ratio.
- 2) Several hypothetical leakage model are tested to estimate the architecture of AES engine by attacking AES-128
- 3) Extend the knowledge of architecture, to demonstrate practical attack on AES-256 which is configured to use internal PUF based key, thus revealing the device key.

The attacks are performed in non-profiled setting, where adversary test dependency of actual SCA measurement with predicted leakage based on leakage model. The adversary test the modeled leakage for 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 [9] to perform a correlation power analysis (CPA). We hypothesize four leakage model for hardware implementation.

- **Model 1:** $HW(S\text{-box}^{-1}[ct_i \oplus k^*])$, $i = 1, \dots, 16$
- **Model 2:** $HW(S\text{-box}[pt_i \oplus k^*])$, $i = 1, \dots, 16$
- **Model 3:** $HD(S\text{-box}^{-1}[ct_i \oplus k^*], \text{ShiftRows}^{-1}[ct_i])$, $i = 1, \dots, 16$
- **Model 4:** $HD(S\text{-box}^{-1}[ct_i \oplus k^*], S\text{-box}^{-1}[ct_{i-8} \oplus k])$, $i = 9, \dots, 16$

where S-box and $S\text{-box}^{-1}$ indicate respectively AES S-box and the inverse of AES S-box, pt_i , ct_i , k^* , and k mean the i -th byte of plaintext, ciphertext, the key candidate, and the correct key respectively. Also, ShiftRows^{-1} represents the inverse of AES ShiftRows function. Additionally, $HW(x)$ indicates Hamming weight of x and $HD(x, y)$ implies Hamming distance value between x and y .

Normally, in the hardware implementation, Hamming distance model is more likely assuming a state register update for a round based implementation [10]. This corresponds to Model 3, where the final ciphertext is updated over final round input, assuming no logical countermeasures in place. The leakage of Model 1 and Model 2 is commonly suitable to software implementation but can still be exploited in hardware [10], specially if the register is cleared before writing new values. Unlike Model 1, Model 2, and Model 3, we employ a very specific leakage model in Model 4. Model 4 considers a 64-bit architecture where the state register is updated from first

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

8 bytes of round value to last 8 bytes of the round value, for the same round. The model is derived from our initial analysis and described in detail in the next section.

III. ESTIMATING ARCHITECTURE OF AES HARDWARE ENGINE

We measured 500,000 and 3,000,000 traces corresponding to fixed key and random plaintext for AES-128 and AES-256, respectively.

A. Identifying AES Operation in SCA Measurement

As mentioned earlier, we should determine the encryption operation within overall trigger signal. The result is shown in Figure 2a. We cannot distinguish some functions such as S-boxes and Mixcolumns as well as the whole AES operation, even though there is AES operation in Figure 2a. Therefore, we compute a correlation with plaintext and ciphertext in order to distinguish the bounds for AES operation.

As shown in Figure 2, the HW leakage of plaintext and ciphertext can be observed. This activity is related to transfer of plaintext for encryption and transfer of ciphertext after encryption between AES engine and main processor. It is well known that data bus of main processor leaks in HW model as its precharged in every cycle. We can estimate that the HW leakage of ciphertext is occurred by software implementation which appears at 4 different instances, each time leaking 4 bytes indicating 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 sample points. In contrast to ciphertext behavior, the leakage of 16 plaintexts is occurred at two time positions respectively for first and last 8 bytes in Figure 2c. Naturally, we can estimate that this leakage might be due by hardware properties when the initialized register is updated to plaintext information. The separate leakage of first 8 and last 8 bytes indicate a 64-bit architecture. The leakage of last 8 bytes seems lower than first 8 bytes as like the AES is activated leading to noise in measurements. We conduct further analysis in the next subsection to confirm this hypothesis.

Fig. 3: Estimation of hardware architecture of AES engine.

B. Architecture Estimation

For estimating the hardware architecture of AES engine, we conduct experiments with 4 leakage models as explained in Section II. Considering round-based AES hardware engine, we first perform CPA with Model 3. However, CPA with Model 3 resulted in a failure with all 500,000 traces. This is not surprising as Model 3 assumes round based architecture which should take about 10~11 cycles but the documentation reveals 35 clock cycles is required to complete one AES operation.

Next, we check for leakage corresponding to first and last round. We tried both Model 1 and Model 2 on the same set of 500,000 traces. With Model 2, we did observe some leakage peaks but the key recovery was not successful with the acquired set of traces.

On the other hand, Model 1 resulted in a clear leakage allowing easy recovery of first 8 bytes of the last round key. Despite of the successful recovery of first 8 bytes, the last 8 bytes in last round key cannot be retrieved. This is surprising as in a hardware implementation all bytes are expected to leak more or less equally, except some statistical bias. Considering, a 33 clock cycle architecture, we hypothesize a 64-bit architecture, which processes first 8 bytes and last bytes of a round in consecutive rounds. The consecutive cycles then enable a Hamming distance leakage between first and last 8 bytes. This results in construction of Model 4. We next perform a CPA using Model 4 with a focus on last 8 bytes of the last round, allowing successful key recovery with Model 4. We observed the peaks of first 8 and last 8 bytes were shifted in time by one clock cycle, confirming a sequential processing of rounds in Figure 2c.

Based on this analysis, we estimate the overall architecture of the AES-128 engine as depicted in Figure 3. Each round of AES takes 3 clock cycles and has a 64-bit datapath. In the first clock cycle, the state register is cleared followed by calculation of first 8 bytes of the round. The leakage in this case follow Model 1. The calculation of first 8 bytes of round are followed by round computation for last 8 bytes resulting in leakage for Model 4. With 3 cycles per round, AES is computed in 33 clock cycles, extra 2 cycles for plaintext/ciphertext processing.

C. Full Key Recovery For User Defined Key AES-128

Based on the previous estimation of AES engine, we conduct a key recovery attack on AES-128 with user defined key. This helps us validate our previous analysis.

The final results of this evaluation with 500,000 traces are provided in Table I. It uses minimum traces to disclosure

TABLE I: MTD for CPA on AES-128.

Byte	1	2	3	4	5	6	7	8
Model 1	60k	136k	90k	34k	23k	19k	3k	10k
Model 4	-	-	-	-	-	-	-	-

Byte	9	10	11	12	13	14	15	16
Model 1	-	-	-	-	-	-	-	-
Model 4	42k	85k	6k	15k	15k	10k	7k	9k

(MTD) [11] as a security metric. As previously mentioned, the leakage of first 8 bytes follow **Model 1** and leakage of last 8 bytes follow **Model 4**. ‘-’ denotes the attack didn’t succeed with all 500k traces. In the worst case, 136,000 traces are required to retrieve the correct key. However, with only 42,000 traces 12 bytes of the correct key can be recovered, remaining 4 bytes can be brute forced with complexity of 2^{32} .

IV. KEY RECOVERY ATTACK ON AES-256 WITH DEVICE INTRINSIC KEY

The targeted LPC55S69JBD100 microcontroller also features a device intrinsic key using an internal PUF. The key can be used by hardware engines like AES but cannot be read by the user in any circumstances. In this section, we show that due to vulnerability of the hardware engine, the intrinsic key of the device can be leaked. According to [8], the PUF is loaded into KEYMASK register, which is protected by random precharging to avoid any advanced SCA on key loading. Nevertheless, the usage of key in AES-256 remains exploitable as shown in the following.

We exploit the previously recovered knowledge on the architecture of AES for this key recovery. Since the intrinsic key can be used with AES-256 only, in this section we target AES-256, which uses the same hardware engine. The attack on AES-256 targets last two rounds [12]. Based on the previous results, we tried both **Model 1** and **Model 4** on the set of 3,000,000 traces for last two rounds. The attack on last round is exactly the same as for AES-128 in the last section and we can recover all the bytes. Once all 16 bytes of last round key are recovered, one can reverse the last round and repeat the same attack on penultimate round. The same traces can be reused. When the attack is repeated on penultimate round, we recovered 13 of 16 bytes with 620k traces, while bytes 5, 9 and 13 did not provide a conclusive result with all the traces. For brevity, Figure 4² show CPA trace for two bytes in penultimate round: one where the attack is successful and other where attack fails. Finally, we brute force the remaining bytes of the key with complexity 2^{24} and recover the master key. The same configuration and plaintext on a copy of this board resulted in different ciphertext confirming that the recovered key is indeed intrinsic to the device.

V. VULNERABILITY TO DEEP LEARNING BASED CROSS-DEVICE ATTACKS

Cross device attacks are a stronger class of SCA, where adversary with complete control over a clone device, learns the leakage profile of the target sensitive activity. Next, with

²In overall key candidates, ghost key indicates the key candidate has maximum correlation except for correct key.

Fig. 4: CPA results for two bytes in penultimate round.

Fig. 5: Cross device attacks on for two bytes in last round.

only a few traces from victim device and leakage profile from the clone device, performs a key recovery attack on victim device. The use of deep neural networks (DNN) have made cross-device attacks effective [13], [14]. DNN are trained with SCA measurements from clone device with known leakage labels, while trained model is queried with unlabeled SCA traces from victim device. Correctly predicted labels lead to key recovery.

For the DNN structure we choose AES_HD structure of convolution NN and the related hyperparameters in [15]. The attack requires could recover the full key with less than 10,000 traces, when trained and tested on same device.

In cross-device setting, the clone device and victim device might differ due to process variations, making the learned model overfit for clone device. For example, [13] reported up to $20\times$ degradation in attack performance in cross-device setting. We measured traces from two identical OKdo E1 development board: 100,000 traces are measured from Device 1 for training the DNN while 20,000 traces from Device 2 (victim). Figure 5 shows the result (only two key bytes are shown for brevity) for attack in self and cross device setting. Indeed, there is degradation of 30-60% degradation in number of traces required, however this is much better than 2000% degradation reported for other devices [13]. 60% degradation correspond to few extra minutes of measurement time, making cross-device a serious threat to such hardware engines.

VI. CONCLUSION

In this paper, we analyze the side-channel security of hardware AES engine integrated in NXP LPC55S69JBD100 microcontroller. We use CPA to perform architecture estimation and key recovery. The key recovery is demonstrated for both user defined key with AES-128 and device intrinsic key derived from PUF with AES-256. Further, successful cross-device attacks are demonstrated.

ACKNOWLEDGMENT

The authors acknowledge the support from the ‘‘National Integrated Centre of Evaluation’’ (NICE); a facility of Cyber Security Agency, Singapore (CSA). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

- [1] P. FIPS, “197, advanced encryption standard (aes), national institute of standards and technology, us department of commerce, november 2001,” *Link in: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>*, 2009.
- [2] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.
- [3] G. Piret and J.-J. Quisquater, “A differential fault attack technique against spn structures, with application to the aes and khazad,” in *International workshop on cryptographic hardware and embedded systems*. Springer, 2003, pp. 77–88.
- [4] D. J. Bernstein, “Cache-timing attacks on aes,” 2005.
- [5] C. O’Flynn and A. Dewar, “On-device power analysis across hardware security domains,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 126–153, 2019.
- [6] *UM111126 LPC55S6x/LPC55S2x/LPC552x User manual Rev. 1.8 – 24 October 2019*, NXP Semiconductors, https://www.mouser.com/pdfDocs/NXP_LPC55S6x_UM.pdf.
- [7] C. Doerr, “Side-channel based intrusion detection for industrial control systems,” in *Critical Information Infrastructures Security: 12th International Conference, CRITIS 2017, Lucca, Italy, October 8-13, 2017, Revised Selected Papers*, vol. 10707. Springer, 2018, p. 207.
- [8] “Lpc55sxx usage of the puf and hash crypt to aes coding.” 2019, <https://www.nxp.com/docs/en/application-note/AN12324.pdf>.
- [9] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.
- [10] S. Bhasin, S. Guilley, A. Heuser, and J.-L. Danger, “From cryptography to hardware: analyzing and protecting embedded xilinx bram for cryptographic applications,” *Journal of Cryptographic Engineering*, vol. 3, no. 4, pp. 213–225, 2013.
- [11] J. Danial, D. Das, S. Ghosh, A. Raychowdhury, and S. Sen, “Sniffer: Low-cost, automated, efficient electromagnetic side-channel sniffing,” *arXiv preprint arXiv:1908.09407*, 2019.
- [12] M. Neve and K. Tiri, “On the complexity of side-channel attacks on aes-256-methodology and quantitative results on cache attacks,” 2007.
- [13] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa, “Mind the portability: A warriors guide through realistic profiled side-channel analysis,” in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.
- [14] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, “X-deepsca: Cross-device deep learning side channel attack,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.
- [15] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, “Methodology for efficient cnn architectures in profiling attacks,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–36, 2020.