Side-channel Attacks: ChipWhisperer

4/23/2024

^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 Tech

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
   - Integrating ADC
   - Hypothetical Attack on the Integrating ADC
   - Proposed Circuitry Design
   - Types of Side-channel Attacks
   - ChipWhisperer Hardware
   - ChipWhisperer Courses
   - AES
   - TVLA
   - Prior Research using ChipWhisperer

V. Experiment
   - Background
   - Setup
   - Capturing Power Traces
   - Analyzing Power Traces
   - Results

VI. Future Work
   - Continuing with ChipWhisperer Labs
   - Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Terminology

- **Cryptography (Cryptographic algorithms):** Mathematical functions that takes two input parameters (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)
- **Analog-to-Digital Converter (ADC)**: A system that converts an analog signal into a digital signal
- **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]
- **Trojan hardware**: A malicious modification of the circuitry of an IC chip. It is done during the design or fabrication of chip [6]
- **Advanced Encryption Standard (AES):** Cryptography algorithm that uses a public key to protect data using multiple rounds of substituting and shuffling, established by NIST

Terminology

- **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:** Power consumption from a target device while it encrypts/decrypts data, which 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]
- **Test Vector Leakage Assessment (TVLA)**: An assessment that evaluates the side-channel leakage of sensitive information from the hardware implementation of a design [25]

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

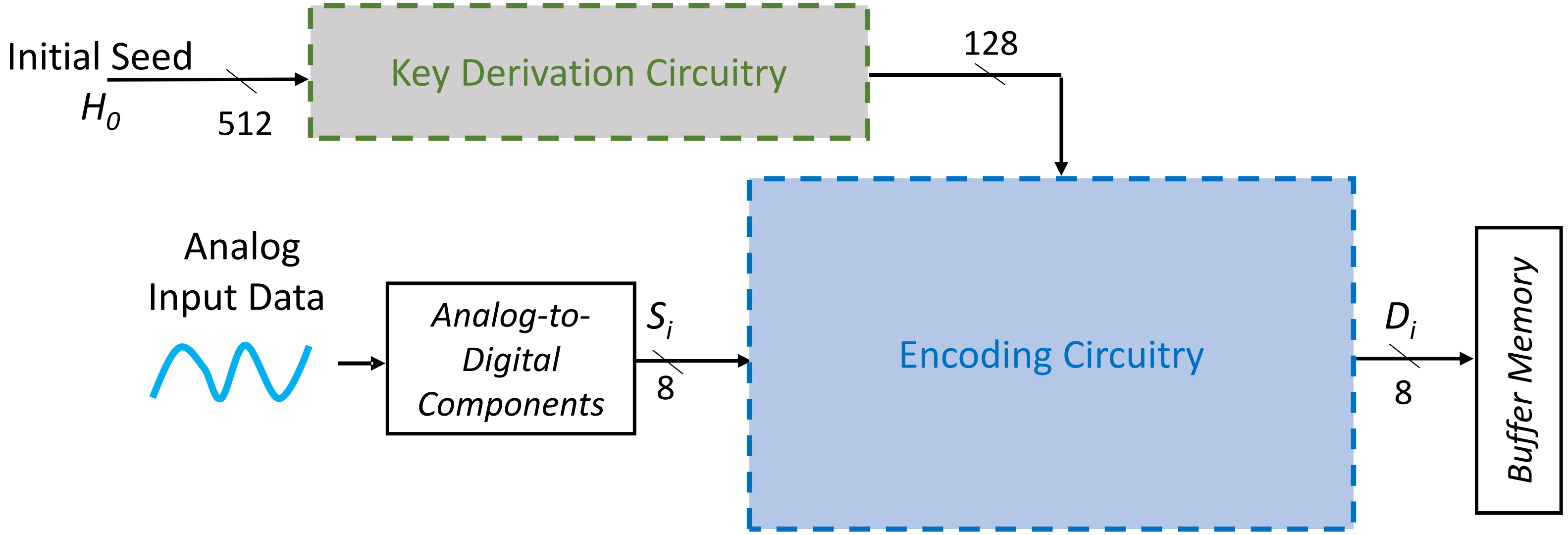
VII. References

VIII. Appendix

Motivation

- Despite various applications, current models of Integrating ADCs are vulnerable to **Trojan hardware** as well as **SCA**
  - Timing, power consumption, electromagnetic and acoustic emissions, etc.

- **Rancode encryption** provides a method to take analog inputs and produce encoded digital values which appear random to an adversary
  - For example, an image appears in memory as static noise
  - Therefore, leakage of the image pixels reveals nothing to an adversary



Motivation

- However, if the adversary could obtain a power side-channel on the analog to digital conversion, then the image could be obtained while bypassing Rancode encoding

- Based on the ideas from Kevin Hutto, the Analog design team is working on a potential solution to prevent power based side-channel attacks by adopting a dummy counter on the ADC

- The idea of the circuitry is based on hypothesis, and has not been proven or tested
  - While prior research has attempted to create a side-channel-preventive model of different ADCs such as Successive Approximation Register ADC or flash ADC, there are little to no work done on the Integrating ADC [28]

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Goals

- Create firm understanding of power SCAs through ChipWhisperer
- Understand and formulate a DPA based SCA experiment that could be done on power traces provided from the analog design team
- Understand, formulate and carry out a side-channel attack experiment on a real target device based on the ChipWhisperer Nano
- Roughly design a side-channel attack experiment on a target device with a closer relationship to our new ADC design equipped with a dummy counter

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
  - Integrating ADC
  - Hypothetical Attack on the Integrating ADC
  - Proposed Circuitry Design
  - Types of Side-channel Attacks
  - ChipWhisperer Hardware
  - ChipWhisperer Courses
  - AES
  - TVLA
  - Prior Research using ChipWhisperer

V. Experiment
  - Background
  - Setup
  - Capturing Power Traces
  - Analyzing Power Traces
  - Results

VI. Future Work
  - Continuing with ChipWhisperer Labs
  - Utilizing ChipWhisperer Husky for Experiments

VII. References

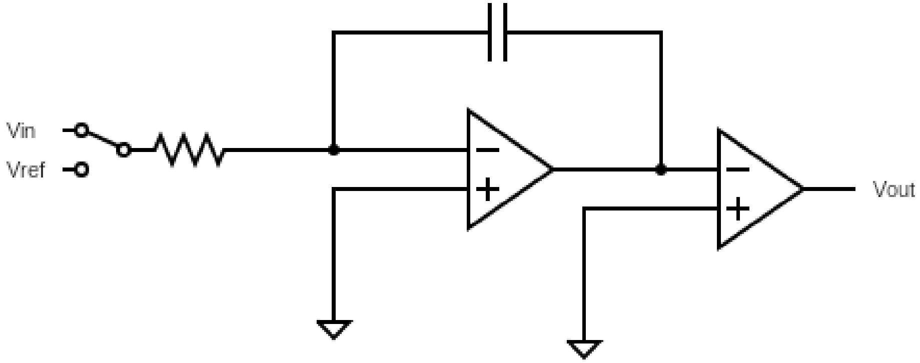
VIII. Appendix

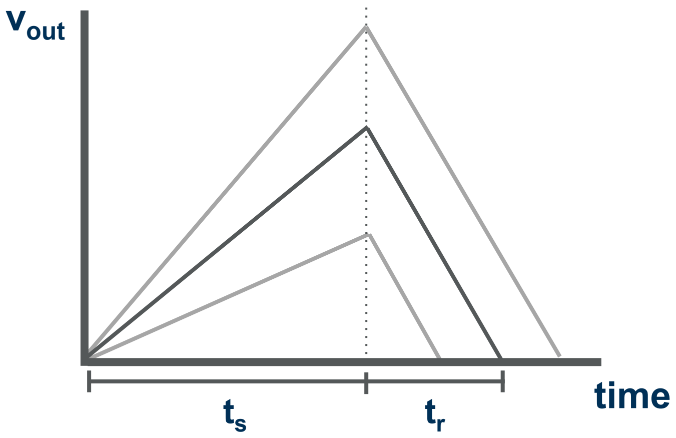
Integrating ADC

- **Components**: integrator, switch (selecting between measured and reference voltage), timer, comparator , and controller

- Two main conversions happen throughout the circuit [8]
  - **Run-up phase**: measured voltage ($v_{in,}$ positive) is set as input to the integrator; integrator ramps for a fixed period of time to charge the integrator capacitor
  - **Run-down phase**: reference voltage ($v_{ref,}$ negative) is set as input to the integrator; the time that it takes for the integrator's output to return to zero is measured through timer

- The time measured during the second phase is proportional to the final digital value





Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
  - Integrating ADC
  - Hypothetical Attack on the Integrating ADC
  - Proposed Circuitry Design
  - Types of Side-channel Attacks
  - ChipWhisperer Hardware
  - ChipWhisperer Courses
  - AES
  - TVLA
  - Prior Research using ChipWhisperer

V. Experiment
  - Background
  - Setup
  - Capturing Power Traces
  - Analyzing Power Traces
  - Results

VI. Future Work
  - Continuing with ChipWhisperer Labs
  - Utilizing ChipWhisperer Husky for Experiments

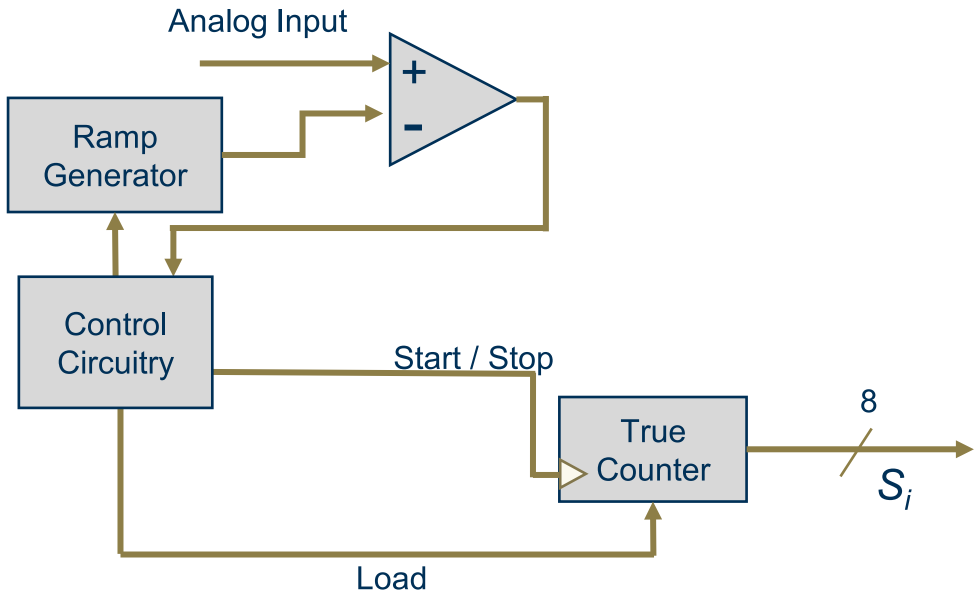
VII. References

VIII. Appendix

Hypothetical SPA attack on the Integrating ADC

- The current model of the Integrating ADC is prone to the most basic power SCA such as the SPA attack
  - Once going through a step of charging the ramp generator, the counter is run as much as the amount of the analog value
  - The amount of time that the counter starts counting which is leaked onto the power consumption, reflects the final digital value

- Based on theory, we can construct a hypothetical SPA attack on the plain Integrating ADC



Attack Surface

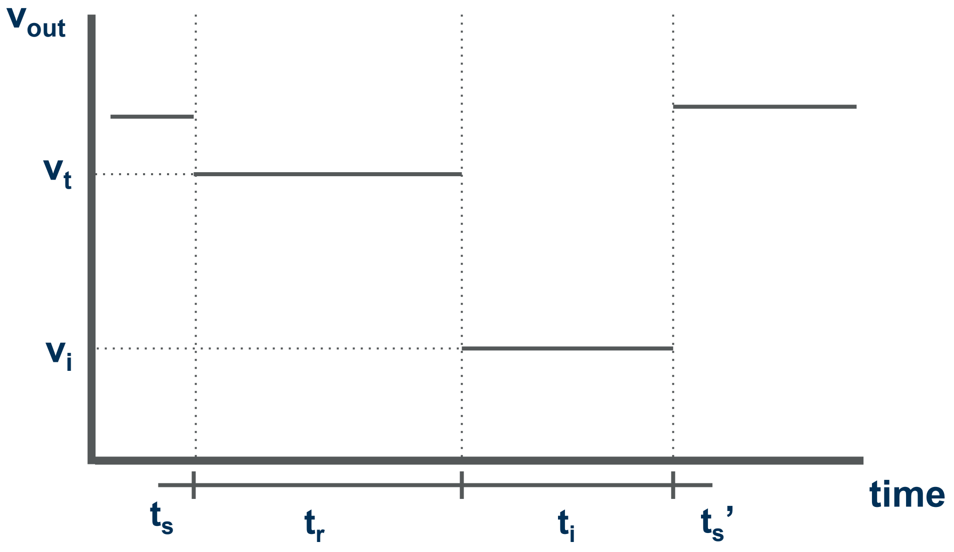
- The aim of the attack is to obtain the digital output from the ADC based on the leakage traces from its power consumption

- We assume the Integrating ADC is using the most basic implementation as described previously, without any protective measures such as masking or hiding

- Assume that the adversary has knowledge on the internal circuitry of the ADC and has an ADC with the same architecture to run on inputs and capture the power traces

Hypothetical SPA attack

- Since the plain Integrating ADC design is based on a single counter to measure the analog input, the power trace would show when the counter is active (t_r), proportional to the digital value
  - After the ramp generator is charged (t_s), the counter starts counting during the discharge phase and stops counting when the discharged value from the ramp generator matches the analog input (t_r)
  - It stays in an idle state until the counter value is processed (t_i)
  - Process is repeated as the ramp generator is recharged with the next analog value (t_i')



- SPA attack will be able to deduct t_r, which is enough to reveal the digital output

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

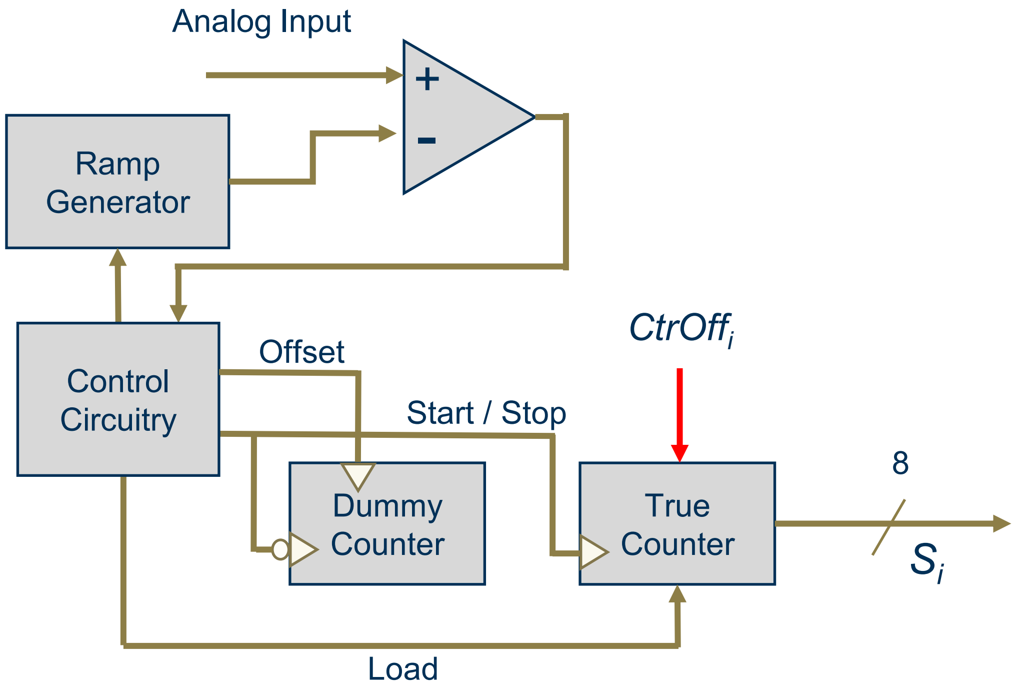
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

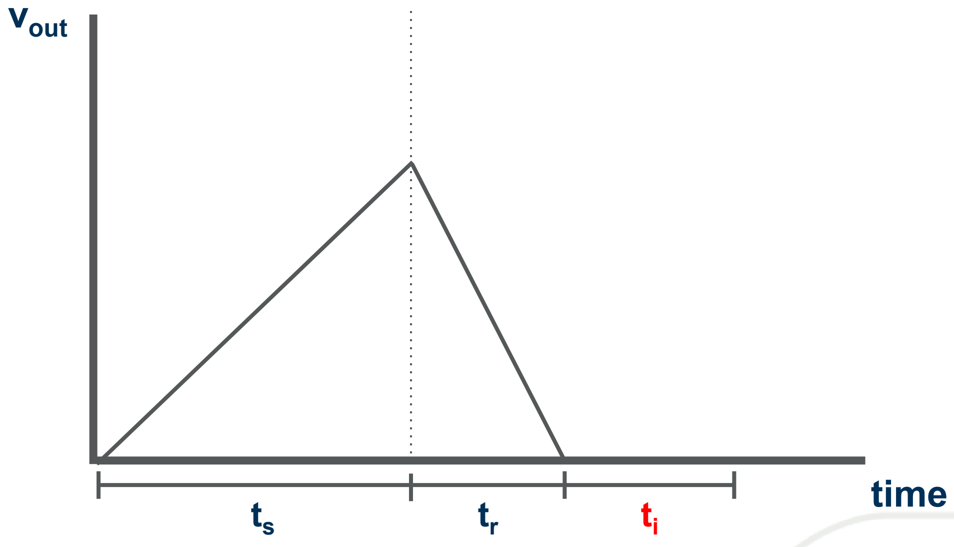
VII. References

VIII. Appendix

Proposed Circuitry Design

- High level idea: using a **dummy counter** in addition to the true counter to constantly be counting for a set length of round
    - Prevents an adversary from simply spotting the time the counter had run to calculate the digital output





$$t_s = t_r + t_i$$

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

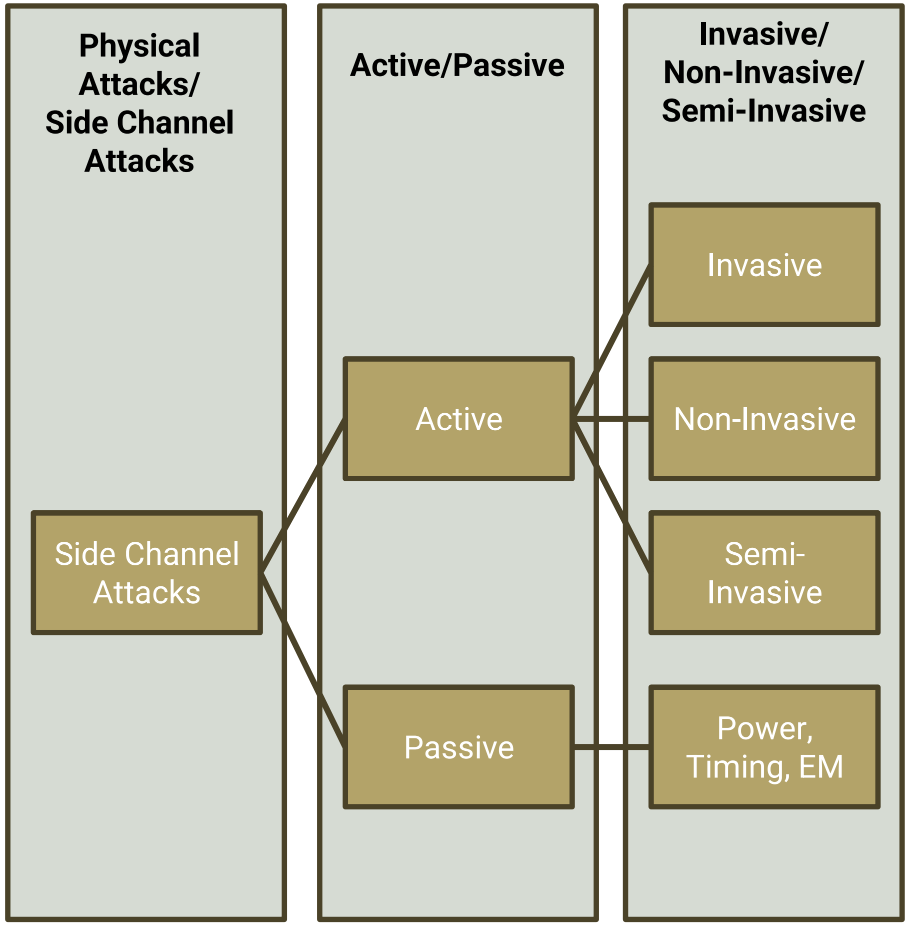
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Types of Side-channel Attacks

- **Types of attacks on cryptographic devices** [5]
    - **Passive:** cryptographic device is operated within its specification, secret key is revealed by observing **physical properties** of the device (e.g. execution time, power consumption).
    - **Active:** the **inputs and/or environment** of the device are manipulated to make the device behave abnormally, secret key is revealed by **exploiting this abnormal behavior.**
    - **Invasive:** strongest type of attack. Typically starts with **depackaging the device,** and then different components of device are accessed directly using a **probing station**.
        - Could be either **passive** or **active.**
        - Typically requires expensive equipment



Types of Side-channel Attacks

- **Types of attacks on cryptographic devices** [5]
  - **Semi-Invasive:** Also de-packages device, but the **passivation layer (**no direct contact to a chip surface) stays intact.
    - **Passive**: goal is typically to **read out content of memory cells without using/probing normal read-out circuits by using side-channel methods such as X-rays, electromagnetic fields, or light.**
    - **Active**: goal is typically to induce faults in device by **using X-rays, electromagnetic fields, or light.**
    - Typically doesn't require expensive equipment, but overall is quite expensive and time-consuming as it has to **locate the right position for an attack** on the surface of a modern chip
  - **Non-Invasive:** cryptographic device is attacked **as it is**, i.e. **only directly accessible interfaces are exploited.**
    - Device is not permanently altered, therefore **no evidence of an attack is left behind**.
    - Typically can be conducted with **relatively inexpensive equipment**, hence poses a serious practical threat to the security of cryptographic devices.
    - **Side-channel attacks:** passive non-invasive attacks. Consists of **timing attacks** [9]**, power analysis attacks** [10]**, electromagnetic attacks** [11][12]

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

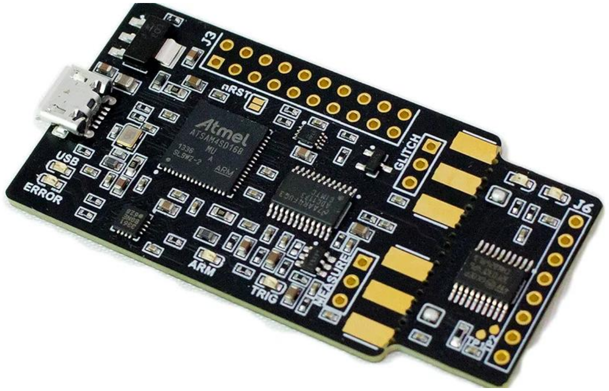
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

ChipWhisperer: CW1101 ChipWhisperer-Nano

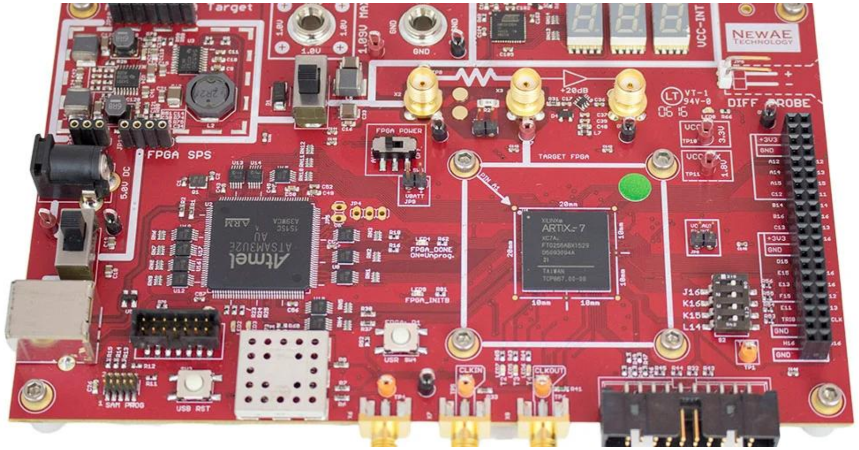
- **Specifications** [2]
  - A tool used for side-channel power analysis training, performing attacks against algorithms such as AES using power analysis.
  - Detachable target device allows for attacks on an external target in conjunction with ChipWhisperer suite of devices
  - Omission of FPGA keeps cost down, but limits functionality
  - The device consists of two parts:
    - **Capture Device:** The capture device utilizes a 8-bit 20 MS/S analog to digital converter that can be used to capture power traces of the internal target and external targets
    - **Target device:** The removable on-chip target device (**STM32F0**) uses the **ARM Cortex-M0** chip which can be programmed with various algorithms for simulations.
  - Uses Jupyter-based environment to control the main board as well as the target device
  - Cost: $50.00 [13]



[2]

ChipWhisperer: CW305 Artix FPGA

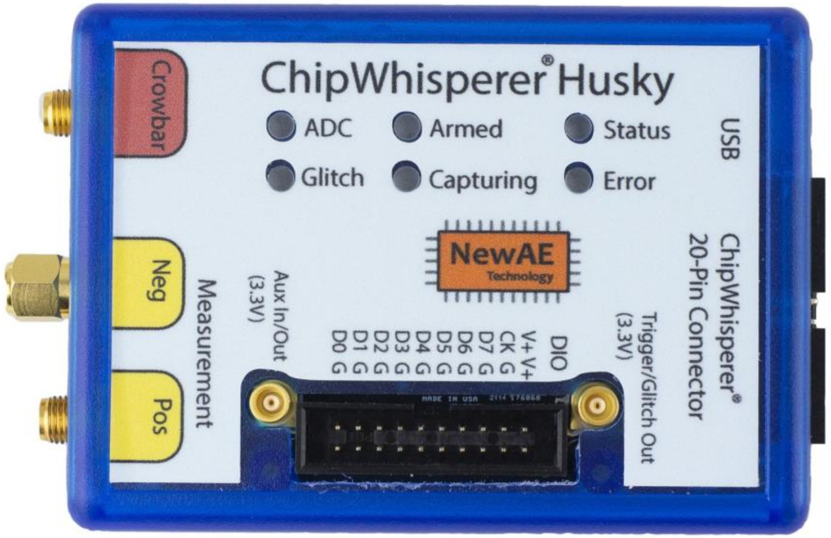
- **Specifications** [4]
  - Standalone FPGA target board made by NewAE. It utilizes CW's proprietary 20 pin interface and features a programmable VCC supply alongside a phase-locked loop for clocking the FPGA
  - The board requires an external oscilloscope or capture box for performing the power measurement
  - Custom USB interface provides address/data bus for FPGA, including data transfer and configuration
  - **Target device:** Xilinx 7 Series FPGA which can be programmed over USB and JTAG and be used as a standalone device
  - Cost: $1,000.00 [14]



[4]

ChipWhisperer: ChipWhisperer-Husky

- **Specifications** [23]
  - A side-channel power analysis tool being the successor of the ChipWhisperer-Lite (containing capabilities more diverse than the ChipWhisperer-Nano)
  - Through the ChipWhisperer 20-pin Connector, the device can connect to numerous targets as long as they can be equipped onto a correct ChipWhisperer target board
  - The Huskey has an internal FPGA which the user can connect to through a 20-pin input and be used as trigger inputs for the device
  - When analyzing traces of ARM processors, a dedicated Arm trace sniffer TraceWhisperer made by NewAE can be used to easily analyze the traces
  - **Capture Device:** A12-bit 200 MS/S analog to digital converter that can be used to capture power traces of the internal target and external targets
  - As same with all other ChipWhisperer capture devices, it uses Jupyter-based environment to control the device
  - Cost: $549.00 [24]

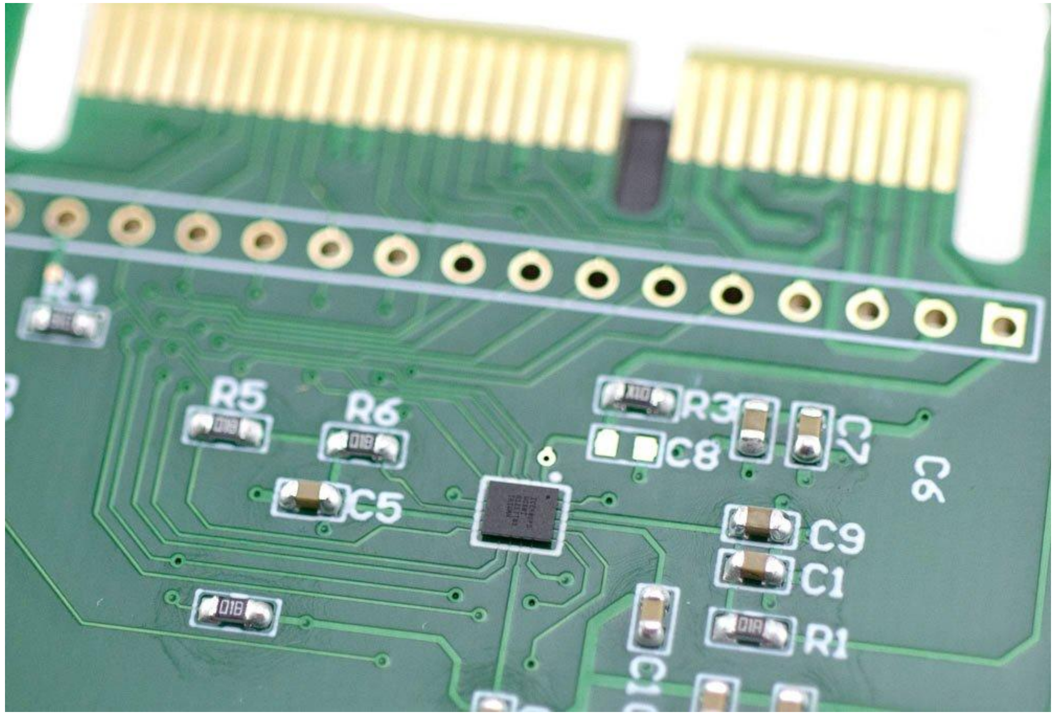


[23]

Lattice iCE40 FPGA

- **Specifications** [27]
  - A small low-power intended FPGA with a form factor of 1.4 mm x 1.4 mm x 0.45 mm
  - Features up to 3,520 4 input Lookup Tables, up to 26 I/Os for customized interfaces and up to 80 Kbits of embedded distributed memory
  - Capable of running FPGA design tools (ex. Verilog), demos and reference designs and kick start designs

- Included in the ChipWhisperer-Husky Starter Kit as a target device that can be connected to the provided CW313 target base-board [24]



[23]

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

ChipWhisperer Courses

- ChipWhisperer is an open-source toolchain for power analysis SCA and glitching attacks
  - The Software for the ChipWhisperer device (Python API for capturing and analyzing) as well as basic tutorial courses on the device are provided on the ChipWhisperer-jupyter project in github under NewAE.

- There are 6 courses for SCA and 2 courses for fault attacks [22]
  - SCA 101 and SCA 201 focuses on power analysis attacks on AES, SCA 202 on power analysis on asymmetric implementations, SCA203 (incomplete) on leakage assessment, SCA204 and SCA 205 on power analysis on hardware/software Elliptic Curve Cryptography

- Each course consists of 1~4 labs each having a jupyter notebook with explanation on the steps of procedure, example code, and solution code and output

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background

- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment

- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work

- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Advanced Encryption Standard (AES)

- Encrypts data into a fixed 128 bit block under a key with the same length
    - Keys can be 128, 192, or 256 each being AES-128, AES-192, AES-256 respectively
- Data and key are represented as an array of bytes with 4 rows and 4 columns (state)
- AES is key-iterated
    - Round transformation is repeatedly applied to state, with 10 repetitions when using AES-128
    - Round key which is generated by a key scheduling algorithm is set for each repetition
    - Decryption works similar to encryption with the round keys being applied in reverse order using the inverse of the round transformation
    - Each round transformation is made of 4 steps
        - AddRoundKey: Individual bytes of the state XORed with the round key
        - SubBytes: Byte substitution on the individual bytes of the state using a lookup table
        - ShiftRows: Shifts the row by a different offsets
        - MixColumns: Takes a column of the state and performs matrix multiplication
    - Final round (10th) skips the mix column operation for efficiency

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Test Vector Leakage Assessment (TVLA)

- An examination of a device to see if it is susceptible to data leakage by of unmasked cryptographic quantities through side channel attacks [25]

- Device fails if data can be deduced through probing (electromagnetic analysis) or signal tapping (power)

- Two classes of TVLA tests:
  - General: Any leakage of information that is directly dependent on input data or key
  - Specific: Leakage reveals intermediary data that can be exploited to reveal hidden data

- A specific test failures guarantees that leakage can be exploited to recover secret data, while a general test failure allows the possibility of exploitation

- Test is executed by comparing the power traces of running a fixed text vs random text, if they are similar by Welch's T-Test then there is a high likelihood that the device is susceptible to attacks

- All rounds of encryption should be examined

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background

- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment

- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work

- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Prior Research using ChipWhisperer

- "A Deep Learning Technique for Efficient Side-channel Attacks"
    - Focuses 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
    - Paper goes through description of statement and proof, simulation on an ARM Cortex-M architecture through micro-ecc framework, and a real device experiment through ChipWhisperer (STM32F415)

- The research utilized the ChipWhisperer CW308 UFO board along with the STM32F415 target chip manufactured by NewAE
    - CW308 UFO board was only used for capturing the power traces of the target device running the fixed routines, which were then passed through deep learning procedures
    - Price: $300.0 [15]
    - Following method would have to be adjusted to be used in our case, as current goals focuses on circuitry level implementation which may be better fit on a FPGA board

Difference on Target Processors

- The research uses STM32F415 chip as the target device, which is equipped by an ARM Cortex-M4 processor. The ChipWhisperer Nano we currently have has the STM32F0 as an on-chip target device, using the ARM Cortex-M0 processor

- Similarities [16]:
  - Single core processors
  - Does not have any internal cache memory, although it is possible for a SoC design to integrate a system level cache (Still looking for STM32F415 and STM32F0 documents to check)
  - 3 stage pipeline
  - 32-bit processor

- Differences [17][18]:
  - While the M0 was made to run basic tasks, the M4 was made to run heavier workloads, and thus consumes more power (Dynamic power of 151µW/MHz vs. 66µW/MHz, roughly x2.5)
  - M4 contains additional instruction in the ISA in order to suit its purpose of executing advanced workloads

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Experiment Background

- Utilized labs provided from ChipWhisperer (SCA101: Part 3, Topic 3 - DPA on Firmware Implementation of AES [4]) as a way to familiarize with their devices

- This lab emulates a real world DPA, and uses the CW Nano both as the target device and the capture device
  - Uses the on-chip STM32F0 target device to physically run the AES encryption process
  - Uses power analysis capabilities of the ChipWhisperer Nano to capture the power traces from the target device to guess the key used for AES

- The key focus of the lab is on the **AddRoundKey** and **SubBytes** step of AES, as the output from the steps is directly related with the round key that was accounted

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

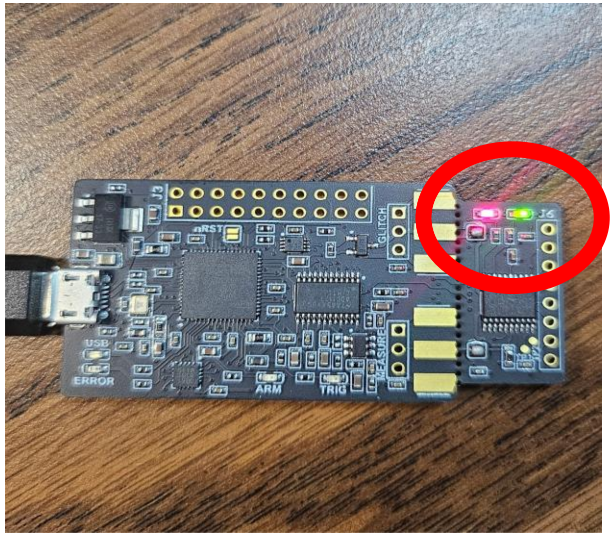
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Experiment Setup

- **Uploading AES Firmware:**
  1. Initial setup procedures on setting up Python, ChipWhisperer bash, and Jupyter lab, and connecting the ChipWhisperer Nano to the PC should be completed
  2. Through ChipWhisperer-bash, upload the appropriate firmware (…/simpleserial-aes) from the PC to the connected ChipWhisperer Nano
     - Despite not being specified on the document provided from ChipWhisperer, one has to add an additional line (“PLATFORM = CWNANO”) to correctly upload the file
  3. Once the light on the top-right part of the ChipWhisperer Nano turns red-green, the scope has been set correctly for the ChipWhisperer Nano to capture



Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

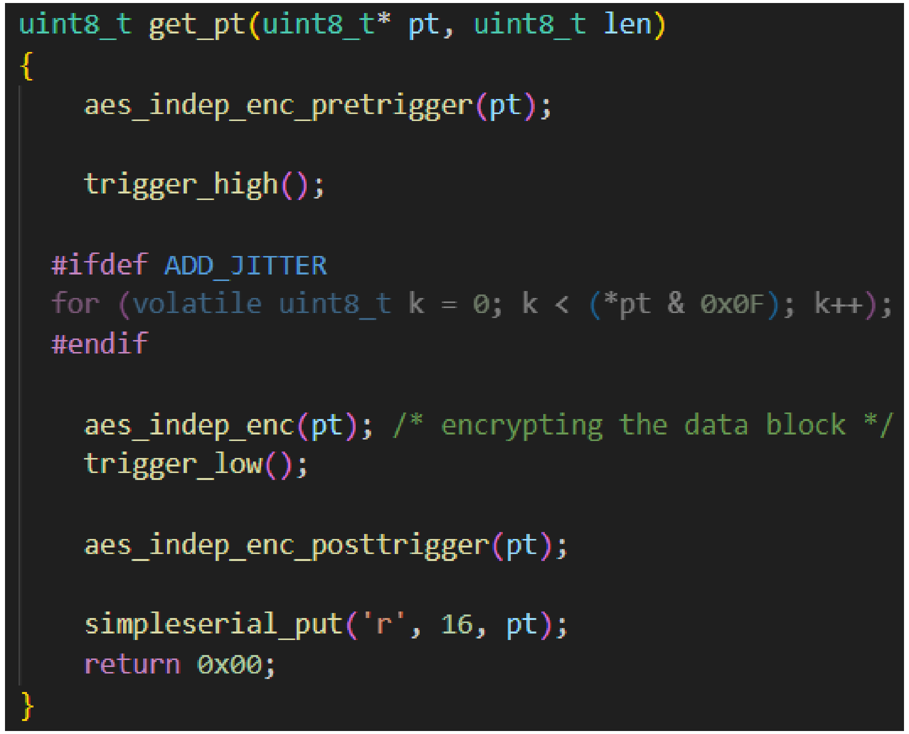
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

ChipWhisperer’s Capture firmware

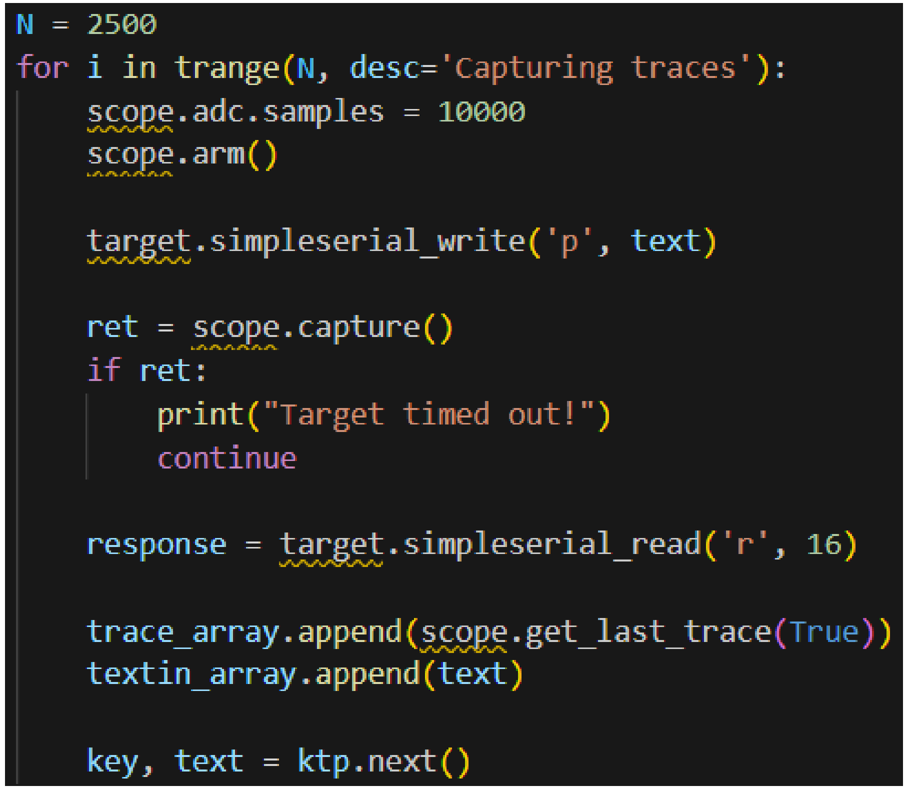
- Within the target firmware (AES-128) provided in the lab, ChipWhisperer uses it’s internal library to start and stop the capture of an individual execution of AES
  - trigger_high(): Under normal circumstances, it will start the capture of the power traces. The duration will be the number of samples the user had previously defined before the trigger.
  - trigger_low(): Currently, the team is not sure of its use. While trigger_high() starts a capture, trigger low does not end a capture [26]. The capturing process only ends when the designated number of samples have been captured.



```
uint8_t get_pt(uint8_t* pt, uint8_t len)
{
    aes_indep_enc_pretrigger(pt);

    trigger_high();

  #ifdef ADD_JITTER
  for (volatile uint8_t k = 0; k < (*pt & 0x0F); k++);
  #endif

    aes_indep_enc(pt); /* encrypting the data block */
    trigger_low();

    aes_indep_enc_posttrigger(pt);

    simpleserial_put('r', 16, pt);
    return 0x00;
}
```

ChipWhisperer’s Capture firmware

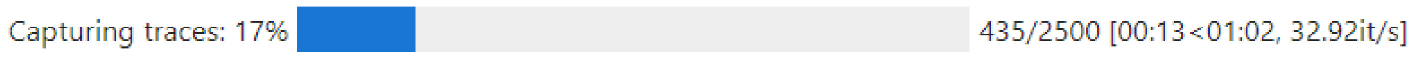
- By placing trigger_high() before each execution of AES and setting the number of samples to not exceed the length of 1 execution, we obtain the power trace that starts and end equally for all executions
  - Since it is difficult to manually calculate the exact number of samples necessary to cover the range of 1 execution, went through a trial-and-error process on choosing the correct sample value
  - As AES-128 goes through 10 rounds of similar procedures with the last round being relatively shorter, we were able to interpret from power trace graph if it was showing all rounds of the AES encryption process



```
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()
```

Capturing Power Traces

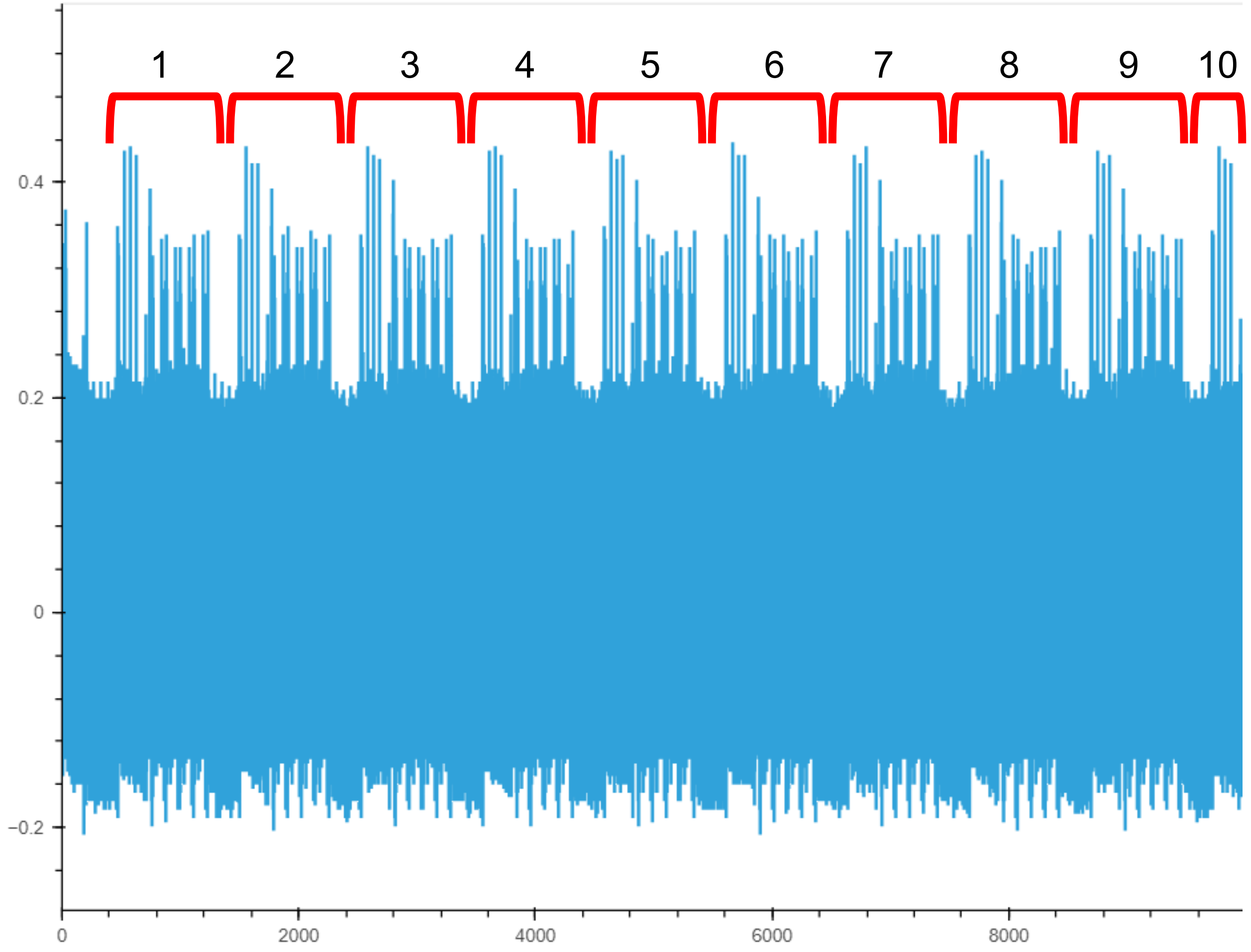
- **Running the Firmware** and **Capturing Traces**:
  1. Through the methods provided from ChipWhisperer API, we set the key and plaintext, encrypt the text, and capture the power traces
     - ktp.Basic(): Generates basic keys and plaintexts
     - simpleserial_write(‘p’, text): Encrypts text (16 bytes) through the simpleserial-aes firmware implemented on the device
     - simpleserial_write(‘p’, 16): Reads response from the target device
     - capture(): Captures power traces from the target device
  2. Repeat the process for 2500 times for accuracy of DPA attack
  3. If correctly captured, the following should appear on Jupyter lab and all 2500 power traces should be captured



```
Capturing traces: 17%  435/2500 [00:13<01:02, 32.92it/s]
```

Capturing Power Traces

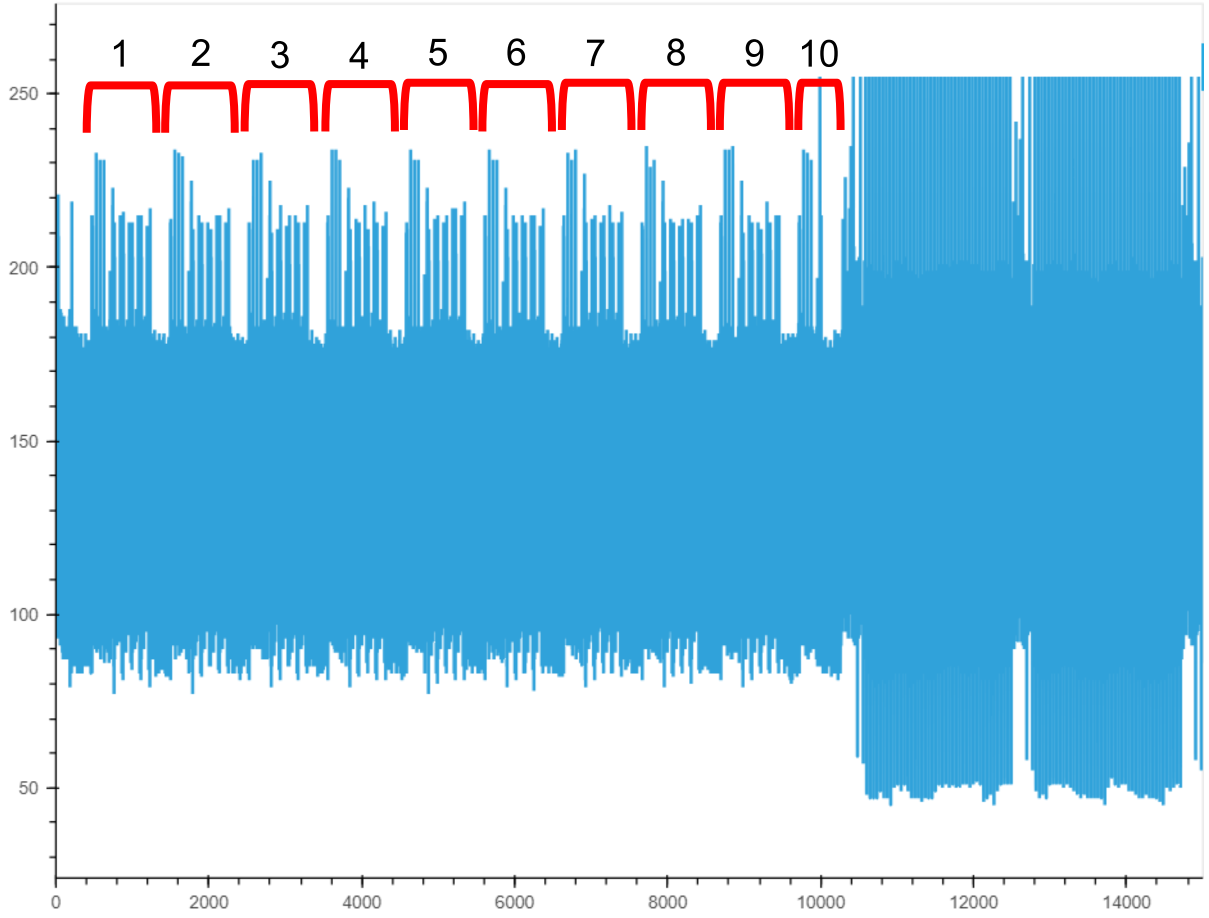
- By running the necessary instructions, the power trace for encrypting each of the 2500 different plain texts through AES encryption can be obtained
  - *get_last_trace*(): Returns the recorded power trace containing 5,000 samples (default sample size can be changed) where values are scaled and shifted to be between -0.5 and 0.5 (can optionally provide raw 8-bit value)



Effects of Larger Sample Size

- When setting the sample size much larger than a single execution (15,000), observed that the sample values after execution are generally random and does not seem to have relationship to the AES process at all

- After the end of the AES-128 encryption, the ChipWhisperer still captures traces to fill up all samples, while possibly being in an idle state



Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

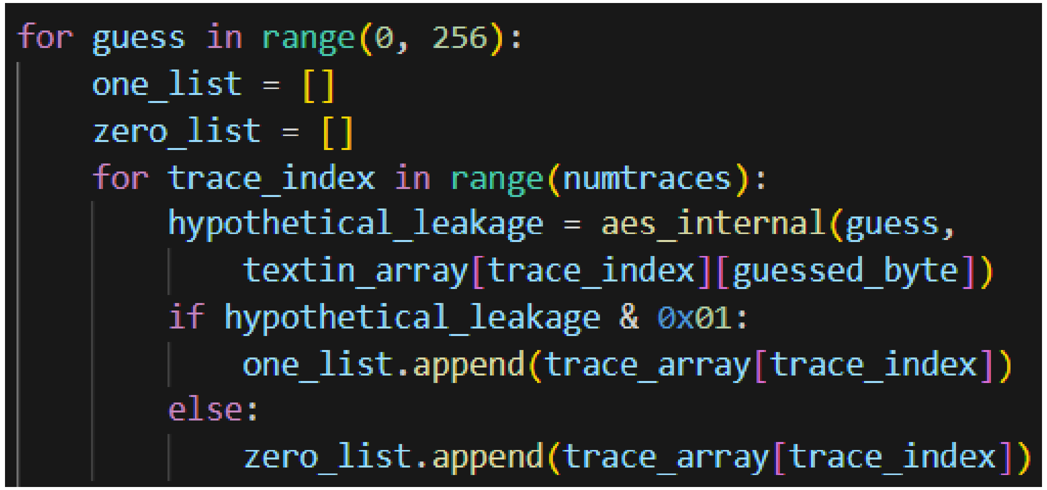
VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Analyzing Power Traces

- Assuming that we know the lookup table and steps of AES, we can test all 256 cases of potential values for the key byte with the actual plain text, mask a single bit (ex. least significant bit), and based on the bit (0 or 1) assign the corresponding power trace into two sets (0 or 1)
    - aes_internal(): a function that calculates the first step of AES procedure where the key and the plaintext is XORed, and then looked up through the lookup table
    - textin_array[]: an array containing the plaintext that was used to obtain each of the 2500 power traces



```
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])
```

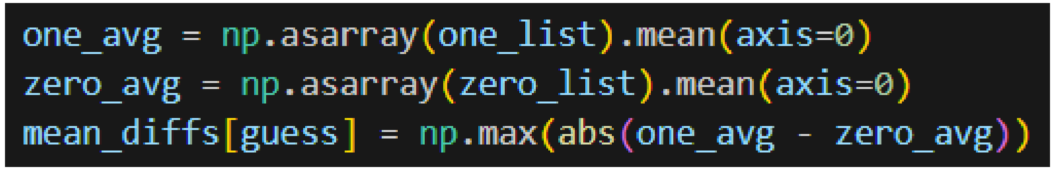
Analyzing Power Traces

- Once all power traces are divided, calculate the average value of each of the group and calculate the difference of these averages

- For the correct key value, the specific bit would have affected the AES process
  - The certain step within the entire AES process (ex. 1st round's AddRoundKey and SubBytes step) would show immediate correlation in power consumption through vectors such as hamming weight

- As step progresses, correlation between the bit and the power consumption decreases

- Thus, select maximum over all differences to easily find the certain step that showed clear correlation



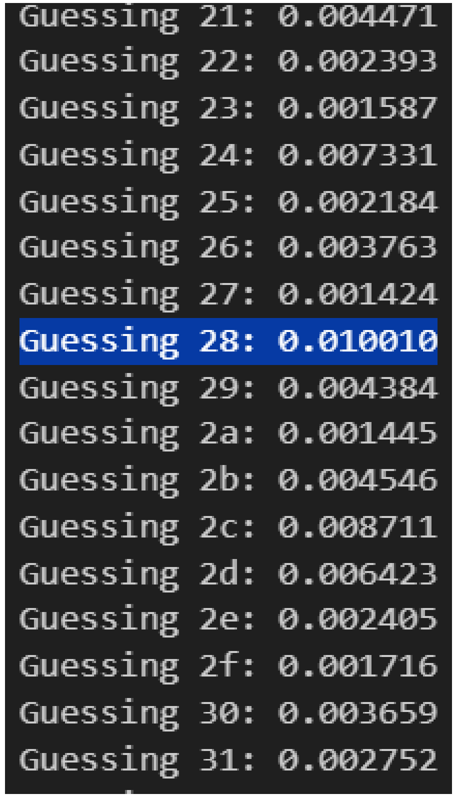
```
one_avg = np.asarray(one_list).mean(axis=0)
zero_avg = np.asarray(zero_list).mean(axis=0)
mean_diffs[guess] = np.max(abs(one_avg - zero_avg))
```

Analyzing Power Traces

- For any other none key value, the result should be generally random and have a value close to 0
    - Even selecting the maximum among them should not have large deviation

- Lining up all difference values from the guesses, the guess with the largest difference is most likely the correct byte value of the key

- Repeating the process for the rest of the 15 bytes, results in obtaining the whole 128 bit key



```
Guessing 21: 0.004471
Guessing 22: 0.002393
Guessing 23: 0.001587
Guessing 24: 0.007331
Guessing 25: 0.002184
Guessing 26: 0.003763
Guessing 27: 0.001424
Guessing 28: 0.010010
Guessing 29: 0.004384
Guessing 2a: 0.001445
Guessing 2b: 0.004546
Guessing 2c: 0.008711
Guessing 2d: 0.006423
Guessing 2e: 0.002405
Guessing 2f: 0.001716
Guessing 30: 0.003659
Guessing 31: 0.002752
```

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

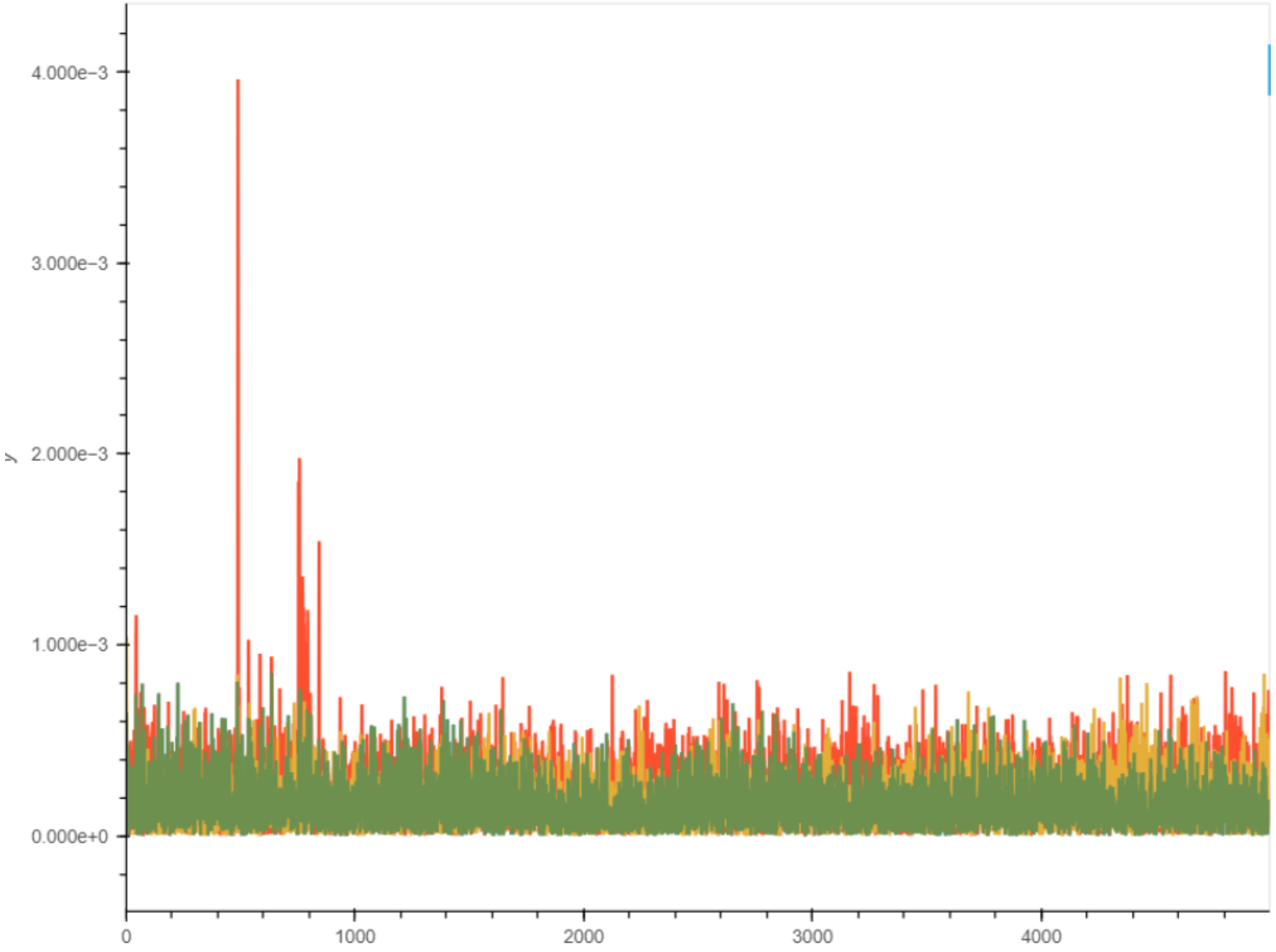
VIII. Appendix

Results: Interpreting Difference values

- This graph plots the difference value between the power consumption and the mean power trace for each key

- Each color represents a different possible key:
  - Blue: 0x2A
  - Red: 0x2B (correct key)
  - Green: 0x2C
  - Yellow: 0x2D

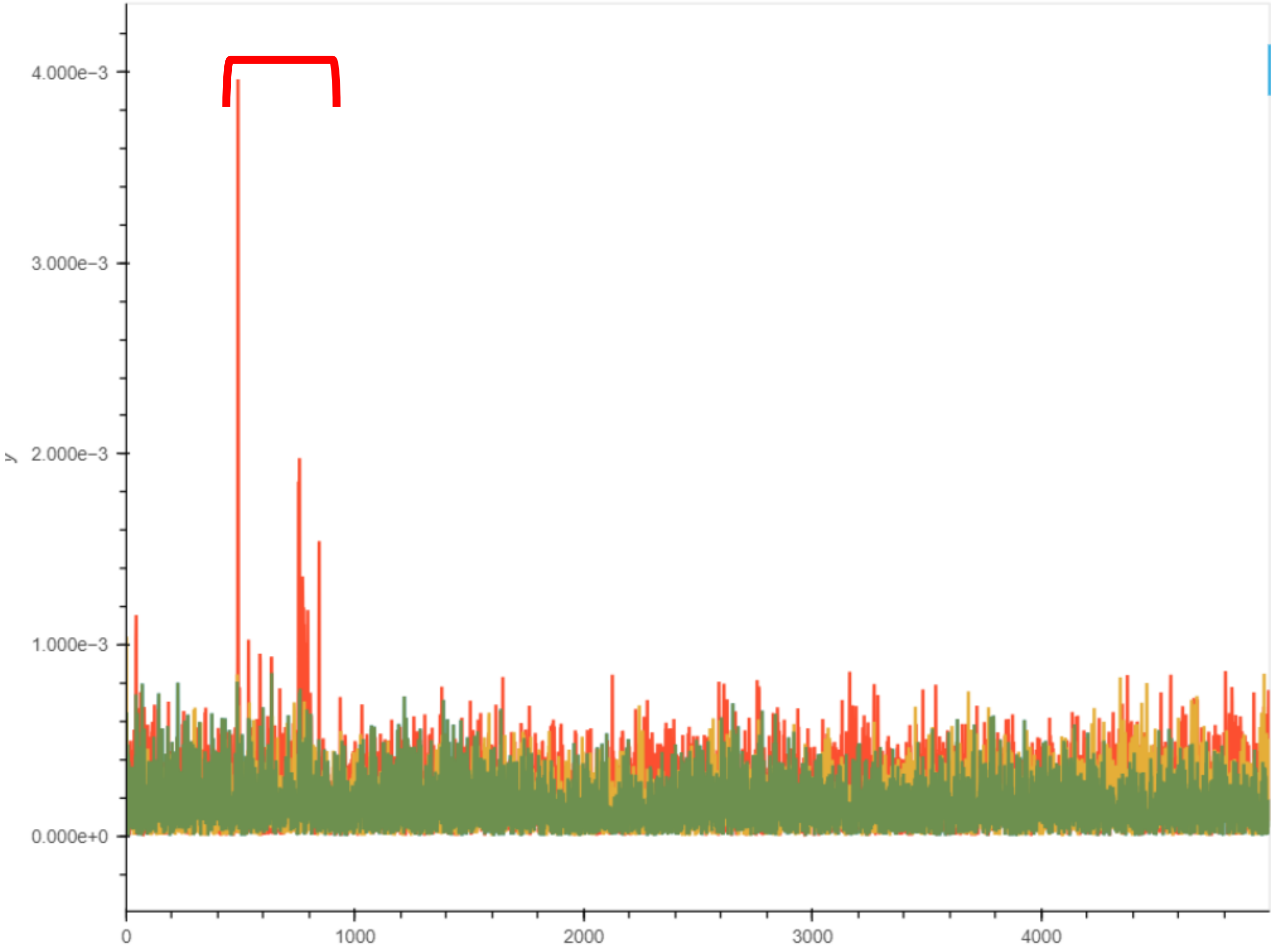
- For any other difference values, the power traces are divided randomly and thus the plot hovers close to 0



Results: Interpreting Difference values

- For the correct key value (red: 0x2B), it has a large spike in the initial part of the plot
  - 1st round AddRoundKey and SubBytes step

- From this it can be concluded that power consumption of the microcontroller is dependent on value 0x2B, which is likely the key for the certain byte



Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Continuing with ChipWhisperer Labs

- Work through labs further provided by Chipwhisperer
  - Current semester, team has focused solely on “SCA101: Part 3, Topic 3 - DPA on Firmware Implementation of AES” for the purpose of familiarizing with the general use cases of the ChipWhisperer hardware as well as demonstrating knowledge on SCA
  - SCA203, SCA204 and any other lab using the ChipWhisperer lite/pro/Huskey will be helpful as they have differences in API compared to ChipWhisperer Nano we have worked on this semester.

- Will require to search through ChipWhisperer API for further detail and methods that are specific to ChipWhisperer lite/pro/Huskey

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

Utilizing ChipWhisperer Husky for Experiments

- Review documentation and plan experiments using the ChipWhisperer-Huskey, a candidate for implementing our counter design and attempting an SCA next semester [23]

- Unlike the ChipWhisperer Nano, can select FPGA boards as target devices.
  - Would allow us to implement group's ADC counter design onto the FPGA board and attempt a power SCA to test our hypothesis on the design

- Purchasing the ChipWhisperer-Husky Starter Kit provides the Huskey hardware as well as an appropriate target board, and two target devices that can be used with the target board (iCE40 FPGA, SAM4S Arm processor)

Potential Steps on Using ChipWhisperer Huskey

- While large and small modifications will be necessary, these are the presumed steps for using the ChipWhisperer Huskey to test our ADC counter design against power SCA
  1. Implement the ADC counter design onto the FPGA target board
  2. Initialize and setup ChipWhisperer Huskey to start trigger and capture traces when the counter is active
  3. Repeat the step for all possible values the counter can measure and capture the power traces from each of the cases
  4. Analyze the resulting traces. If traces are all similar and show no abrupt changes, we show that the counter design is safe against SPA attacks.
  5. Hypothesize certain difference values within the circuit for potential weak spots of DPA attack and proceed with the attack. If no values present such results, we show that the counter design is safe against DPA attacks.

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

VIII. Appendix

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

References

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

References

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/

References

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," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 9, pp. 1269-1279, Sep. 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

Contents

I. Terminology

II. Motivation

III. Goals

IV. Background
- Integrating ADC
- Hypothetical Attack on the Integrating ADC
- Proposed Circuitry Design
- Types of Side-channel Attacks
- ChipWhisperer Hardware
- ChipWhisperer Courses
- AES
- TVLA
- Prior Research using ChipWhisperer

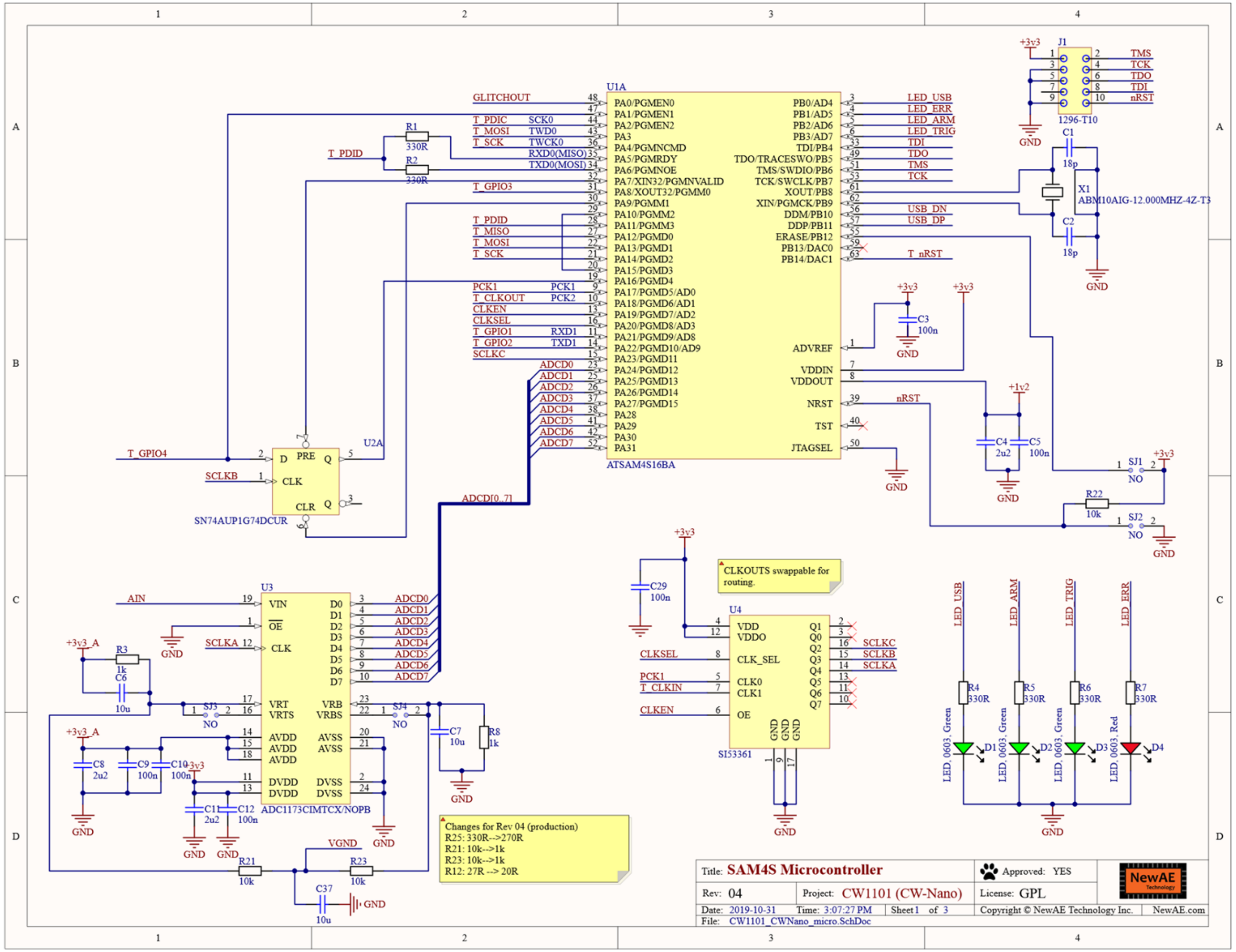
V. Experiment
- Background
- Setup
- Capturing Power Traces
- Analyzing Power Traces
- Results

VI. Future Work
- Continuing with ChipWhisperer Labs
- Utilizing ChipWhisperer Husky for Experiments

VII. References

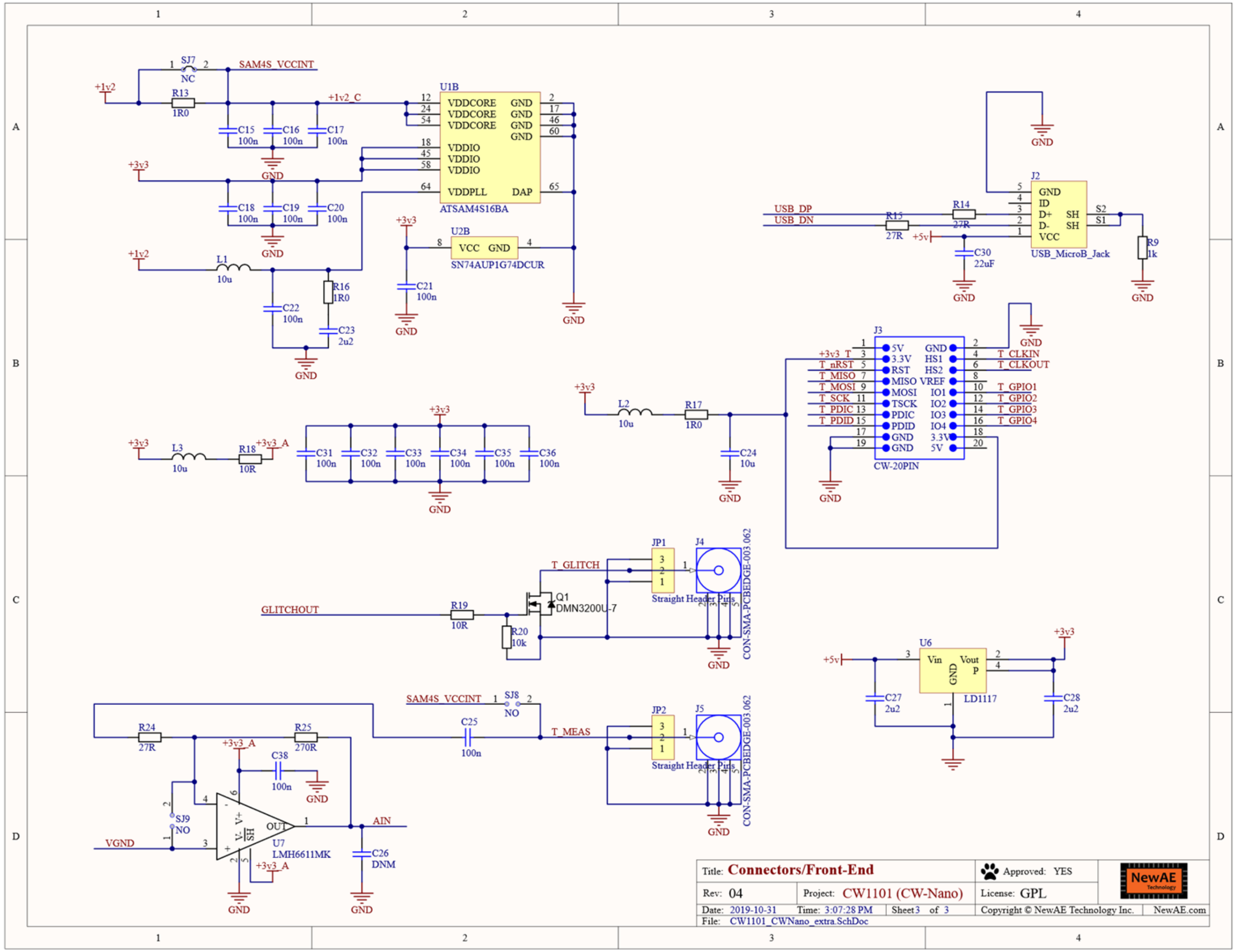
VIII. Appendix

Appendix A: ChipWhisperer-Nano Schematic (Main)



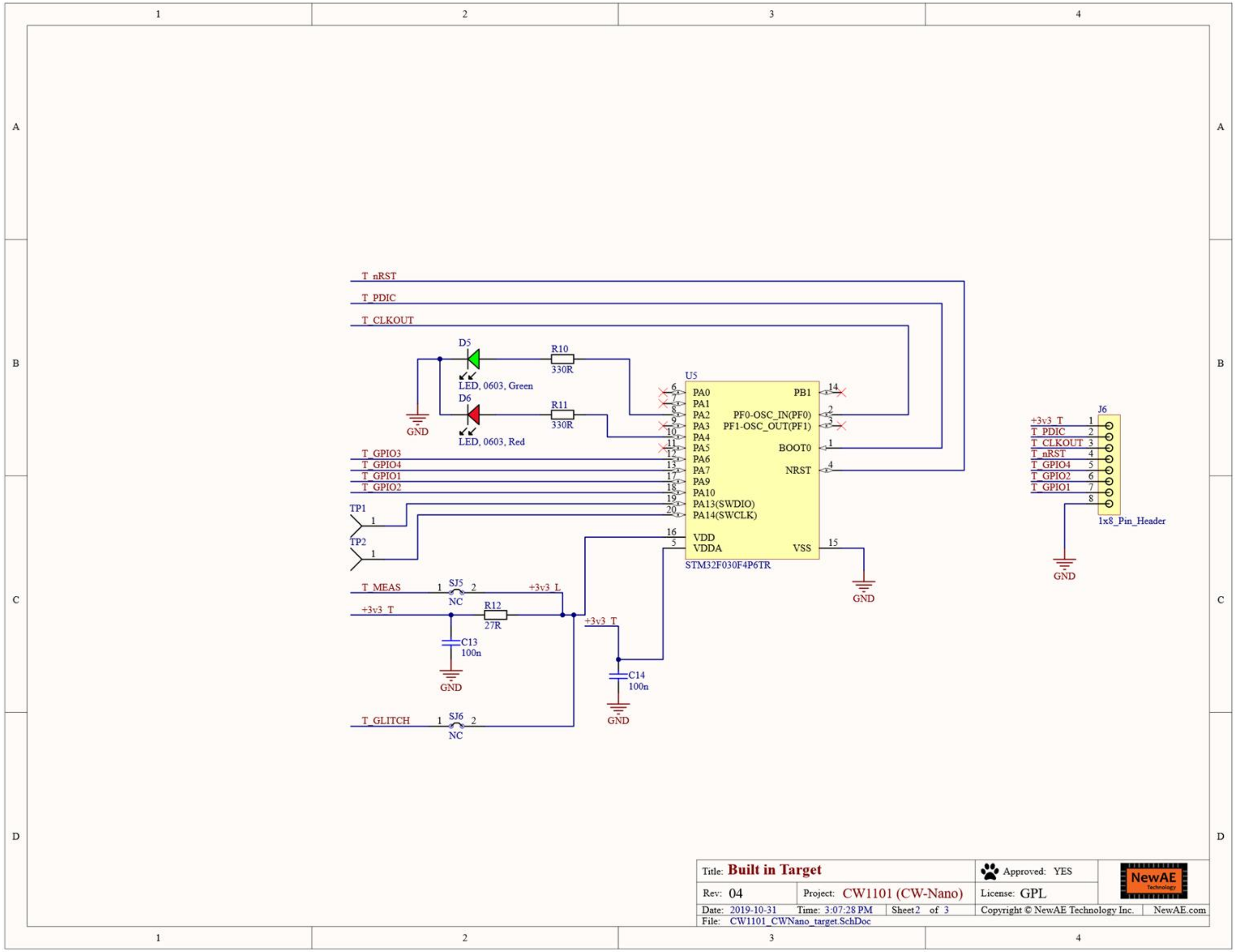
[3]

Appendix A: ChipWhisperer-Nano Schematic (Main)



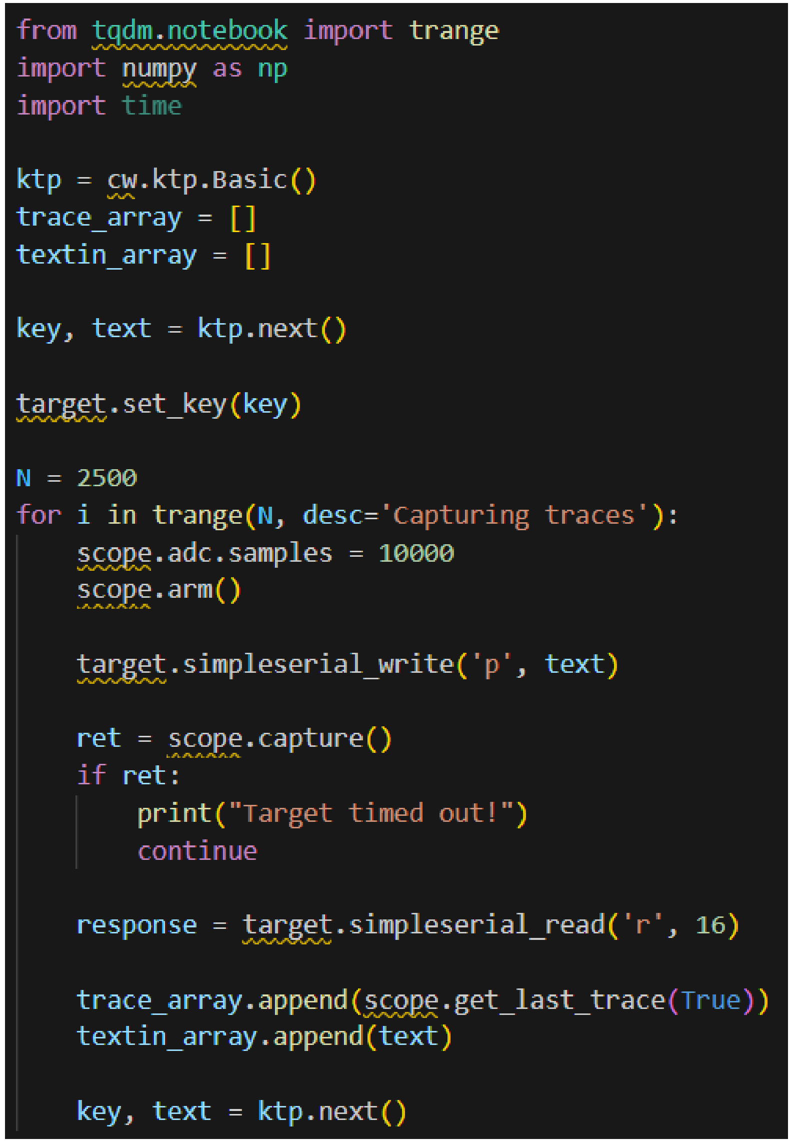
[3]

Appendix A: ChipWhisperer-Nano Schematic (Target)



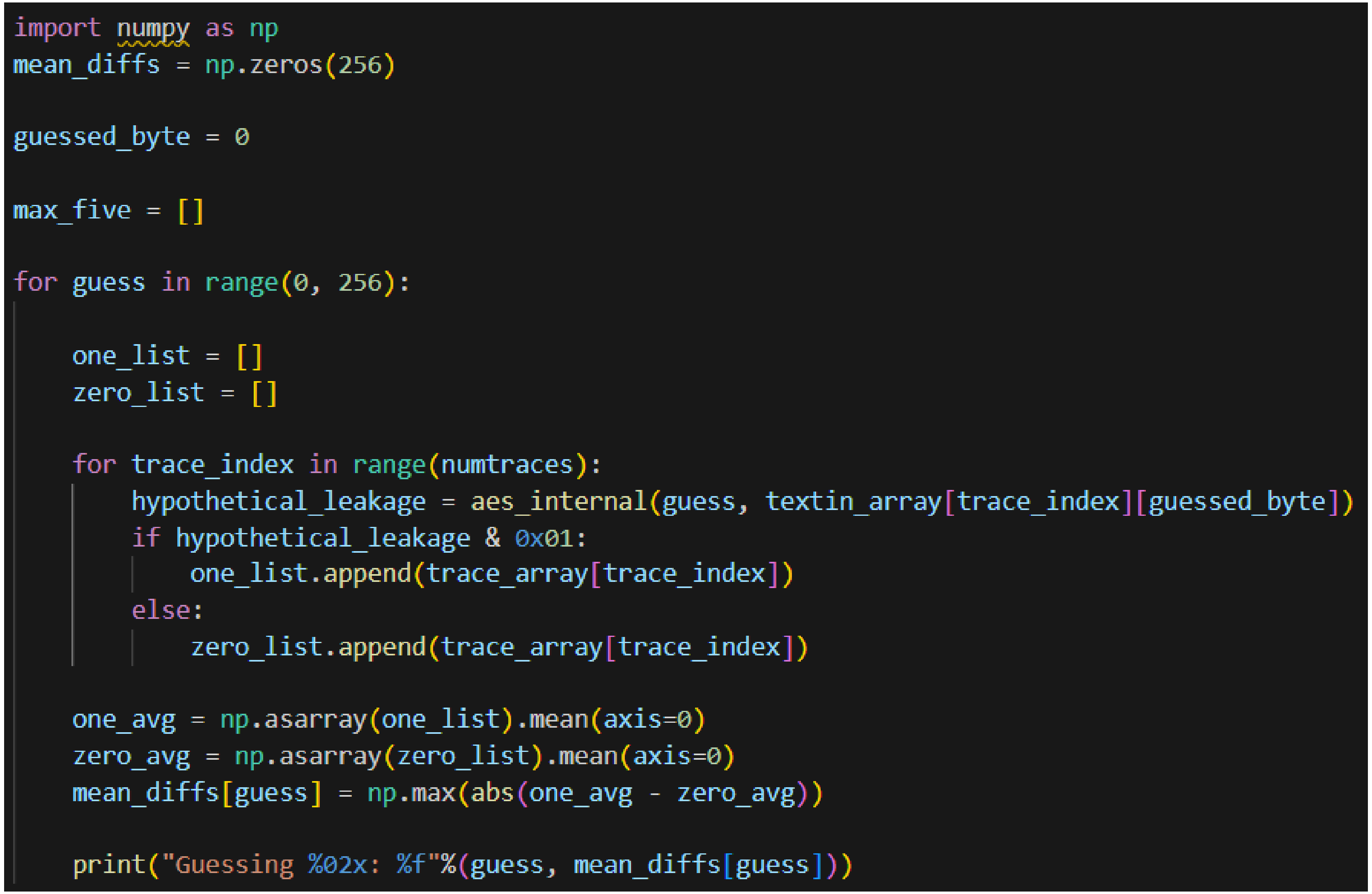
[3]

Appendix 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()
```

Appendix C: Experiment - Code for Analyzing Power Traces



```python
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]))
```