Side Channel Analysis of ASCON-128: Winner of NIST LWC Challenge

Adeel Nasrullah, Arjun Suresh, Ebenezer Asabre and Kwabena Gyasi Bawuah

University of Massachusetts Amherst, Amherst, USA
{anasrullah,arjunsuresh,easabre,kbawuah}@umass.edu
Link to Project Demo

Abstract. With the advent of IoT, a huge number of edge devices share data across the internet. These edge devices are typically resource constrained and need to operate on lower power than traditional devices. Within this context, lightweight ciphers are extremely essential to provide encryption capabilities to such devices without significantly draining power. Given that such devices are deployed on the network edge, an adversary could have physical access over it, which opens up new attack vectors that engineers need to consider. We look at one such attack vector, side channel attacks, and specifically power side channel attacks. This project presents a side channel assessment of lightweight cryptographic ciphers in the NIST lightweight crypto standard finalists, focusing on the ASCON family of ciphers. In line with the threat model, we deploy the ASCON cipher on a hardware Field Programmable Gate Array (FPGA) target, collect power traces and attempt to recover the secret key using attacks such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA). We use the ChipWhisperer platform to collect and analyse the power traces for side channel leakage. The results of the analysis provide insights into the susceptibility of the ASCON cipher to side-channel attacks and highlight the importance of implementing countermeasures to mitigate the risk of side-channel leakage. This work contributes to the ongoing efforts to standardize lightweight cryptography and to ensure the security of cryptographic systems in the face of side-channel attacks.

Keywords: Side Channel Analysis · Lightweight Cryptography · ASCON

1 Introduction

1.1 Motivation

In today's interconnected world, where resource-constrained devices and environments are increasingly prevalent, ensuring data security remains a paramount concern. Hence, any attempt at implementing security on embedded devices has to ensure a delicate balance between robust cryptographic protection and efficient resource utilization. The motivation for conducting a side-channel analysis of ASCON-128, the winner of the NIST Lightweight Cryptography (LWC) Challenge, stems from the critical need to evaluate the security of cryptographic algorithms against various attack vectors. Reasons for side-channel analysis:

- Side channel attacks can be highly effective in breaking the security of a cryptographic system, even if the underlying cipher is theoretically secure. By analyzing side-channel leakages such as power consumption, electromagnetic radiation, or timing information, we can gain information about the secret key or input plain text used in the encryption process.

- Lightweight ciphers are often used in resource-constrained devices, which may be vulnerable to side-channel attacks due to their limited processing power and memory. Therefore, it is important to evaluate the security of these ciphers against side-channel attacks to ensure that the devices are adequately protected against real-world threats
- Side channel analysis can also provide valuable insights into the implementation of a cryptographic system. By analyzing side channel leakage, we can identify vulnerabilities in the implementation, such as unintended information leakage due to algorithmic choices or implementation errors. This information can be used to improve the security of the cryptographic system and to develop more robust countermeasures against side-channel attacks.

1.2 Background

Lightweight ciphers are cryptographic algorithms that have been fine-tuned to address the unique challenges encountered in resource-constrained computing environments, specifically targeting embedded processors and devices. These specialized ciphers are designed with a keen focus on optimizing security while accommodating the limitations imposed by limited computational power, energy efficiency, code size, and memory requirements.. [KA21, BLA17]. Use cases for lightweight ciphers are many, but some of the most common ones include, Internet of Things (IoT) devices, Smart cards, RFID tags and Wireless sensor networks. [A.M21] By effectively minimizing power consumption, reducing code size, and optimizing memory utilization, lightweight ciphers offer promising solutions for ensuring secure data encryption and decryption. Their applicability in resource-limited scenarios makes them invaluable tools for safeguarding the confidentiality and integrity of sensitive information, even in the presence of severe hardware constraints.

1.3 Problem Statement

The security of cryptographic systems is paramount in guaranteeing the confidentiality and integrity of sensitive data. Nonetheless, these systems are susceptible to side-channel attacks, in which attackers leverage unintended information leakages through channels such as power consumption, electromagnetic radiation, or timing measurements. The problem statement of this research is to investigate the vulnerability of ASCON-128, the winner of the NIST Lightweight Cryptography (LWC) Challenge, to side-channel attacks.

1.4 Approach

The goal of this project is to perform an assessment of side channel leakage in the ASCON cipher to check the strength of the cipher under adversarial conditions. As a first step, we did some background research in order to conduct an in-depth study of ASCON-128, its design principles, and the cryptographic techniques employed. We accessed the attack scenarios, such as simple power analysis (SPA), differential power analysis (DPA), or template attacks, depending on the resources and capabilities available. Next, we set up the ChipWhisperer boards and work on deploying an HDL implementation of ASCON to the ChipWhisperer target board. We will then use the setup for data acquisition to capture side-channel traces (information). Apply advanced side channel analysis techniques to the collected data, depending on the attack models defined in order to identify possible sources of leakage.

1.5 Expected Results

The side-channel analysis of ASCON-128 is expected to yield valuable insights into the algorithm's vulnerability to different side-channel attack scenarios. The anticipated results include the following:

- The Number of bits of the ASCON secret key that we can recover
- The Number of traces required to recover the secret key
- Success rate of secret key recovery using DPA

Secondary Expectations: We expect to know to be able to recommend countermeasures that can be used to mitigate these vulnerabilities and gain insights that may enable other kinds of attacks on ASCON e.g. template attacks.

1.6 Connection with the rest of the course

This paper has a strong connection with the four class topics covered in Security Engineering, namely Power Analysis, Template Attacks, Microarchitectural Leakage, and Cross-VM Covert and Side-Channel Attacks in Cloud FPGAs. Our project was simply built on the side-channel analysis techniques we learnt. The Power analysis with ML optional assignment in topic 10 was also used to get some practical understanding of the lessons.

1.7 NIST LWC Challenge

The National Institute of Standards and Technology (NIST) brought out a process to standardize lightweight cryptographic algorithms to be deployed in constrained environments [NIS18]. After multiple rounds of review and comments from the public forum, the ASCON family of ciphers was selected as the new lightweight cryptography standard on Feb 3, 2023.

1.8 Organization of the rest of the paper

This paper is organized as follows: SECTION 2 provides a comprehensive literature review, aiming to highlight the contributions and limitations of previous studies. By examining the existing body of knowledge, we gain valuable insights to enhance our system. SECTION 3 outlines the system methodology by describing the deployment of the ASCON ciphers unto the chipwhisperer and measuring traces by performing DPA. SECTION 4 elaborates upon the design and hardware setup. A breakdown of how we partitioned the work as well as a schedule we followed to achieve our deliverable can be seen in SECTION 6. SECTION 5 discusses what is Correlational power attack and how we attempted to perform it. A discussion of the results as well will be done in this section. For SECTION 7, we will reflect on what we learnt, and what we would have done with more time. We will also discuss additional skills or expertise that would have helped us with this project. Finally, in SECTION 8, we discuss the implications for the security of ASCON-128, and the broader implications for lightweight cryptography. We draw conclusions regarding the effectiveness of ASCON-128 in real-world scenarios, analyze the limitations of the analysis and suggest future research directions to address any remaining vulnerabilities.

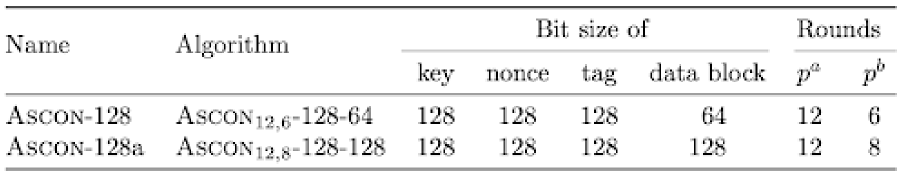
2 Literature Review and Related Work

Our literature review consists of three parts: i) understanding how the cryptographic algorithm under analysis, ASCON, works, ii) reading the literature that has explored power analysis of ASCON, and iii) getting familiar with our tools for data collection. We present the literature we reviewed for our project in the same order.

2.1 ASCON: The Lightweight Cryptographic Cipher

ASCON is a symmetric key and authenticated encryption algorithm. It uses the same key for encryption and decryption, which puts it into the category of symmetric key algorithms. The other category is asymmetric encryption algorithms, such as RSA and SHA, which use separate keys for encryption and decryption. Authenticated encryption means that it not only encrypts the plaintext to generate cyphertext, but also generates a tag for encrypted plaintext. This tag is required for decryption of the cyphertext. This prevents an adversary from mounting replay attacks, where the attacker can record the encrypted data sent by the sender and replay them in the hopes that the receiver will accept these data as valid. With a tag associated with the cyphertext, the receiver can authenticate that the data was sent by the client, since the attacker will not have access to the tag.

ASCON performs encyption and decryption functions in four stages: i) Initialization, ii) Associated Data, iii) Encryption/Decryption, and iv) Finalization. Furthermore, the algorithm provides tunable parameters that may be adjusted to achieve the desired level of security. These parameters include key length k, encryption block size r, internal permutation round number b and initialization round number a. While these parameters can be tuned, there are two standard versions that have been adopted, which are given in the figure 1. Moving on we will only focus on the encryption function, as it is the part we analyze and exploit to recover the secret key.



Name	Algorithm	Bit size of				Rounds	
		key	nonce	tag	data block	p^a	p^b
Ascon-128	$\text{Ascon}_{12,6}$-128-64	128	128	128	64	12	6
Ascon-128a	$\text{Ascon}_{12,8}$-128-128	128	128	128	128	12	8

Figure 1: Parameter configurations for ASCON [DEMS16]

As mentioned above ASCON performs encryption over four stages (shown in figure 2). The first stage is initialization which initializes the 320-bit state comprising five 64-bit registers. First two registers are initialized with a 128-bit secret key, a 128-bit nonce fills the next two registers while the final register is initialized by concatenating encryption rate r (8-bit), external round number a (8-bit), internal round number b (8-bit) and zeros (40 bits). This is followed by permutation rounds and xor operations on the initial state values. it is important to note here that the computations of this round are dependent on secret key which we will consider unknown and the nonce which is randomly generated number which may or may not be fixed across encryption sessions. If the nonce is fixed, the same computations will be performed in each initialization round, and no input dependent power fluctuations would be observed across encryption sessions for this phase. Hence, we ignore this phase for the purpose of power analysis. The next stage is named associated data and as the name suggests it processes associated data using permutations and xor operations. Here, again associated data input may be fixed from the attacker's perspective and we will see no input dependent computations. However, the encryption stage computations does depend on the input plaintext, which is the likely interface made available by an encryption hardware. This part is a potential target of differential power analysis.

2.2 Power Side Channel Attacks

The experiment is done using VHDL code which runs chip-whisperer board. In the Ascon encryption scheme, the internal state is composed of five 64-bit registers [SP16]. During initialization, we have access to the state's initial values, excluding the key portion.

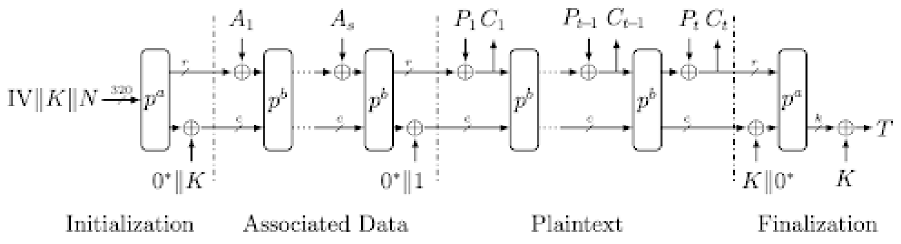


Figure 2: ASCON Encryption Phases [DEMS16]

Additionally, we have the flexibility to vary the nonce used in each encryption run. Leveraging this knowledge and control, we focus our attack on the end of the first round. By understanding the state at that particular stage and manipulating the inputs, we can exploit potential vulnerabilities or weaknesses in the encryption algorithm. This targeted approach allows us to concentrate our efforts on a specific point in the encryption process, as shown in figure 2, to increase the chances of a successful attack. In this attack, we employ the Hamming distance model [SD17].To carry out the attack, it is necessary to compute intermediate values for each nonce and different key guess.

$$\Sigma_0(x_0) = x_0 \oplus (x_0 \gg 19) \oplus (x_0 \gg 28) \tag{1}$$

The non-linear S-box output for x_0 can be represented as follows:

$$y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \tag{2}$$

Despite having knowledge of the register contents prior to the first round, we still include the term a_0 in the equation in accordance with the Hamming distance model as in Section 3.

$$y_0 = x_1(x_4 + x_2 + x_0 + 1) + x_3 + x_2 + x_0 + a_0 \tag{3}$$

After the first round, the equation now determines the register's activity. Examining the state initialization, we observe that x_1 and x_2 represent constant key bits, while x_3 and x_4 correspond to variable bits of the nonce, and x_0 represents a constant bit of the IV. Any bits that contribute a constant value to the register's activity can be eliminated from the equation. As a result of doing so, we obtain:

$$y_0 = x_1(x_4 + 1) + x_3 \tag{4}$$

Thus, the resulting intermediate value becomes reliant on a single bit from a key register and two bits from nonce registers, reducing the dependency to these specific bits.

Consequently, the intermediate value now relies exclusively on a single bit from a key register and two bits from nonce registers, limiting its dependence to these specific bits. If we combine equations (1) and (5) we get the following selection function.

$$S_i(N, K^*) = \kappa_0^*(\nu_{i+64} + 1) + \nu_i + \kappa_1^*(\nu_{i+109} + 1) + \nu_{i+45} + \kappa_2^*(\nu_{i+100} + 1) + \nu_{i+36} \tag{5}$$

In this scenario, N represents the 128-bit nonce, and κ_i^* denotes a bit from a key guess. Since the selection function involves three key bits, there are a total of eight possible key guesses.

Due to the substantial number of unknown key bits, it is impractical to enumerate all possibilities. Therefore, it becomes necessary to also attack the least significant half of the key. Notably, in the S-box, only the register x1 yields a result with a quadratic term that includes x2, which represents the least significant half of the key before the first round.

Restarting the process from the linear step:

$$\sum_{1}(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \tag{6}$$

The resulting value from the S-box for x_1 is:

$$y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \tag{7}$$
$$y_1 = x_3(x_2 + x_1 + 1) + x_4 + x_2x_1 + x_2 + x_1 + x_0 \tag{8}$$
$$y_1 = x_3(x_2 + x_1 + 1) + x_4 \tag{9}$$
$$y_1 = x_3(x_{12} + 1) + x_4 \tag{10}$$

As the number of traces increases, the attack on register x_1 demonstrates a success rate close to 1 at fifty thousand traces. After analyzing approximately fifty thousand traces, it becomes feasible to extract the XOR (x_{12}) of key-relevant bits from registers x_1 and x_2. These bits contain crucial information about the key used in the cryptographic function prior to the first round. Since the first attack resulted in the key bits stored in x1 this can be XOR'ed with x12 to obtain x2. With high success rate it is possible to obtain the key from ASCON using DPA.

Examining the remaining output registers of the S-box, we can analyze their characteristics and behavior.

$$y_2 = x_4x_3 + x_4 + x_2 + 1 \tag{11}$$
$$y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \tag{12}$$
$$y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \tag{13}$$

Upon examination of the equations, we can observe that y_2 and y_3 do not contain any non-linear terms involving both a key and a nonce register. As a result, these registers are not susceptible to attacks. However, in the case of register y_4, it does feature a non-linear term that involves both a key and a nonce register. Consequently, this register can be targeted and exploited through appropriate attack techniques.The expression can be reformulated in a similar manner [SP16].

$$y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \tag{14}$$
$$y_4 = x_4(x_1 + 1) + x_3 + x_1x_0 + x_1 \tag{15}$$
$$y_4 = x_4(x_1 + 1) + x_3 \tag{16}$$

Given that the entire key was successfully obtained by attacking registers y_0 and y_1, register y_4 was not targeted in this instance. However, it may be of interest to investigate register y_4 in the future. This is because the leakage characteristics of registers y_0 and y_1 may not be identical, suggesting the possibility of variations in the leakage behavior of register y_4 as well. Exploring this aspect could offer valuable insights into the leakage patterns and further enhance our understanding of the cryptographic system under analysis.

2.3 Side Channel Analysis

Side-channel analysis, utilized by attackers, takes advantage of the information leakage resulting from cryptographic functions performed by a processor, which is dependent on various factors [SP16]. This paper specifically concentrates on exploring power consumption, which is one of the side channels capable of exposing information. Other potential side channels include time and electromagnetic fields. In order to quantify the power consumption of a device, a circuit is constructed with a small resistor integrated in series and connected to the ground. Measuring the voltage difference across the resistor enables the determination of the power consumption. Oscilloscopes operating at high sampling rates can accurately sample this voltage difference.

2.4 Power Analysis

First, we took a look at [KJJR11] from class to formalize our understanding of Differential Power Analysis and look into [SP16, SD17, LWL$^+$22, Rama] which are existing works on side-channel attacks on ASCON. Finally, we also look into [Ramb] which is a side channel report on many of the NIST lightweight cryptographic ciphers. Together, these references would help us evaluate the side channel resilience of ASCON and give us guidelines on planning an evaluation with our setup.

[EB] looks into the math of CPA with the aim of improving the Hamming weight model, and enhancing the DPA itself. The paper [DPA13], discusses the design and evaluation of the Advanced Encryption Standard (AES) algorithm with a focus on its resistance against side-channel attacks. The implementation is protected against univariate side-channel attacks by a masking scheme called “Rotating Sbox Masking” (RSM). RSM involves randomizing intermediate values during the encryption process to make it harder for attackers to extract sensitive information from power consumption measurements. It goes into detail on how the masking protection is an additive Boolean masked scheme, with statically masked sboxes. The resource [OL16] was then used to get an overall picture of how to perform DPA on AES-128 SBOX using DPA AND CPA. We mainly focused on how the CPA attack is implemented by building a power model of the device using the Hamming Weight Power Model method.

3 Methodology

The methodology outlines the approach and procedures we followed to conduct the side channel analysis of ASCON-128. The required steps to be taken are to collect data, perform the analysis, and evaluate the vulnerability of the algorithm to side-channel attacks. The methodology was divided into the following key steps:

- Understand the ASCON permutation: This involved examining the structure of the ASCON permutation in detail to understand the sequence of operations performed in each round, including substitution, permutation, and bitwise operations. We paid specific attention to the non-linear operations employed in the ASCON permutation in order to analyze the potential non-linear side channel leakage

- Experimental Setup: Setup the ChipWhisperer (CW) capture and target boards, Synthesize an HDL implementation of ASCON cipher and generate a bitstream to program the FPGA.

- Data Acquisition: We will encrypt using ASCON while simultaneously collecting side-channel power traces. We will gather an adequate number of power traces to achieve statistical significance.

- Preprocessing: Preprocess the acquired power traces to remove any noise or artifacts that may affect the analysis. Apply signal processing techniques such as filtering, normalization, and alignment to enhance the quality and consistency of the power traces.

- Perform DPA on obtained traces to recover key bits: Conduct a statistical analysis of the power consumption traces to identify patterns, correlations, and potential leakage sources. We will use the correlation analysis technique to extract meaningful information from the power traces.
- Confirm success of key recovery: Assess the vulnerability of the cipher based on the identified leakage sources and statistical analysis results. Quantify the amount of leaked information and evaluate the effectiveness of the attacks.

3.1 Block Diagram

Figure 3 presents a high level view of our experimental setup. It has two basic components: i) hardware setup for data collection and ii) statistical analysis of the data in software which aims to extract secret key contained in the hardware. The hardware setup consists of a scope device (chipwhisprer CW1200) which is equipped with analog-to-digital converter (ADC) module used to collect power consumption data. The other component is the target device (CW05 FPGA) which has a secret key and it implements ASCON encryption engine. The ADC module is driven by the external clock which is fed from the target device. This allows synchronized operation of ADC module and FPGA, and ensures that ADC module, by tapping on target's input power lines (3.3V and Ground), can obtain one or more samples per second. To collect a power trace, we provide a key and plaintext input to the target device which immediately starts the encryption process and in parallel ADC module collects power consumption samples. The ADC is triggered as soon as the target device starts it operation which is aided by the fact that ADC module works using target's clock. This process is automated using Chipwhisperer software [Tec] APIs which allows the collection of thousands of power traces within few minutes.

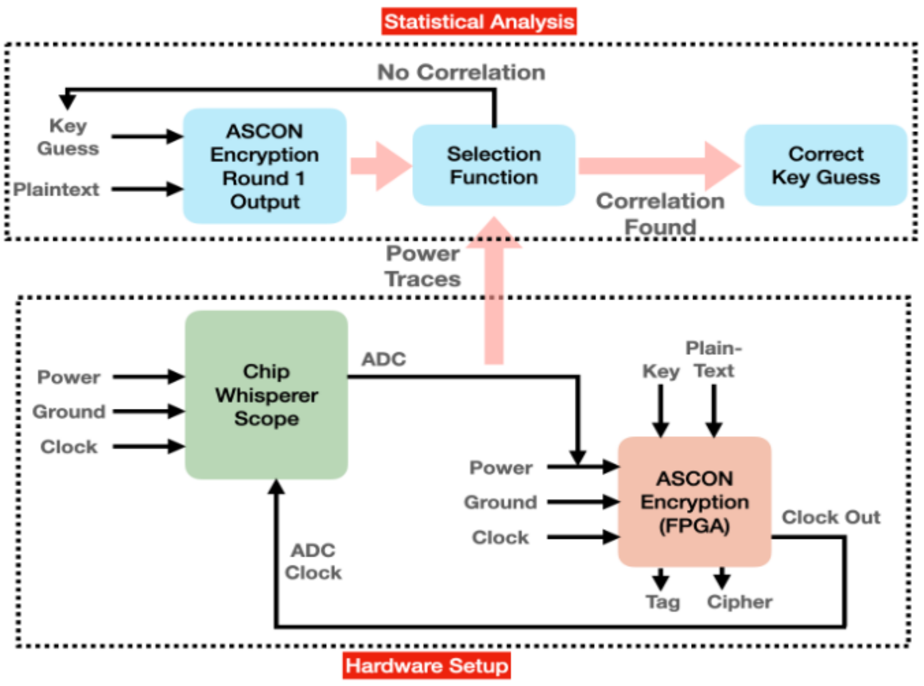


Figure 3: Block Diagram

The software part of our system comprises of one major part, that is, the selection function which is the backbone of our analysis. Selection function is based on a specific part of the encryption algorithm which performs plaintext dependent computations e.g. first permutation round followed by XOR operations in the ASCON. The power traces are divided into two groups based on the selection function output. If the two groups show a significant difference in their power consumption, it shows the corresponding assumptions about the key are correct. Usually, a part of the secret key is revealed at a time, so the process is repeated until the complete key is known.

3.2 Differential Power Analysis Testing

In order to test DPA, resources from the DPA contest (v4) was used. It had a set of 100,000 traces that used a Masked version of AES as shown in [DPAb, DPA13]. This resulted in us using Mathlab to run the AES instrumentation of the cipher and use the use Neural Network Pattern Recognition app to draw a neural network to make predictions. The ASCON repository [Eic23] was also referred to in order to build the leakage models.

3.3 ChipWhisperer

ChipWhisperer is an open-source toolchain developed by NewAE Technology Inc. dedicated to performing and analyzing side channel and fault analysis resistance of hardware designs. The toolchain comprises three main parts:

- Hardware - Dedicated target (on which the target design is deployed) and capture (which records side channel traces from the target) boards
- Firmware - The capture boards have their own firmware and there are various different firmware images for the different hardware target boards
- Software - A Python API for communicating with the ChipWhisperer hardware and processing side channel traces obtained from the capture boards

The ASCON cipher is deployed on the ChipWhisperer target board and a capture board is used to record power traces for various encryptions with different inputs. The chipwhisperer [chi23] API in conjunction with Jupiter Notebook as an IDE is used to analyze the recorded traces, perform DPA and recover the secret key bits.

4 Implementation

The system was implemented with a ChipWhisperer Pro CW1200 capture board on a CW305 target board, which has an Artix A100 FPGA target. A synthesizable hardware description, or RTL (Register Transfer Level), of the ASCON-128 cipher was synthesized by Vivado and deployed on the CW305 target board using the ChipWhisperer Python API.

4.1 Synthesizing the ASCON RTL

Synthesis is the process of converting high level hardware description into logic gates. Xilinx Vivado was used to perform synthesis on the ASCON RTL and generate a bitstream, which was used to program the FPGA. The top level hardware description and constraint files were provided by NewAE Technology Inc. for their CW305 target board, and the ASCON implementation was added to their top module in order to enable correct programming of the target and recording of power traces from encrytion cycles. The ASCON implementation analyzed was developed by Micheal Fivez, as part of their Master thesis at KU Leuven, Belgium [Fiv16].

4.2 ChipWhisperer Hardware Setup

The CW305 target and ChipWhisperer Pro CW1200 capture boards were set up as detailed in the whitepaper by NewAE Technology Inc. [DTO20].
The CW305 board was set up with the following configuration:

- S1 Switch - M2, M1 and M0 switches were all set to high
- S2 Switch (for PLL) - J16 was set to low, and K16, K15 and L14 were set to high
- SW5 - FPGA power was set to "Auto"
- VCC-INT was selected for the on-board switch-mode power supply for the VCC-INT supply
- Input power was selected as 5V DC supply

The CW1200 and CW305 were connected together by the following wires:

- 20 pin target connector on CW1200 to J1 on CW305
- SMA cable from "Measure" SMA on CW1200 to X5 (amplified shunt output) on CW305

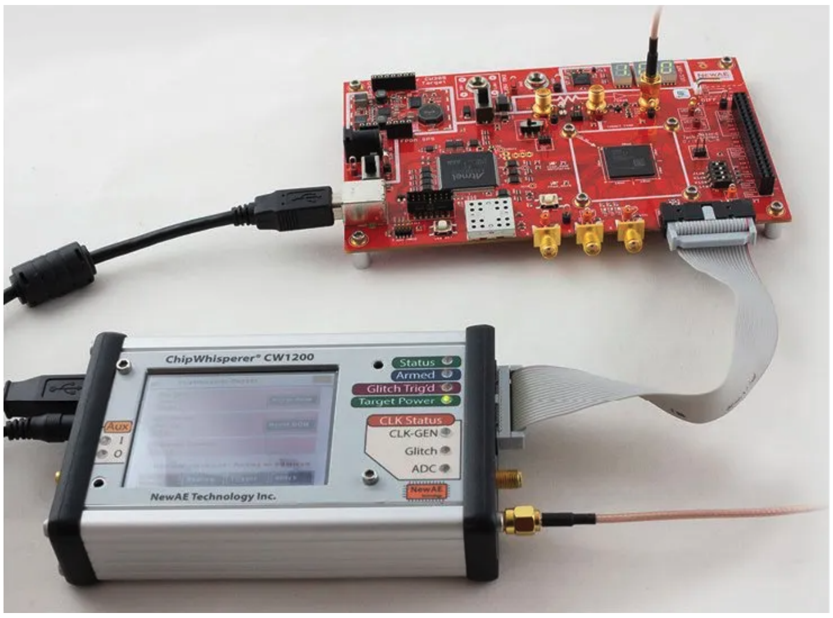


Figure 4: ChipWhisperer CW305 Target Board Setup with ChipWhisperer Pro CW1200 Capture Board [DTO20]

4.3 Collection of Traces

Once setup, we communicate with both of our boards (scope and target) using python APIs that are part of the chipwhisperer python library [Tec]. We first configure the scope (CW1200) to capture 200 samples for each power trace with a gain of 25dB. The gain values represent the amplification factor for the signal being sampled, which is the voltage level in this case. The target board is then programmed to run the ASCON-128 encryption engine on its ARTIX A100 FPGA. We then configure the target to run at

10MHz. Finally, the ADC in the scope is configured to operate using the external clock of the target FPGA. The external clock is multiplied by a factor of 4 internally. This locks the ADC to the FPGA clock signal and ensures that 4 samples are captured for each clock cycle of the FPGA. Although we can capture one sample for the FPGA clock cycle, it may not align with the peak power consumption during the cycle and produce incorrect results. Collecting 4 samples for each clock cycle ensures that we can record power consumption accurately. Having configured the two devices, we use the ChipWhisperer API $capture_trace(scope, target, text, key)$ to capture a power trace. This API inputs the text and key to the FPGA board and then triggers the encryption module. At the same time, it starts the scope's ADC to record power trace. For power analysis, we typically need several thousand traces. This is achieved by invoking the above mentioned API repeatedly, and each time we input the same key and, but, a different text. Changing the plaintext would result in different computations each time, which would be captured in the power traces. It is important to note that ascon also takes a nonce and associated data as input. We have fixed both these inputs in the implementation and would rely on changing plaintext to induce variations in power consumption.

5 Experiments and Results

5.1 Collected Traces

Figure 5 shows an example power trace collected for our hardware implementation of the ASCON encryption engine.

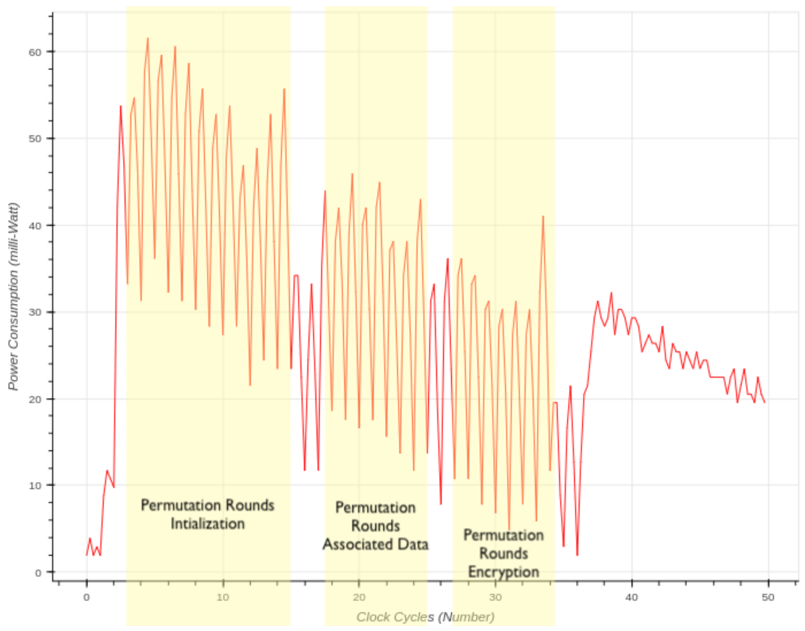


Figure 5: Power Trace for ASCON Execution on FPGA

Each spike in the figure shows one clock cycle of the encryption engine. We clearly see three phases the power consumption trace. First, a series of high power consumption which lasts 12 clock cycles. This corresponds to the initialization phase of the ASCON which executes 12 permutation rounds. During this round, the computations rely on the secret key and nonce inputs which are fixed for our setup. It is followed by a transition phase after which we observe a series of 7 clock cycles which corresponds to 6 permutation

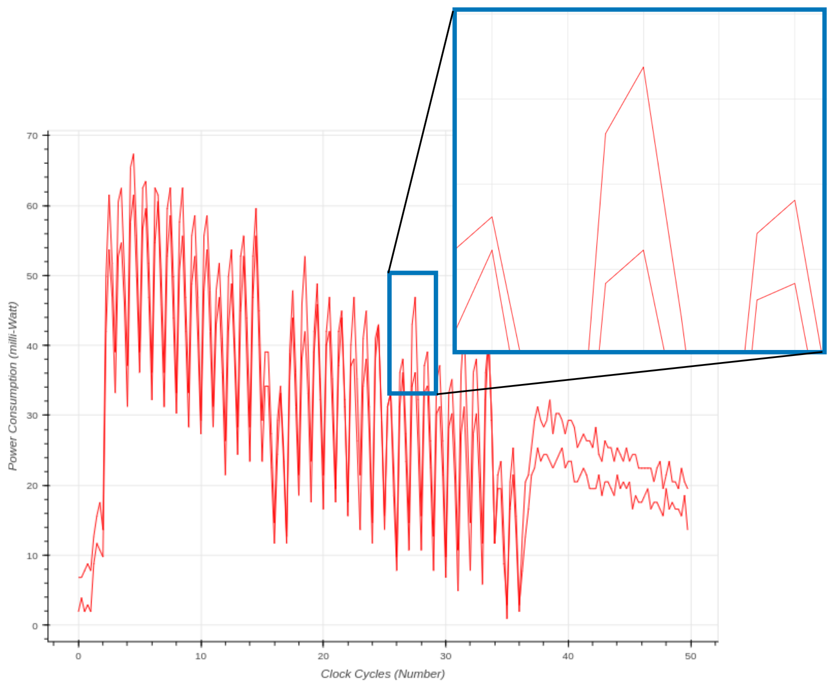


Figure 6: Difference in power consumption of a permutation round for two different plaintexts

rounds of the associated data stage. This is where the associated data, same as plaintext in our implementation, is absorbed into the ASCON's internal state. This is the part where computations rely on the input plaintext and where potential variations in the power consumption may reveal information about the secret key. The final stage also consists of 7 clock cycles corresponding to the encryption stage of the ASCON where the plaintext is input and ciphertext will be the output after 6 rounds of permutations. Figure 6 shows power traces corresponding to two different plaintext inputs. We can see that for most part the two traces closely follow each other, especially, during the initialization phase which is independent of the input plaintext. However, we can see a difference in power consumption for the first permutation round of ASCON's third stage i.e. encryption. This may be the indicative of data dependent computations, so the first permutation round in the encryption stage is a good candidate for us to base our power analysis.

5.1.1 Power analysis with ML

Given the delayed nature of this deliverable, we decided to test our hand on doing some version of DPA ahead of the traces being delivered. Whiles one part of the team was working on the implementation of Ascon on the FPGA, the other side of the team tried to do the option assignment in Topic 10 which had to do with trying power analysis with ML [DPAa]. The aim of this was to experiment with the traces. There was some data processing and then a function to plot the correlation in order to try and identify the leakage points before and after the Sbox. The produced plot can be seen in Figure 7. This is done by measuring the correlation from the traces with the templates created by the InterBytes function and Mask Bytes function with respect to different points of attack. So in other words we could edit the InterBytes function which has the aim of computing the inter values of one specified byte of a state in the AES which sounds more like the selection functions job in [SP16]. This could be edited to change the points of attack to create templates.

5.2 Power Analysis

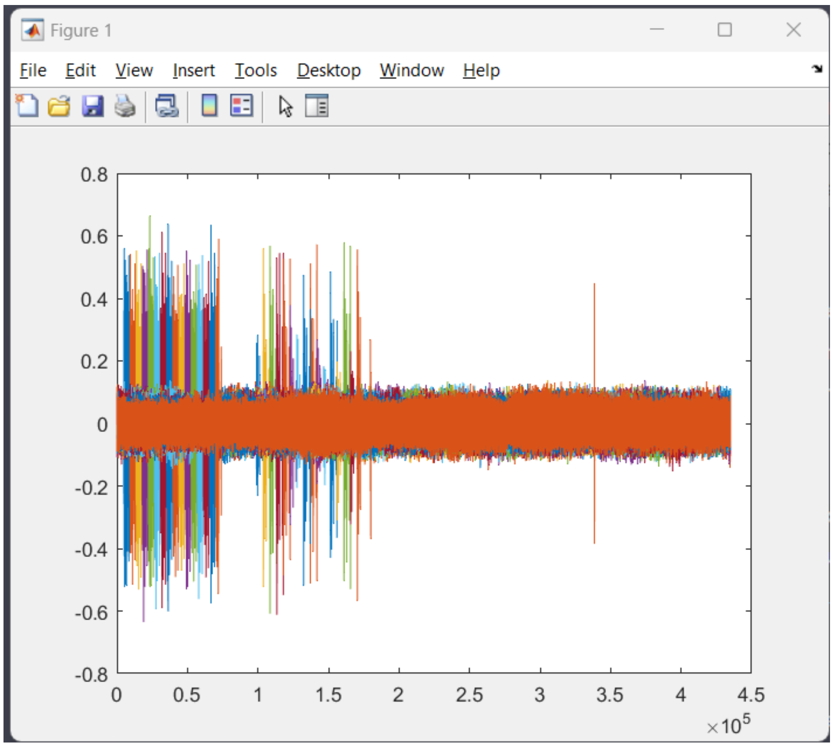


Figure 7: DPA test

Starting with our initial experiment with ML on AES, we run some tests with ML to try and detect the mask. We used the leakage points seen in Figure 7 to then train a neural network to identify the offset for the mask used. We created the training set and ensure that the neural network is working. In this case, achieving a validation performance of 2.6389xe05 at epoch 33 which can be seen in Figure 8 suggests that the neural network is performing very well on the validation dataset, with a very low error rate or loss. This can be considered a positive outcome and indicates that the network has learned to make accurate predictions on unseen data by epoch 33. The aim of this test was to see how we could easily identify the mask offset if we eventually complete the ASCON implementation and we decided to carry out a power analysis of a masked ASCON referred to in [LB22].

5.2.1 Correlation Power Analysis of AES

Due to a delay in getting the traces for ASCON, we decided to look into breaking the first-order implementation of AES with DPA giving AES is somewhat similar to ASCON in the sense that it also uses S-boxes [LB22]. So we started building the ASCON attack on the AES attack resources found. A summary of how AES works can be referred to in Figure 9 and can be read up in [OL16].

In order to attack the system using CPA, we concentrate on the state after the add round key and the sub bytes state of the first round. The CPA setup will involve encrypting known plaintext a number of times with the same key whiles measuring the power consumption during encryption. Given we were using a hardware implementation and expected a lot of

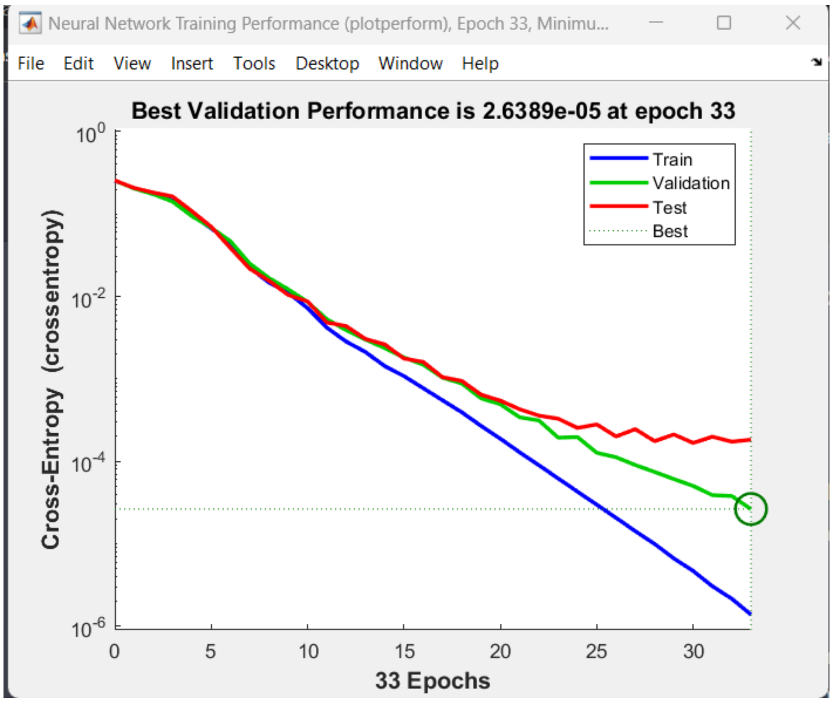


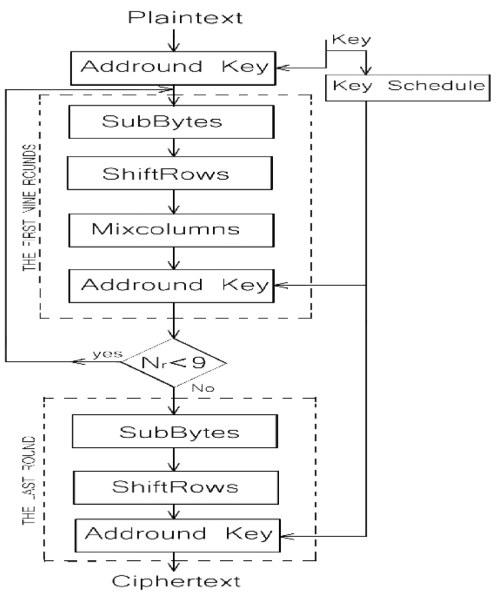
Figure 8: Graph to show validation performance of 2.6389e-05 at epoch 33

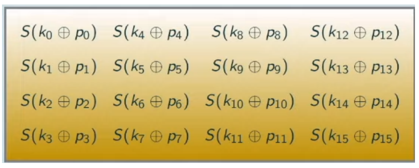
noise, the more traces we had to collect. The state of the matrix at the attack point can be seen in Figure 10 for plaintext bytes $\{p_0, p_1, ...p_{15}\}$ and round key bytes $\{k_0, k_1, ...k_{15}\}$. Just like ASCON, we will aim for the first byte by exhaustively searching all possible values and correlating that with the power traces. Our assumption will follow the Hamming weight theory that the power consumption will correlate with the number of bits sets. In summary, the steps to find the correlation is:

- Guess that first round key byte $k_0 == 0x00$
- Compute Hamming weight of $S(p_0 \bigoplus k_0)$ for the plaintext blocks
- Compute correlation (r_{00}) between the power consumption and the values of $HW(S(p_0 \bigoplus k_0))$.
- Repeat the process, guessing that $k_0 = 0x01, 0x02, ..., 0xFF$, and computing corresponding correlations $r_{o1}, r_{o2}, ...r_{ff}$.
- Pick the maximum of the absolute of the correlation r_i, which corresponds to $k_0 = i$.
- Find the rest of the bytes in the round key (input key) $k_1, k_2, ..., k_{15}$.
- Deduce the secret key from the round key.

Ideally, the key should be found after 4096 guesses.

To test this, we used the Chipwhisperer API [chi23] and created some demo traces. We used Python code and it is run in a Jupyter Notebook. The functions and packages can be looked into from the Chipwhisperer repository [new23]. For this part, we did not concern ourselves with the collection of the traces so as a first step, we loaded the traces and then we get the key. The analyzer includes various leakage models that can be passed to the CPA as the leak model argument. The relevant python script used can be seen in Appendix A. The text in and out as well as the Key guess can be seen in Figure 12. Unfortunately, due to a lack of time, we were not able to explain the low correlation.





$S(k_0 \oplus p_0)$	$S(k_4 \oplus p_4)$	$S(k_8 \oplus p_8)$	$S(k_{12} \oplus p_{12})$
$S(k_1 \oplus p_1)$	$S(k_5 \oplus p_5)$	$S(k_9 \oplus p_9)$	$S(k_{13} \oplus p_{13})$
$S(k_2 \oplus p_2)$	$S(k_6 \oplus p_6)$	$S(k_{10} \oplus p_{10})$	$S(k_{14} \oplus p_{14})$
$S(k_3 \oplus p_3)$	$S(k_7 \oplus p_7)$	$S(k_{11} \oplus p_{11})$	$S(k_{15} \oplus p_{15})$

Figure 11: a) Overview of AES [AES12] b) AES state matrix at attack point

5.2.2 Correlation Power Analysis of ASCON

After obtaining the traces from the ASCON implementation we also attempt to follow the same method used above to perform CPA on the traces. The only expected difference is the fact that the API is currently built for a Hardware implementation of AES so we will have to rewrite the code for the leakage model. We tried to built our leakage model using the hardware implementation. Unfortunately, we were not able to get the working model of ASCON before the due date of the final report. See the implementation code in Appendix B.

6 Partitioning of Work and Schedule

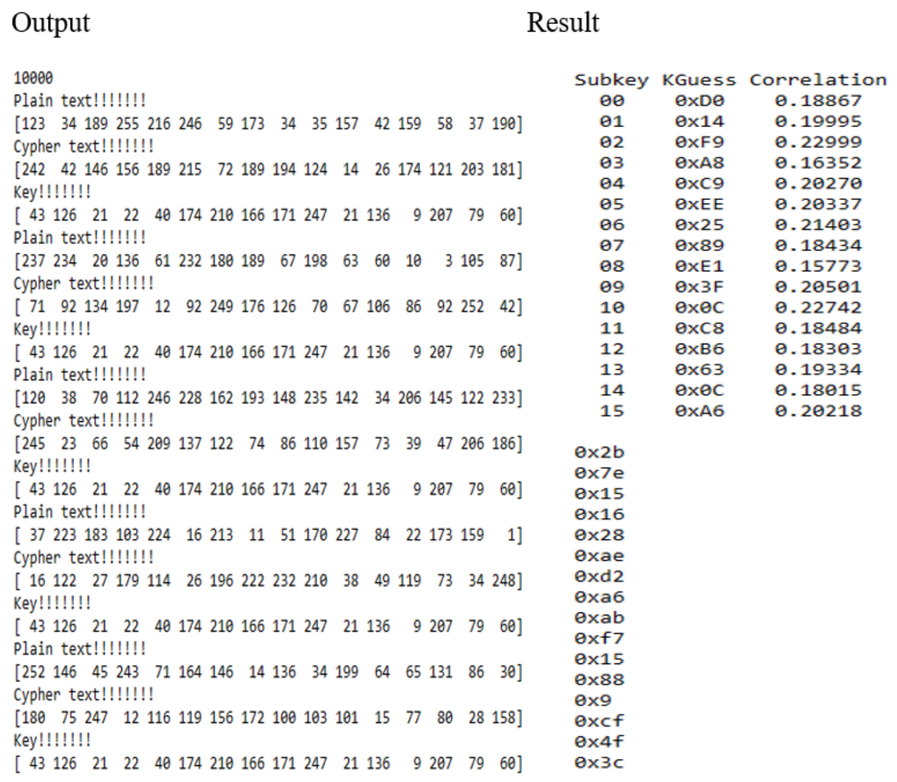
In our pre-proposal alignment, we had the initial plan to partition the work accordingly:

- Literature survey to find relevant papers - all
- ChipWhisperer setup - Arjun and Asabre
- Cipher identification and deployment - Adeel and Asabre
- Data collection - Arjun and Adeel
- Data analysis and deliverables management - Kwabena

When it came time for our first presentation we had lined up our schedule as seen in Figure 13

Some of the potential causes of delay include:

- Validation of collected traces
- Chip-whisperer Hardware debugging issues
- Redundancy in running setup to ensure quality of work



Output

```
10000
Plain text!!!!!!!
[123  34 189 255 216 246  59 173  34  35 157  42 159  58  37 190]
Cypher text!!!!!!!
[242  42 146 156 189 215  72 189 194 124  14  26 174 121 203 181]
Key!!!!!!!
[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
Plain text!!!!!!!
[237 234  20 136  61 232 180 189  67 198  63  60  10   3 105  87]
Cypher text!!!!!!!
[ 71  92 134 197  12  92 249 176 126  70  67 106  86  92 252  42]
Key!!!!!!!
[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
Plain text!!!!!!!
[120  38  70 112 246 228 162 193 148 235 142  34 206 145 122 233]
Cypher text!!!!!!!
[245  23  66  54 209 137 122  74  86 110 157  73  39  47 206 186]
Key!!!!!!!
[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
Plain text!!!!!!!
[ 37 223 183 103 224  16 213  11  51 170 227  84  22 173 159   1]
Cypher text!!!!!!!
[ 16 122  27 179 114  26 196 222 232 210  38  49 119  73  34 248]
Key!!!!!!!
[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
Plain text!!!!!!!
[252 146  45 243  71 164 146  14 136  34 199  64  65 131  86  30]
Cypher text!!!!!!!
[180  75 247  12 116 119 156 172 100 103 101  15  77  80  28 158]
Key!!!!!!!
[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
```

Result

```
Subkey KGuess Correlation
  00    0xD0    0.18867
  01    0x14    0.19995
  02    0xF9    0.22999
  03    0xA8    0.16352
  04    0xC9    0.20270
  05    0xEE    0.20337
  06    0x25    0.21403
  07    0x89    0.18434
  08    0xE1    0.15773
  09    0x3F    0.20501
  10    0x0C    0.22742
  11    0xC8    0.18484
  12    0xB6    0.18303
  13    0x63    0.19334
  14    0x0C    0.18015
  15    0xA6    0.20218

0x2b
0x7e
0x15
0x16
0x28
0xae
0xd2
0xa6
0xab
0xf7
0x15
0x88
0x9
0xcf
0x4f
0x3c
```

Figure 12: Results of CPA on AES

To ensure effective resource allocation, our partition of work fell along the lines of:

- Understand the ASCON permutation - all
- Inner understanding of capture trace function - Arjun, Adeel, Asabre
- Chip-Whisperer (CW) Setup - Arjun, Asabre
- HDL Synthesis and implementation of ASCON cipher- Adeel, Arjun
- Power traces collection and redundancy testing - Adeel, Arjun, Kwabena
- Data analysis - Kwabena, Adeel
- Paper writing and editing - all

How our schedule ended up can be seen in Figure 14

7 Analysis and Reflections

As a team, we gained valuable hands-on experience in setting up the ChipWhisperer system, configuring the hardware targets, and collecting power traces for analysis. Through regular and constructive discussions, we deepened our knowledge of differential power analysis (DPA) and correlation power analysis (CPA) techniques, specifically in the context of

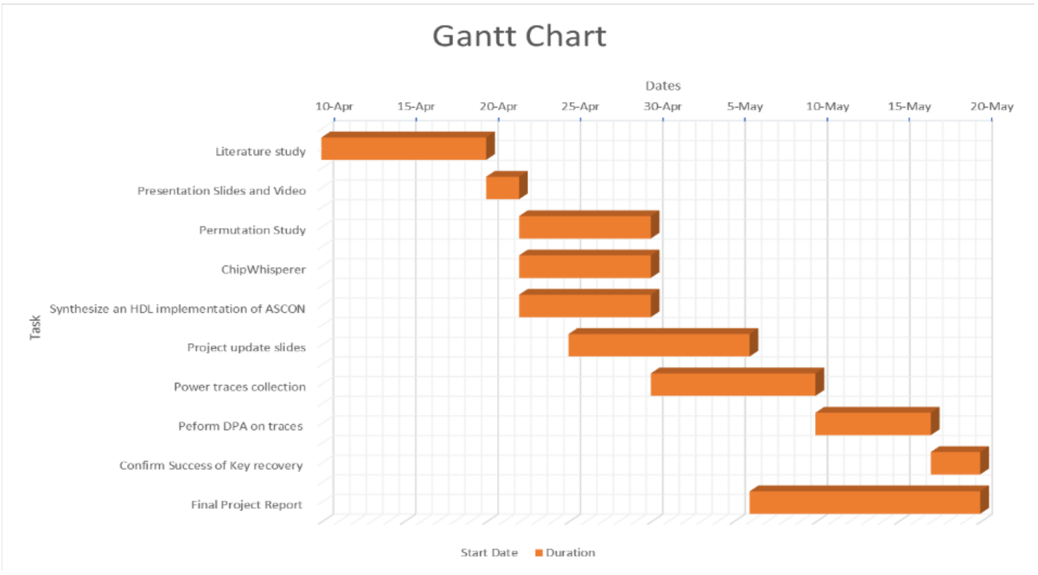


Figure 13: Gantt chart showing our initial schedule to Completion

hardware targets. Working together allowed us to leverage each team member's expertise and insights, resulting in a more comprehensive approach to the project. We collectively explored and implemented best practices for capturing and analyzing side channel power traces, ensuring the reliability and accuracy of our results. The collaborative environment fostered open communication and the sharing of ideas, enabling us to tackle challenges more effectively.

Furthermore, this project emphasized the significance of teamwork, coordination, and effective task allocation. Each team member contributed to different aspects of the project, such as setting up the hardware, programming the targets, and analyzing the collected data. This division of responsibilities facilitated a smooth workflow. By working together, we not only deepened our understanding of side channel analysis and cryptographic systems but also developed valuable skills in project management, communication, and collaboration.

With one more week of work, an opportunity to further enhance the Differential Power Analysis (DPA) testing on the FPGA platform could have been pursued. A comprehensive investigation could have been conducted by running the attack with varying numbers of traces, such as 100, 500, 1000, 2000, 3000, 5000, 7000, and up to 10,000 or more. The success rate could have been plotted against the number of traces required, providing valuable insights into the scalability and effectiveness of the attack. This iterative approach would have allowed for a fine-tuning of the attack parameters and optimization of the power analysis process.

To maximize the limited time available, a focused and efficient approach would have been crucial. Prioritizing the collection of power traces from key areas of interest would have ensured a more representative and targeted data set. The FPGA setup would have been meticulously fine-tuned to enhance synchronization and accuracy, ensuring reliable and consistent results. Additionally, a preliminary assessment of potential countermeasures could have been conducted within the given time frame. This evaluation would have facilitated the identification of immediate mitigation strategies to enhance the security of the cryptographic system against DPA attacks.

The additional week of work would have provided an opportunity to gather a more comprehensive dataset, explore the scalability of the attack, and optimize the experimental

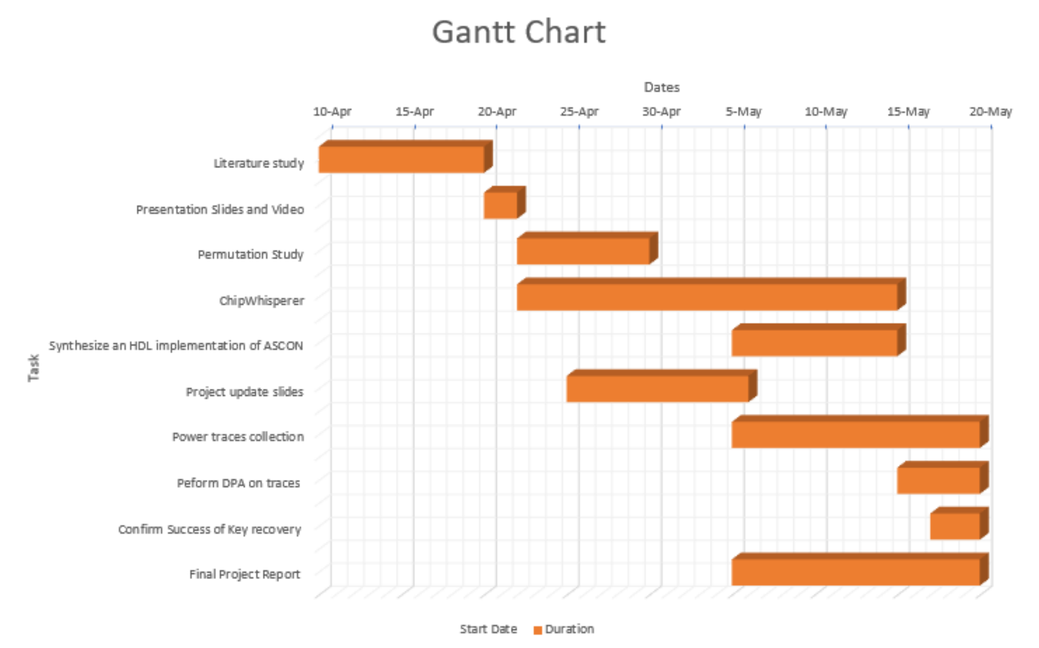


Figure 14: Gantt chart showing our final schedule

setup. The results obtained from the varying numbers of traces would have contributed to a deeper understanding of the attack's behaviour and its dependency on the number of traces. This knowledge would have been invaluable in guiding future DPA mitigation strategies and designing more secure cryptographic systems. The findings could have also served as a foundation for further research and advancements in the field of side-channel analysis and hardware security.

With an additional month of work, several improvements could have been made to enhance the success of Differential Power Analysis (DPA) testing on the Chipwhisperer platform. Firstly, more extensive data collection could have been conducted, increasing the number of power traces to improve the statistical power of the analysis. Furthermore, a longer testing period would have allowed for the exploration of different machine learning algorithms and feature selection techniques to optimize the performance of DPA. Additionally, more time could have been allocated to fine-tune the chipwhisperer setup, improving the synchronization and accuracy of the power traces. We would have also had the opportunity to carry out a power analysis of a masked ASCON referred to in [LB22]. Finally, with the extra time, comprehensive countermeasures could have been implemented and evaluated to mitigate the effectiveness of DPA attacks.

To effectively perform side channel analysis using ChipWhisperer, practical experience with the hardware would have been beneficial for quick debugging of setup errors, minimizing deployment time for ciphers. Familiarity with hardware design, particularly with FPGAs, is crucial for configuring and optimizing the FPGA setup for analysis.

We collected a lot of traces, hence expertise in signal processing techniques and statistical analysis would have played a vital role in extracting meaningful information from power traces and accurately interpreting the results. This knowledge allows for effective identification and exploitation of side channel leakage. A solid foundation in security engineering and risk assessment is essential for successful side channel analysis. This expertise enables effective threat modeling, understanding the attack surface, and selecting appropriate countermeasures to mitigate side channel vulnerabilities. Understanding the principles of cryptography and cryptographic systems is also crucial for comprehending the algorithms employed by the target cipher.

Overall, a combination of practical experience with ChipWhisperer hardware, expertise in signal processing and statistical analysis, and a strong foundation in security engineering and risk assessment would have enhanced the proficiency and accuracy of our side channel analysis on the ChipWhisperer.

8 Conclusion and Future Scope

This paper presents an evaluation of the side channel leakage resistance of the ASCON family of ciphers, aiming to contribute to the ongoing standardization efforts of lightweight cryptographic ciphers in resource-constrained environments. Through a comprehensive study, we gained a deep understanding of the ASCON cipher's operational mechanisms, including its various stages in the encryption process. We successfully implemented standard Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) attacks on hardware targets, enabling the recovery of secret cryptographic keys.

Moreover, we effectively set up the ChipWhisperer system, ensuring seamless integration with the hardware targets. Leveraging the ChipWhisperer API, we proficiently recorded and analyzed side channel power traces obtained from the encryption circuits. However, due to time constraints, our analysis was limited to a preliminary stage. Given additional time, we would have further refined and finalized the leakage model specific to ASCON. This enhanced model would have been applied to the collected power traces, allowing for a more accurate and precise recovery of the private key.

The findings presented in this study provide valuable insights into the vulnerability of the ASCON cipher to side channel attacks. The successful implementation of DPA and CPA attacks highlights the importance of considering side channel leakage in the design and deployment of lightweight cryptographic ciphers. Furthermore, our work demonstrates the effectiveness of the ChipWhisperer system as a tool for side channel analysis, emphasizing its potential for future research and development in the field of hardware security.

In conclusion, this research contributes to the understanding and evaluation of side channel vulnerabilities in lightweight cryptographic ciphers. By addressing the limitations and exploring further refinements, future studies can build upon our findings to enhance the security and resilience of cryptographic systems in resource-constrained environments.

References

[AES12] Fpga implementations of advanced encryption standard: A survey, figure 4, 2012.

[A.M21] O.Kara M. J. Mihaljević A.Mileva, V.Dimitrova. Catalog and illustrative examples of lightweight cryptographic primitives. 2021.

[BLA17] William J Buchanan, Shancang Li, and Rameez Asif. Lightweight cryptography methods. *Journal of Cyber Security Technology*, 1(3-4):187–201, 2017.

[chi23] Analyzer — chipwhisperer 5.7.0 documentation, 2023.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2. *Submission to the CAESAR Competition*, 5(6):7, 2016.

[DPAa] Aes-256 rsm reference traces, lecture 10.

[DPAb] Dpa contest lab, attack on the masked aes.

[DPA13] Description of the masked aes of the dpa contest v4, 2013.

[DTO20] Alex Dewar, Jean-Pierre Thibault, and Colin O'Flynn. Naean0010: Power analysis on fpga implementation of aes using cw305 & chipwhisperer, 2020.

[EB] Francis Olivier Eric Brier, Christophe Clavier. Correlation power analysis with a leakage model.

[Eic23] M. Eichlseder. Pyascon: A python implementation of the ascon authenticated encryption scheme, 2023.

[Fiv16] Michael Fivez. Energy efficient hardware implementations of caesar submissions. *Master's thesis, KU Leuven*, 2016.

[KA21] Shilpa K. and Chinchu A. A review on lightweight block ciphers. 2021.

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1:5–27, 2011.

[LB22] Lukasz Chmielewski Ellen Gunnarsd´ottir Vahid Jahandideh Tom Stock L´eo Weissbart Lejla Batina, Ileana Buhan. Side-channel evaluation report on implementations of several nist lwc finalists, 2022.

[LWL$^+$22] Sinian Luo, Weibin Wu, Yanbin Li, Ruyun Zhang, and Zhe Liu. An efficient soft analytical side-channel attack on ascon. In Lei Wang, Michael Segal, Jenhui Chen, and Tie Qiu, editors, *Wireless Algorithms, Systems, and Applications*, pages 389–400, Cham, 2022. Springer Nature Switzerland.

[new23] Chipwhisperer jupyter notebook repository, 2023.

[NIS18] NIST. Lightweight cryptography, 2018.

[OL16] Douglas Carson Owen Lo, William J. Buchanan. Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa). *Journal of Cyber Security Technology*, 1(6):88–107, 2016.

[Rama] Keyvan Ramezanpour. Active and passive side-channel key recovery attacks on ascon.

[Ramb] Keyvan Ramezanpour. Side-channel evaluation report on implementations of several nist lwc finalists.

[SD17] Niels Samwel and Joan Daemen. Dpa on hardware implementations of ascon and keyak. In *Proceedings of the Computing Frontiers Conference*, CF'17, page 415–424, New York, NY, USA, 2017. Association for Computing Machinery.

[SP16] Niels Samwel and Kostas Papagiannopoulos. Side-channel analysis of keccak and ascon. 2016.

[Tec] NewAE Technology. Chipwhisperer.

A Appendix 1 - Correlation Power Analysis of AES

```
import chipwhisperer.analyzer as cwa
import chipwhisperer as cw
from chipwhisperer.analyzer.attacks.models.aes.key_schedule import
    key_schedule_rounds

# load project from disk
project = cw.open_project('ascon_analysis/aes-data-may-20-1106/aes-data-may
    -20-1106/Tutorial_HW_CW305.cwp')

# print total number of traces
print(len(project.traces))
for ind, trace in enumerate(project.traces):
    print("Plain text!!!!!!!")
    print(trace.textin)
    print("Cypher text!!!!!!!")
    print(trace.textout)
    print("Key!!!!!!!")
    print(trace.key)

    if ind==4:
        break

attack = cwa.cpa(project, cwa.leakage_models.last_round_state_diff)
cb = cwa.get_jupyter_callback(attack)
attack_results = attack.run(cb, update_interval =10)

#Recovered the key from the last round of AES. We then used the analyzer to
    get the actual AES key:
recv_lastroundkey = [kguess[0][0] for kguess in attack_results.find_maximums
    ()]
recv_key = key_schedule_rounds(recv_lastroundkey, 10, 0)
for subkey in recv_key:
    print(hex(subkey))

#Key in Hex
#[ 43 126  21  22  40 174 210 166 171 247  21 136   9 207  79  60]
# ==>>>>  2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
```

B Appendix 2 - Correlation Power Analysis of ASCON

```
import ascon
import chipwhisperer.analyzer as cwa

import chipwhisperer as cw
proj = cw.open_project('ascon_analysis/data-may19-1616/data-may19-1616/
    ASCON_CW305_same_key_final.cwp')


# print total number of traces
print(len(proj.traces))

# print project data [first 10 elements]
for ind, trace in enumerate(proj.traces):
    print("Plain text!!!!!!!")
    print(trace.textin)
    print("Cypher text!!!!!!!")
    print(trace.textout)
    print("Key!!!!!!!")
    print(trace.key)

    if ind==4:
        break


#Test to lay out How ASCON works

keysize = 16

# choose a cryptographically strong random key and a nonce that never repeats for the same key:
key   = get_random_bytes(keysize) # zero_bytes(keysize)
nonce = get_random_bytes(16)      # zero_bytes(16)

associateddata = b"ASCON"
plaintext      = b"ascon"

print("Automatic test with general function")
ciphertext = ascon_encrypt(key, nonce, associateddata, plaintext)
print(ciphertext)
print()
print()


S = [0, 0, 0, 0, 0]
k = len(key) * 8   # bits
a = 12   # rounds
b = 6   # rounds
rate = 8   # bytes

print("Manual implementation of Perputation-----------")
print("Before the storm->>>",S)
print()
for r in range(0, 1):
        # --- add round constants ---
        S[2] ^= (0xf0 - r*0x10 + r*0x1)
        print("Addition of round constant")
        print("TARGET!!!!!->>>>",S)
        print()

         #yi
        #y0 = x4x1 + x3 + x2x1 + x2 + x1x0 + x1 + x0
        #y0 = x1(x4 + x2 + x0 + 1) + x3 + x2 + x0
        #y0 = x1(x4 + 1) + x1x2 + x1x0 + x3 + x2 + x0
        #y0 = x1(x4 + 1) + x3
        #y0 = key(nonce+1) + nonce

        #y0 = k( m   + 1) + m

        #Si(M, K  ) =  k 0 ( m i  + 1) + mi +  k 1 ( m i +45 + 1) (5)+ mi+45 +  k 2 ( m i +36 + 1) + mi+36
        # M is the 128-bit nonce split up into two registers m, m
        #  k i  is a bit from a key guess

        #Second part of the key:
        #Si(M, K  ) = mi( k 0  + 1) +  m i  + mi+3( k 1  + 1) (6)+  m i +3 + mi+25( k 2  + 1) +  m i +25

        print("Will target using Hardware implentation of the leakage model ->>>>",S)

        # --- substitution layer ---
        S[0] ^= S[4]
        S[4] ^= S[3]
        S[2] ^= S[1]
        T = [(S[i] ^ 0xFFFFFFFFFFFFFFFF) & S[(i+1)%5] for i in range(5)]
        for i in range(5):
            S[i] ^= T[(i+1)%5]
        S[1] ^= S[0]
        S[0] ^= S[4]
        S[3] ^= S[2]
        S[2] ^= 0XFFFFFFFFFFFFFFFF

        print("After non-linear round")
        print("TARGET!!!!!->>>>",S)
        print()


        # --- linear diffusion layer ---
        S[0] ^= rotr(S[0], 19) ^ rotr(S[0], 28)
        S[1] ^= rotr(S[1], 61) ^ rotr(S[1], 39)
        S[2] ^= rotr(S[2],  1) ^ rotr(S[2],  6)
        S[3] ^= rotr(S[3], 10) ^ rotr(S[3], 17)
        S[4] ^= rotr(S[4],  7) ^ rotr(S[4], 41)
        print("After final round")
        print(S)
        print()

val = bytes.fromhex('00000000000000007f7f7f7f7f7f7f7f')
print ("bytes.fromhex-->",val)

print()
print("Leakage function start")
class ASCON128A_Round13_Model(cwa.AESLeakageHelper):
    def leakage(self, pt, ct, guess, bnum):
        guessedit = to_bytes(guess)

        S = [0, 0, 0, 0, 0]
        k = len(key) * 8 # bits
        a = 12 # rounds
        b = 8
        rate = 16

        ascon_initialize(S, k, rate, a, b, guessedit, bytes.fromhex('
    00000000000000007f7f7f7f7f7f7f7f'))
        ascon_process_associated_data(S, b, rate, pt)


        print("moment of truth",bnum)

        result = S[1] [bnum]
        if bnum > 7:
            result = S[2] [bnum%8]
        return result


print("Leakage instance creation")
# Create an instance of the leakage model
leak_model = cwa.leakage_models.new_model(ASCON128A_Round13_Model)
print("done")
print()

print("CPA Start!!!!")
attack = cwa.cpa(proj, leak_model)
results = attack.run()
print("Attacked!!!!")
print(results)
```