# Side-Channel Attacks: ChipWhisperer

Final Technical Document

^Sang Yoon Kim, *Ryanh Tran, *Kevin Hutto and *^~Vincent Mooney

*Secure Hardware VIP Research Group

**School of Electrical and Computer Engineering, College of Engineering*

*^School of Computer Science, College of Computing*

*~School of Cybersecurity and Privacy, College of Computing*

Georgia Institute of Technology

April 30th, 2024

## Table of Contents

## I. Terminology

**Advanced Encryption Standard (AES)**: Cryptography algorithm that uses a public key to protect data using multiple rounds of substituting and shuffling, established by NIST

**Analog-to-Digital Converter (ADC)**: a system that converts an analog signal into a digital signal

**Cryptography (Cryptographic algorithms)**: Mathematical functions that takes two input parameters (message [plaintext] and a cryptographic key) and maps the input to an output (ciphertext)

**Symmetric cryptography**: Function that share a common key

**Asymmetric cryptography**: Function that has utilizes a key pair (public key/parameter and a secret key/parameter)

**Differential Power Analysis**: Specific side-channel attack that utilizes power traces and signal processes and error correction to reveal secret keys

**Side-Channel Attack (SCA)**: an attack enabled by leakage of information from a physical cryptosystem exploiting characteristics such as timing, power consumption, and electromagnetic and acoustic emissions. [1]

**Power Trace**: As a target devices encrypts / decrypts data, the power consumption varies depending on the operations and data processed

**Simple Power Analysis (SPA)**: SCA accomplished by directly interpreting the measuring the current and power consumption of a device [5]

**Differential Power Analysis (DPA)**: SCA accomplished by statistically analyzing power consumption measurements from a device, depending on the data being processed [5]

**Field Programmable Gate Array (FPGA)**: A chip whose logic cells and internal connections can be modified to different behaviors [5]

**Trojan hardware**: a malicious modification of the circuitry of an IC chip. It is done during the design or fabrication of chip [6]

**Integrating ADC**: An ADC which goes through the run-up and run-down phase to compute the digital result. The value is computed as function of the reference voltage, the constant run-up time period, and the measured run-down time period [7]

**Test Vector Leakage Assessment (TVLA)**: An assessment that evaluates the side-channel leakage of sensitive information from the hardware implementation of a design [25]

## II. Motivation

Based on ideas from Kevin Hutto, the Analog design team is working on a potential solution to prevent side-channel-based attacks on an Integrating ADC by adopting a dummy counter. Prior research has been done to create a side-channel-preventive model of different ADCS such as the Successive Approximation Register ADC or flash ADC [28], but little to no work has been done on an Integrating ADC. The adoption of a dummy counter is based on hypothesis and has not been proven or tested. Our motivation lies in the viability of a side channel attack of this proposed solution, and ways to verify its side-channel-preventative capabilities.

## III. Goals

For this stage in our research, we seek to create a firm understanding of power SCAs through the ChipWhisperer tool suite. Using the ChipWhisperer Nano, our goal is to carry out a physical SCA experiment on the target device furthermore, we seek to understand and formulate a differential power analysis based SCA experiment that could be done on power traces from the parallel analog design team. Our goal is to use the understanding and techniques learned from our previous steps to formulate a SCA on a target device that has a close relationship to the Integrating ADC with a dummy counter. Our current research leads to the ChipWhisperer Husky, where we can utilize the iCE40 FPGA target device to implement the ADC dummy counter design.

## IV. Background

### IV.1 Integrating ADC

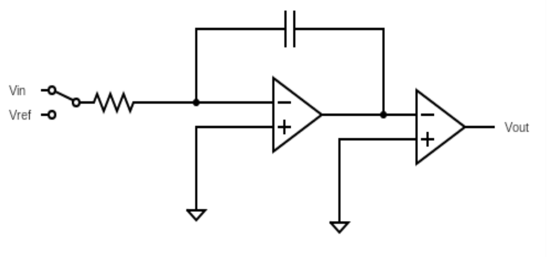


***Figure 1.*** A simple circuitry of an Integrating ADC

The Integrating ADC comprises several key components, including an integrator, a switch for selecting between measured and reference voltage, a timer, a comparator, and a controller. Within the circuit, two primary conversions occur [8]. Firstly, during the run-up phase, the measured voltage is designated as the input to the integrator. During this stage, the integrator ramps up over a fixed period, charging its capacitor. In the subsequent run-down phase, the reference voltage is applied as the input to the integrator. The duration taken for the integrator's output to return to zero is measured using a timer. The time recorded during the run-down phase is direct proportional to the final digital value attained by the ADC.

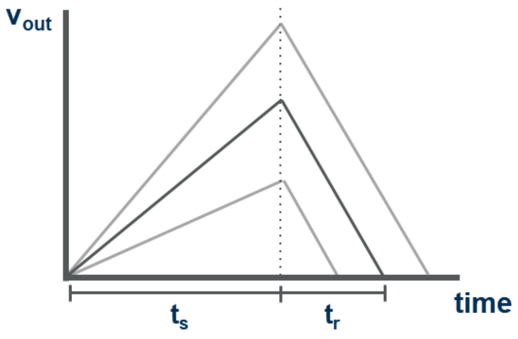


*Figure 2.* The power output during the two phases of an Integrating ADC

## IV.2 Hypothetical Attack on the Integrating ADC

The plain model of the Integrating ADC is susceptible to a basic power SCA, notably the SPA attack. The ADC by design, forces the counter to operate in accordance with the magnitude of the analog value. The duration for which the counter is active, inadvertently leaked through power consumption, directly correlates with the final digital value obtained. Building upon this theoretical background, a hypothetical SPA attack on the plain Integrating ADC model can be constructed.

The following attack aims to obtain the digital output from the ADC by analyzing the power leakages of the device while it is measuring the input. We assume that the Integrating ADC employs a basic implementation without any protective measures such as masking or hiding. Furthermore, we presume that the adversary possesses knowledge of the internal circuitry of the ADC and has possession of an ADC with the same model to execute inputs and capture its power traces.

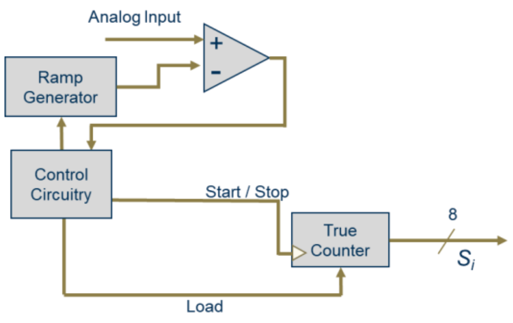


*Figure 3.* A simplified circuitry of the targeted Integrating ADC

Given that the plain Integrating ADC design relies on a single counter to measure the analog input, the power trace indicates the counter's activity proportional to the digital value. Following the charging of the ramp generator, the counter initiates counting during the discharge phase and will halt when the discharged value from the ramp generator matches with the analog input, like shown in Figure 4 as $t_r$. Subsequently during $t_i$, it will remain in an idle state until the value from the counter is processed, with the counter being inactive. Lastly during $t_r$', the process is repeated as the ramp generator recharges itself with the next analog input. Therefore, a straightforward simple analysis of the power trace from the ADC will be able to deduct $t_r$ which allows to predict the analog values being measured, and thus this hypothetical attack shows the inherent weakness in the plain Integrating ADC model.

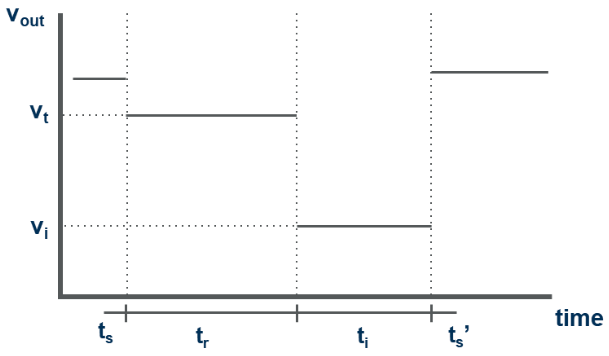


***Figure 4.*** A hypothetical power trace from the targeted Integrating ADC

## IV.3 Proposed Circuitry Design

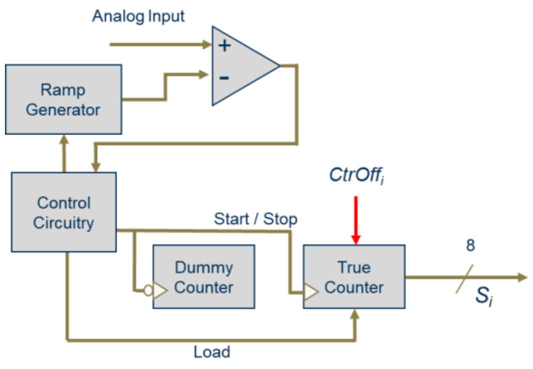


***Figure 5.*** The proposed circuitry design of the Integrating ADC

Therefore, we propose a circuitry design for the Integrating ADC as a potential solution for preventing such power analysis attacks. The high-level idea of it follows by using a dummy counter in addition to the true counter to constantly be counting for a set length of round. This prevents an adversary from simply spotting through the power leakage, the time the counter had run to calculate the digital code and the corresponding analog input.

## IV.4 Types of Side-channel Attacks

There are three main types of attacks on cryptographic devices: Passive, Active, and Invasive. A passive attack exploits the physical properties of the device, (e.g. execution time or power consumption) to reveal the secret key [5]. An active device manipulates the device's input or environment to act abnormally, and it is through this abnormal behavior that the secret key is revealed. Invasive attacks depackaging the device and then accessing the components directly using expensive equipment, such as a probing station. Invasive attacks can be both passive or active. Another categorization of attacks involves the invasiveness of such attacks. A semi-invasive attack depackages the device, but no contact is made with the chip surface, and so the passivation layer stays intact. Passive semi-invasive attack can read out the contents of a memory cell without the use of probing and side-channel methods. Active semi-invasive attacks induce faults using X-rays, electromagnetic fields, or light. Although these attacks do not require expensive equipment, there is challenge in finding the precise position to mount an attack on the surface. Non-invasive attacks only exploit the directly accessible interfaces. Because only the interfaces are used, the device is not permanently altered and therefore no evidence is left behind. This attack can be done using relatively inexpensive equipment, which poses a serious practical threat to the security of cryptographic devices. A side-channel attack is a passive, non-invasive attack that includes timing attacks, power analysis, and electromagnetic attacks [11][12].

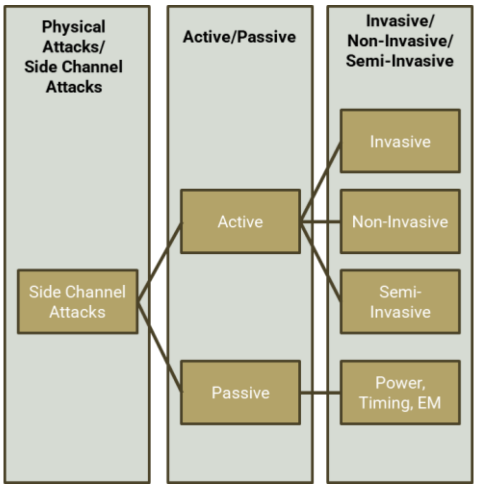


***Figure 6.*** A visualization of the types of SCAs

## IV.5 ChipWhisperer Hardware

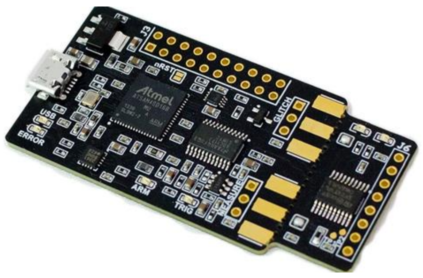


***Figure 7.*** The ChipWhisperer Nano [2]

The CW1101 ChipWhisperer Nano is a tool designed for side-channel power analysis training, particularly for conducting attacks against algorithms such as AES using power analysis techniques. It can perform attacks on its detachable target device as well as on external targets when used in conjunction with the ChipWhisperer housing boards. While the omission of an FPGA helps to keep costs down, it also imposes limitations on functionality. The capture device is equipped with an 8-bit 20 MS/S analog-to-digital converter, facilitating the capture of power traces from both internal and external targets. And the removable on-chip target device, powered by the STM32F0 and utilizing the ARM Cortex-M0 chip, allows for the programming of various algorithms for simulation purposes. To control both the main board and the target device, the ChipWhisperer Nano works on a Jupyter-based environment.

It has a price tag of $50.00 [13].

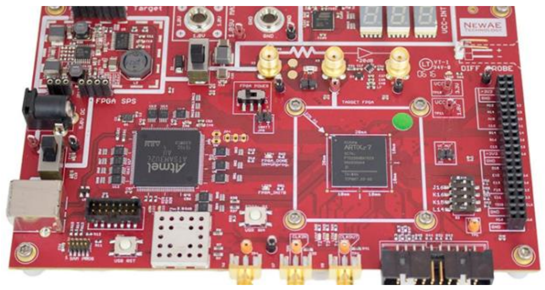


***Figure 8.*** The CW305 Artix FPGA [4]

The CW305 Artix FPGA is a standalone FPGA target board developed by NewAE, designed to integrate with ChipWhisperer's proprietary 20-pin interface. It contains a programmable VCC supply and a phase-locked loop for clocking the FPGA, enhancing flexibility and precision in operation. However, the board requires an external oscilloscope or capture box for performing power measurements. The custom USB interface allows communication with the FPGA, offering an address/data bus for data transfer and configuration purposes. The target device comprises a Xilinx 7 Series FPGA, which can be programmed over USB and JTAG, and can be utilized as a standalone device. It has a price point of $1,000.00 [14].

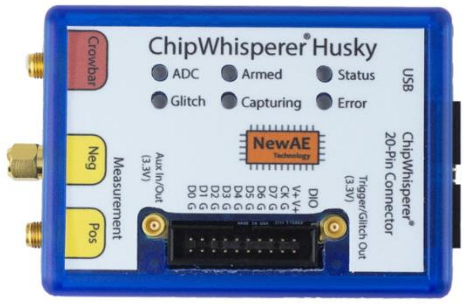


***Figure 9.*** The ChipWhisperer Husky [23]

The ChipWhisperer Husky is an advanced side-channel power analysis tool, serving as the successor to the ChipWhisperer Lite and offering capabilities that surpass those of the ChipWhisperer Nano. Utilizing the ChipWhisperer 20-pin Connector, it can establish connections with a wide range of targets, provided they can be accommodated onto a compatible ChipWhisperer target board. Notably, the Husky is equipped with an internal FPGA accessible via a separate 20-pin input, allowing users to utilize trigger inputs for enhanced functionality. For the analysis of ARM processor traces, a dedicated Arm trace sniffer, TraceWhisperer, developed by NewAE, can be used specifically for trace analysis. The capture device uses a 12-bit 200 MS/S analog-to-digital converter, enabling the capture of power traces from both internal and external targets. Like all other ChipWhisperer capture devices, the Husky operates within a Jupyter-based python environment. It has a price point of $549.00 [23].

***Figure 10.*** The Lattice iCE40 FPGA [23]

The Lattice iCE40 FPGA is a small, low-power FPGA designed with a compact form factor measuring 1.4 mm x 1.4 mm x 0.45 mm. The FPGA contains features including up to 3,520 4-input Lookup Tables, facilitating flexible logic implementation, and up to 26 I/Os for customized interfaces. Additionally, it offers up to 80 Kbits of embedded distributed memory. Capable of executing FPGA designs written in Verilog, as well as running demos, reference designs, and kick-start designs, the iCE40 FPGA serves as a versatile platform for FPGA development. It is included in the ChipWhisperer Husky Starter Kit, which serves as a target device that can be connected to the provided CW313 target baseboard, offering an opportunity for side-channel power analysis on FPGA-based projects [24].

## IV.6 ChipWhisperer Courses

ChipWhisperer is an open-source toolchain for teaching power analysis SCA and glitching attacks, developed by Collin Flynn and acquired by NewAE technologies. The software for the ChipWhisperer device (based on Python API for capturing and analyzing) as well as basic tutorial courses on the ChipWhisperer suite of devices are provided on the ChipWhisperer-Jupyter project in GitHub under NewAE. In total, there are 6 total courses for SCA and 2 course for fault attacks [22]. Specifically, SCA 101 and SCA 201 focus on power analysis attacks on AES, SCA 202 on power analysis on asymmetric implementations, SCA 203 (Incomplete) on vector leakage assessment, and SCA 204 and SCA 205 on power analysis on hardware/software Elliptic Curve Cryptography. Each course consists of 1~4 labs, with each lab having a accompanying Jupyter notebook with the explanations of the steps of procedure, example code, solution code, and output. Additionally, there are hinted to be additional training courses locked behind a paywall available on the NewAE website [29].

## IV.7 AES

Advanced Encryption Standard (AES) is a specification established by NIST in encryption of electronic data. Data is encrypted into a fixed 128-bit block under a key with the same length, with keys being 128, 192, or 256-bit length each being AES-128, AES-192, and AES-256 respectively. The data and key are represented as an array of bytes with 4 rows and 4 columns (the state). The encryption process is key-iterated, where round transformations are repeatedly applied to state, with 10 repetitions when using AES-128. The encryption starts with a round key being generated by a key scheduling algorithm for each repetition. Decryption works similarly to encryption with the round keys being applied in reverse order using the inverse of the round transformation. Each of these round transformations consists of 4 steps, AddRoundKey where individual bytes of the state are XORed with the round key, SubBytes, where byte substitution is done on the individual bytes of the state using a lookup table, ShiftRows where the rows of the blocks are shifted by different offsets, and finally mix columns where a column of the state is shuffled using matrix multiplication. The last (10$^{th}$) round of AES-128 skips the mix column operation for efficiency.

## IV.8 TVLA

The Test Vector Leakage Assessment (TVLA) is an examination of a device and its susceptibility to data leakage of unmasked cryptographic quantities through side channel attacks [25]. A device fails the TVLA if any secret data can be deduced through electromagnetic analysis or power signal tapping. TVLAs are split into two main classes: general and specific assessments. A general assessment tests if information directly dependent on input data or the key can be deduced through SCAs, while a specific assessment tests if any intermediary data can be exploited to reveal hidden data. A general test failure allows the possibility of a SCA vulnerability, while a specific test guarantees that leakage can be exploited to recover secret data. In both tests, all rounds of encryption should be run and exploited. The test is executed by running a comparison of running a fixed text encryption vs a random text. If the power trace is similar by Welch's T-Test, then there is a high likelihood that the device is susceptible to attacks.

## IV.9 Prior Research using ChipWhisperer

We were provided with a research paper that similarly used ChipWhisperer to research Side-Channel weakness. The paper titled "*A Deep Learning Technique for Efficient Side-channel Attacks*" focused on a side-channel weakness of the Randomization solution equipped on the Elliptic Curve Digital Signature Algorithm through Long short-term memory (LSTM) network architecture. The paper goes through the description of statement and proof, of a simulation on an Arm Cortex-M architecture through micro-ecc framework, and a real physical experiment through the ChipWhisperer platform (STM32F415). The researchers utilized the ChipWhisperer CW308 UFO board alongside the STM 32F415 target chip manufactured by NewAE. The UFO target board was only used for capturing the power traces of the target devices running the fixed routines, which were then passed through deep learning procedures. The following method would have to be adjusted to be used in our case as current goals focus on circuitry level implementation which would better be suited to a FPGA target board.

The STM target chip used in the research contains an ARM Cortex-M4 processor, while the ChipWhisperer Nano in our experiments contains the STM32F0 as an on-chip target device, using the ARM Cortex-M0 processor. Both processors are single core, do not contain internal cache memory, a 4-stage pipeline, and are 32 bits. The M0 was made to run basic tasks, while the M$ was made to run heavier workloads and therefore contains more power. Furthermore, the M4 contains additional instructions in the ISA to suit its purpose of executing advanced workloads.

## V. Experiment

### V.1 Background

We utilized labs provided from ChipWhisperer (SCA101: Part 3, Topic 3 – DPA on Firmware Implementation of AES [4]) to familiarize ourselves with ChipWhisperer hardware and software. The lab emulates a real world physical DPA and uses the ChipWhisperer Nano both as the target device and the capture device. The on-chip STM32F0 parget device is used to run the AES encryption process, while the power analysis capabilities of the ChipWhisperer Nano to capture the power traces from the target device to run a SCA and guess the key used for AES. The lab focuses on the AddRoundKey and SubBytes step of AES, as the output from the steps is directly related with the round key that was accounted for.

### V.2 Setup

To set up the experiment, the AES firmware needs to be uploaded onto the target device on the ChipWhisperer Nano. In order to do so, Python, ChipWhisperer bash, and Jupyter Lab needs to be installed on to the PC before establishing a connection between the PC and the ChipWhisperer Nano. Once these prerequisites are in place, the appropriate firmware located in the on the PC – for example, ending in ".../simpleserial-aes" - is uploaded to the connected ChipWhisperer Nano using ChipWhisperer-bash. It was found during the experiment that additional steps were necessary than just the instructions provided by ChipWhisperer. The group was able to find that an additional line "PLATFORM = CWNANO" was needed to ensure the correct upload of the file. Upon successful firmware upload, one should observe the light on the top-right part of the ChipWhisperer Nano blinking from red to green, indicating that the scope has been correctly set for the ChipWhisperer Nano to start capturing the power trace.

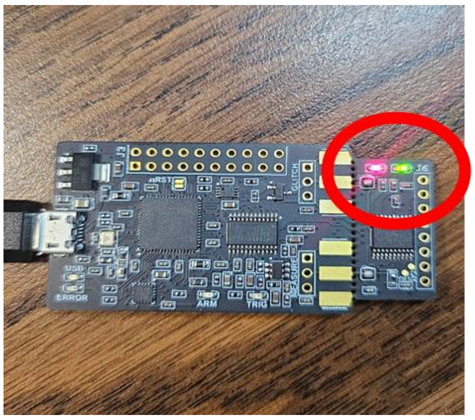


***Figure 11.*** The top-right part of the ChipWhisperer Nano blinking green as the scope has been set correctly

## V.3 Capturing Power Traces

In the target firmware provided for the experiment, ChipWhisperer uses its internal library to initiate and terminate the capture of individual executions of AES. Two key functions, trigger_high() and trigger_low(), is employed in this process. Under normal circumstances, trigger_high() triggers the commencement of the power trace capture, with the duration defined by the number of samples specified by the user beforehand. Conversely, the purpose of trigger_low() remains uncertain at present, as it does not terminate a capture, unlike trigger_high(). Instead, the capture process only concludes once the designated number of samples has been acquired.

To ensure consistent capture of power traces for each execution of AES, trigger_high() is  positioned before each AES execution, with the number of samples set not to exceed the duration of a single execution. Determining the optimal sample value required a trial-and-error approach, given the challenge of accurately calculating the exact number of samples required to cover the duration of one execution of AES. Considering that AES-128 involves 10 rounds of similar procedures, with the final round shorter than others, analysis of the power trace graph provided insights into whether all rounds of the AES encryption process were effectively captured.

To capture the power traces of the AES encryption process, we used a series of methods provided by the ChipWhisperer API. Initially, the key and plaintext are set using the ktp.Basic() function, followed by the encryption of the text using simpleserial_write('p', text) through the simpleserial-aes firmware implemented on the device. Subsequently, the response from the target device is read using simpleserial_write('p', 16), and power traces are captured using the capture() function. This process is repeated 2500 times to ensure accuracy for Differential Power Analysis (DPA) attack purposes. Upon successful capture, the Jupyter Lab interface should display that all 2500 power traces have been accurately recorded.

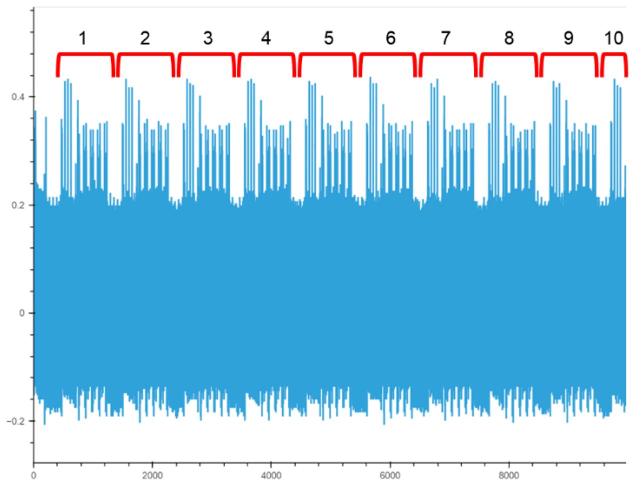


***Figure 12.*** A power trace of AES encryption (10,000 samples, normalized to -0.5~0.5) containing 10 repetitive traces

By executing the necessary instructions, the power trace for encrypting each of the 2500 different plaintexts through AES encryption can be obtained as can be seen in Figure 12. The get_last_trace() function retrieves the recorded power trace, containing 5000 samples - the default sample size can be modified - with values scaled and shifted to range between -

0.5 and 0.5 - optionally, raw 8-bit values can be obtained as well. When capturing the power trace with a sample size larger than the AES execution, it was noted that the sample values obtained were mostly random and appeared to lack any relationship to the executed process. Despite the encryption process concluding, the ChipWhisperer continued to capture traces, filling up all samples, potentially while in an idle state.

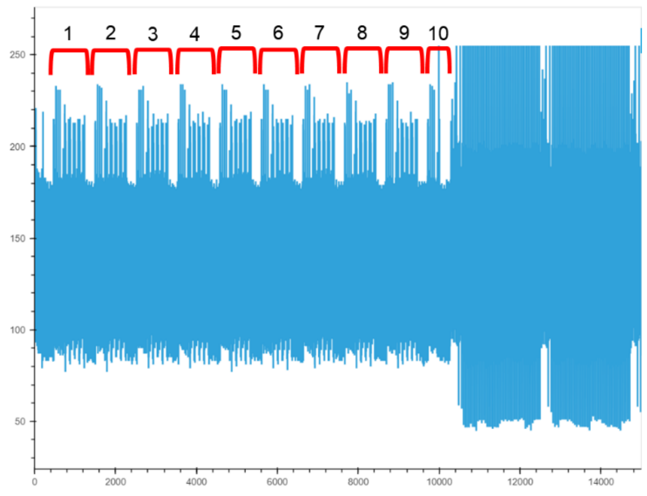


***Figure 13.*** A power trace captured with much larger sample size (15,000 samples, raw 8-bit value)

## V.4 Analyzing Power Traces

To analyze the power traces, we go thorugh the process of testing all potential values for the key byte with the actual plaintext while masking a single bit. This enables the segregation of power traces into two distinct sets (0 or 1) based on the bit value being a 0 or 1. Knowing the internal steps, we simulate the calculation of the initial AES procedure, involving an XOR operation between the key and plaintext, followed by lookup table substitution. Once the power traces are categorized, the average value of each group is calculated, and the differences between these averages are computed. The correct key value will show a clear correlation with the AES steps, specifically at the AddRoundKey and SubBytes steps of the first round, through power consumption which is typically leaked through vectors like hamming weight. As the AES process progresses however, the correlation between the bit and power consumption decreases. Therefore, selecting the maximum across all differences can find the difference value from the step demonstrating the largest correlation. Conversely, for any other non-key value, the results tend to be random, with values close to 0, even when selecting the maximum among them. By arranging the difference values from these guesses, the guess with the largest maximum difference is likely the correct byte value of the key. This process is iteratively repeated for the remaining 15 bytes, ultimately resulting in the derivation of the complete 128-bit key.

## V.5 Results

As a result, we were able to obtain the correct key bytes from the power traces. Figure 14 displays the difference values between power consumption and the difference of the mean power trace for each potential key, with distinct colors representing different possible key values: blue for 0x2A, red for 0x2B (the correct key), green for 0x2C, and yellow for 0x2D. Notably, for any other difference values, the power traces are randomly divided, resulting in the plot hovering close to 0. The correct key value (red) exhibits a significant spike in the initial part of the plot, corresponding to the 1st round AddRoundKey and SubBytes step. This observation indicates a strong dependency of the power consumption on the value 0x2B, strongly suggesting it as the key for the particular byte.

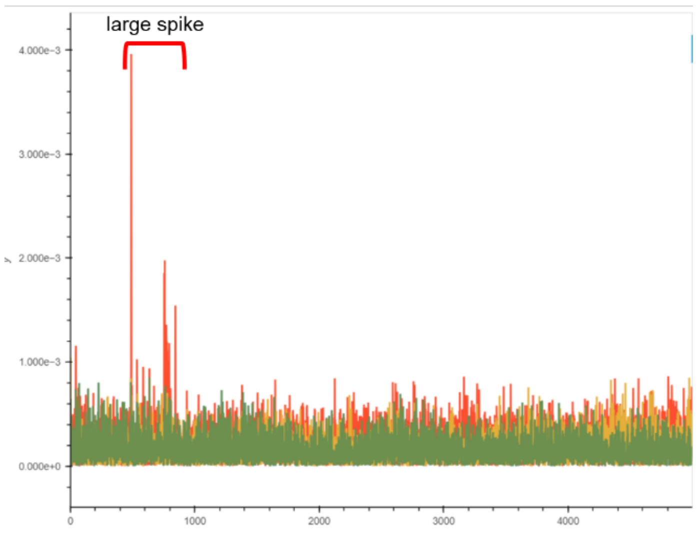


***Figure 14.*** A plot of the difference values of 4 key guesses (blue: 0x2A, red: 0x2B, green: 0x2C, yellow: 0x2D)

## VI. Future Work

### VII.1 Continuing with ChipWhisperer Labs

Moving forward, a potential future work we believe is to delve deeper into the ChipWhisperer labs provided, extending beyond the current focus on "SCA101: Part 3, Topic 3 - DPA on Firmware Implementation of AES." This initial lab served as a means to familiarize with the general functionalities of ChipWhisperer hardware and to showcase proficiency in power analysis SCA. The follow-up labs, such as SCA203 and SCA204 which utilizes the ChipWhisperer lite/pro/Huskey, will be helpful as they present variances in API compared to the ChipWhisperer Nano, which has been the primary focus this semester.It will be necessary, however, to look into the ChipWhisperer API for understanding the specialized methods used on the ChipWhisperer lite/pro/Huskey platforms.

### VII.1 Utilizing ChipWhisperer Husky for Experiments

Another potential path the group is currently planning is to explore the potential of utilizing the ChipWhisperer Huskey for conducting experiments, setting it as a prime candidate for implementing our ADC counter design and undertaking a power SCA in the upcoming semester. Unlike the ChipWhisperer Nano, the ChipWhisperer Husky offers the flexibility to select FPGA boards as target devices, presenting an opportunity to validate our hypothesis regarding the digital counter design. Acquiring the ChipWhisperer Husky Starter Kit will equip us with the necessary hardware components, including the Huskey device itself, an appropriate target board, and two target devices compatible with the target board (iCE40 FPGA and SAM4S Arm processor) [23]. While certain modifications will be required, we anticipate the following steps to utilize the ChipWhisperer Huskey in testing our ADC counter design against power-based SCAs:

1. Implementation of the ADC counter design onto the FPGA target device.
2. Initialization and setup of the ChipWhisperer Husky to initiate triggers and capture traces during counter activity.
3. Iteration through all potential values measurable by the counter, capturing power traces for each scenario.
4. Analysis of the resulting traces to assess their similarity and identify any abrupt changes. Consistent and uniform traces will indicate potential resilience against SPA attacks.
5. Hypothesizing specific difference values within the circuit that may serve as potential weak points for DPA attacks. Subsequent execution of the attack will validate the design's potential resilience against DPA threats if no significant deviations are observed.

Through these systematic steps, we will be able to validate the effectiveness and robustness of our counter design against both SPA and DPA attacks, proving our hypothesis of a robust side-channel preventive ADC model.

## VII. References

[1] NIST, "Glossary: Side-Channel Attack," Accessed 05/02/2023:

csrc.nist.gov/glossary/term/side_channel_attack

[2] NewAE Technology Inc., "ChipWhipserer-Nano", Accessed 05/02/2024:

www.newae.com/products/nae-cw1101

[3] NewAE Hardware Product Documentation, "CW1101 ChipWhisperer-Nano", Accessed 05/02/2024:

rtfm.newae.com/Capture/ChipWhisperer-Nano/

[4] Github, newaetech//chipwhisperer-jupyter, Accessed 10/02/2024:

github.com/newaetech/chipwhisperer-jupyter/blob/master/courses/sca101/Lab%203_3%20-%20DPA%20on%20Firmware%20Implementation%20of%20AES%20(MAIN).ipynb

[5] S. Mangard, E. Oswald and T. Popp, "Power analysis attacks: Revealing the secrets of smart cards" (2010a), Springer

[6] Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low- bandwidth acoustic cryptanalysis (extended version). In: IACR Cryptology ePrint Archive, 2013:857 (2013)

[7] Analog Devices, "ADC Architectures", Accessed 1/12/2023:

www.analog.com/en/technical-articles/adc-architectures.html

[8] Scottr9, “Integrator output voltage in a basic dual-slope integrating ADC”, Wikipedia, Accessed 1/12/2023:

en.wikipedia.org/wiki/Integrating_ADC#/media/File:Dual_slope_integrator_graph.svg

[9] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems - [KoC96],” In Neal Koblitz, editor, Advances in Cryptology CRYPTO '96, 16th Annual International Cryptology Conference, Santa Bar- bara, California, USA, August 18-22, 1996, Proceedings, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer, 1996

[10] P. C. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis - [KJJ99].” In Michael Wiener, editor, Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 388-397. Springer, 1999.

[11] K. Gandolfi, C. Mourtel, and F. Olivier, Electromagnetic Analysis: Concrete Results - [GMO01]. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings, volume 2162 of Lecture Notes in Computer Science, pages 251-261. Springer, 2001.

[12] J. Quisquater and D. Samyde, ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle At- tali and Thomas P. Jensen, editors, Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings, volume 2140 of Lecture Notes in Computer Science, pages 200-210. Springer, 2001.

[13] NewAE Technology, “ChipWhisperer-Nano”, Accessed 18/02/2024:

https://store.newae.com/chipwhisperer-nano/

[14] NewAE Technology, “CW305 Artix FPGA Target Board”, Accessed 18/02/2024:

https://store.newae.com/cw305-artix-fpga-target-board/

[15] NewAE Technology, “CW308 UFO Board”, Accessed 18/02/2024:

https://store.newae.com/cw308-ufo-board/

[16] ARM, “Application Note 321 ARM Cortex-M Programming Guide to Memory Barrier Instructions”, ARM, Sep. 2012

[17] ARM Developer, “Cortex-M0 Specification”, Accessed 20/02/2024:

https://developer.arm.com/Processors/Cortex-M0

[18] ARM Developer, “Cortex-M4 Specification”, Accessed 20/02/2024:

https://developer.arm.com/Processors/Cortex-M4

[19] J. Hu, Side-Channel Attacks on Analog-to-Digital Converters: A Survey and Comparison with Cryptos. TechRxiv. Nov., 2023

[20] G. Goodwill, B. Jun, J. Jaffe, P. Rohatgi: Cryptography Research Inc., A testing methodology for side channel resistance validation, NIST.

[21] Github, newaetech//chipwhisperer-jupyter, Accessed 10/02/2024:

github.com/newaetech/chipwhisperer-jupyter/blob/master/courses/sca204/CW305_ECC_part1.ipynb

[22] Github, newaetech//chipwhisperer-jupyter, Accessed 10/02/2024:

github.com/newaetech/chipwhisperer

[23] NewAE Technology, “ChipWhisperer-Huskey”, Accessed 18/03/2024:

rtfm.newae.com/Capture/ChipWhisperer-Husky/

[24] CrowdSupply, ChipWhisperer-Huskey, Accessed 18/03/2024:

www.crowdsupply.com/newae/chipwhisperer-husky

[25] A. Jayasena, E. Andrews and P. Mishra, "TVLA*: Test Vector Leakage Assessment on Hardware Implementations of Asymmetric Cryptography Algorithms," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 9, pp. 1269-1279, Sept. 2023

[26] NewAE Technology, “Question about trigger high/low in CWLite firmware: short op”, Accessed 20/03/2024:

forum.newae.com/t/question-about-trigger-high-low-in-cwlite-firmware-short-op/178/3

[27] Lattice Semiconductor., “iCE40 Ultra / Ultra Lite”, Accessed 20/03/2024:

www.latticesemi.com/Products/FPGAandCPLD/iCE40Ultra

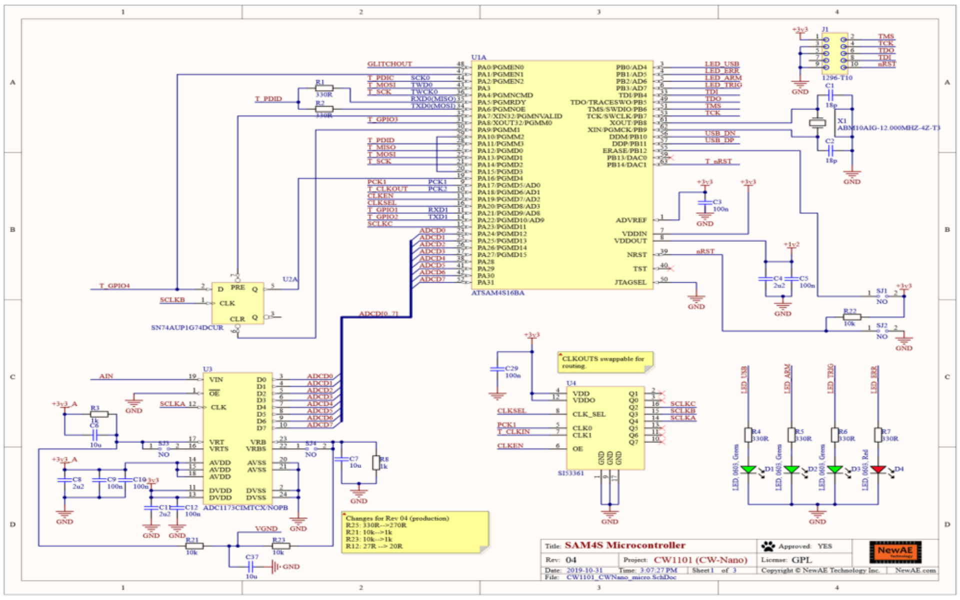
[28] T. Jeong, A. Chandrakasan, and H. Lee, “S2ADC: A 12-bit, 1.25-MS/s Secure SAR ADC With Power Side-Channel Attack Resistance.” IEEE Journal of Solid-State Circuits 56, 844–854, 2021

[29] NewAE Technology, “https://learn.chipwhisperer.io/courses/power-analysis-101” Accessed 20/03/2024:

https://learn.chipwhisperer.io/courses/power-analysis-101

## VIII. Appendix

### VIII.A ChipWhisperer Nano Schematic



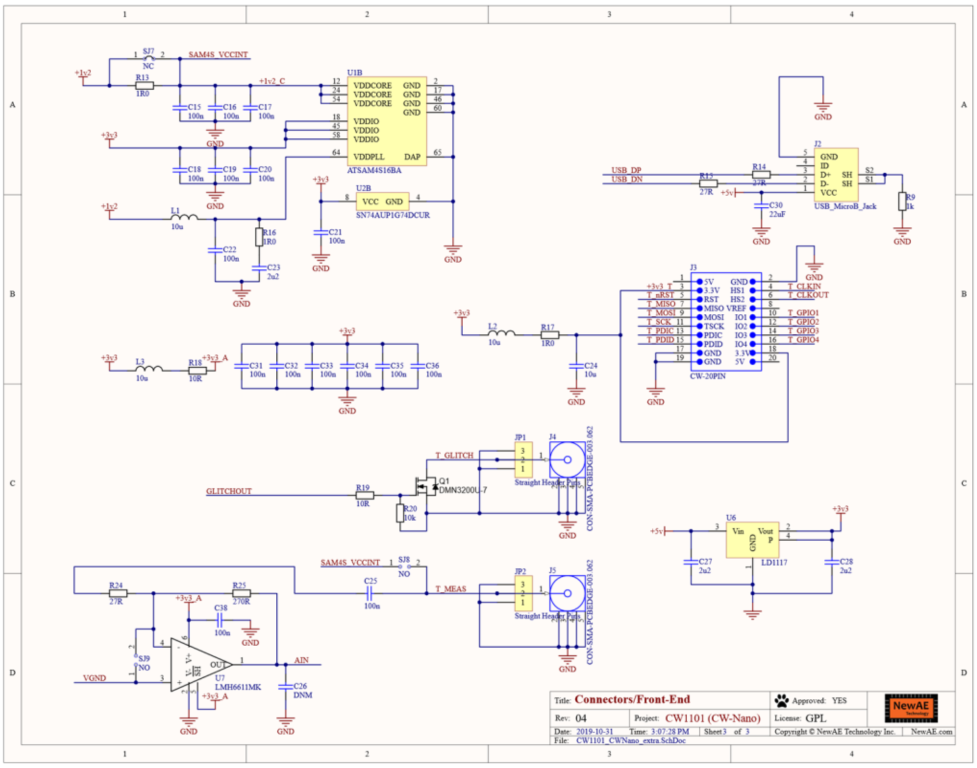


***Figure 15.*** A schematic of the ChipWhisperer Nano [3]

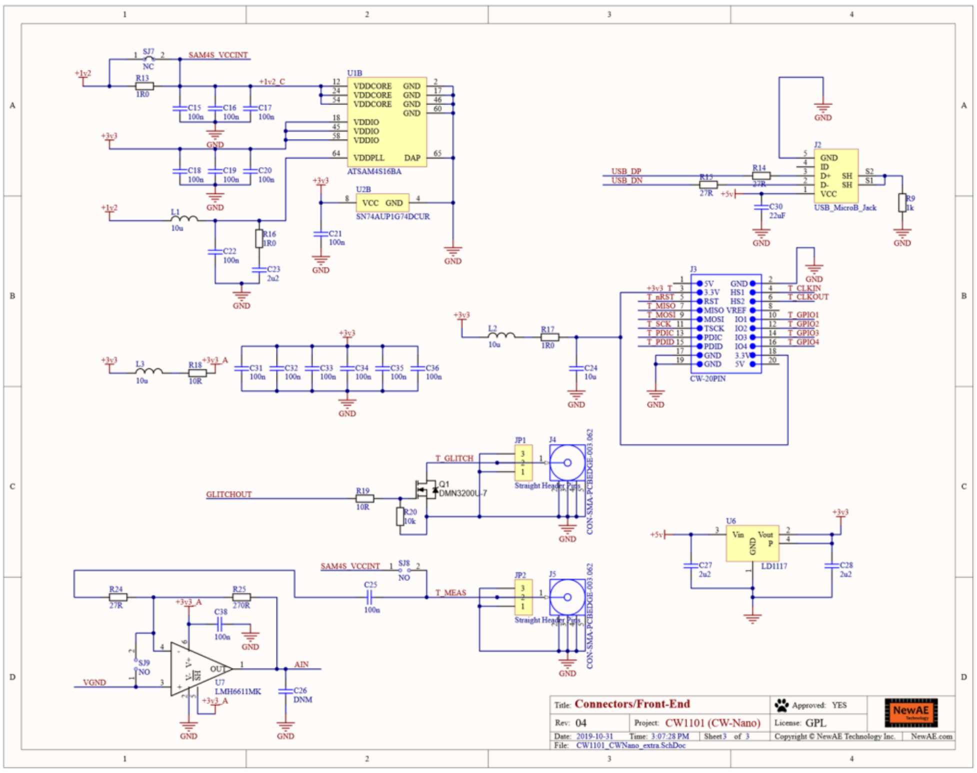


***Figure 16.*** A schematic of the ChipWhisperer Nano's target device [3]

## VIII.B Experiment – Code for Creating Power Traces

```
from tqdm.notebook import trange
import numpy as np
import time

ktp = cw.ktp.Basic()
trace_array = []
textin_array = []

key, text = ktp.next()

target.set_key(key)

N = 2500
for i in trange(N, desc='Capturing traces'):
    scope.adc.samples = 10000
    scope.arm()

    target.simpleserial_write('p', text)

    ret = scope.capture()
    if ret:
        print("Target timed out!")
        continue
    response = target.simpleserial_read('r', 16)

    trace_array.append(scope.get_last_trace(True))
    textin_array.append(text)

    key, text = ktp.next()
```

## VIII.C Experiment – Code for Analyzing Power Traces

```
import numpy as np
mean_diffs = np.zeros(256)

guessed_byte = 0

max_five = []

for guess in range(0, 256):

    one_list = []
    zero_list = []

    for trace_index in range(numtraces):
        hypothetical_leakage = aes_internal(guess, textin_array[trace_index][guessed_byte])
        if hypothetical_leakage & 0x01:
            one_list.append(trace_array[trace_index])
        else:
            zero_list.append(trace_array[trace_index])

    one_avg = np.asarray(one_list).mean(axis=0)
    zero_avg = np.asarray(zero_list).mean(axis=0)
    mean_diffs[guess] = np.max(abs(one_avg - zero_avg))

    print("Guessing %02x: %f"%(guess, mean_diffs[guess]))
```