

# Hardware Vulnerability Tool

A Major Qualifying Project submitted to the faculty of  
WORCESTER POLYTECHNIC INSTITUTE  
in partial fulfillment of the requirements for the degree of  
Bachelor of Science

Submitted by

Jeffery Collard

Valentina Harrison

Date

May 5, 2021

Advised by

Adrienne Hall-Phillips

Patrick Schaumont

*This report represents work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.*

### Abstract

All types of devices are potentially vulnerable to physical data leakage. In order to assess device security pre-fabrication, a simulator that can determine a cryptographic system's vulnerability to side-channel attacks is useful. Researching and understanding differential power analysis (DPA) attacks and side channel vulnerabilities of devices, is of great interest to manufacturers, consumers, and security scholars. The purpose of our project is to create a simulator to detect the vulnerabilities in a user's hardware. Our product uses ModelSim to simulate the operation of the device, then runs a side-channel-based attack on the data from the simulation. Based on the results of this attack, the user can understand their systems vulnerability to side-channel attacks and take appropriate precautions. This way, the developer can make the needed adjustments without the need for manufacturing a physical prototype on which side-channel leakage can be tested.

**Acknowledgements**

Our team would like to thank our advisors for their help and input within the project. Without them the project would not have been able to happen. Our team would like to thank Professor Hall-Phillips from the Foisie Business School for her knowledge and help with the management aspect of our project. Our team would also like to thank Professor Schaumont from the Electrical Computer Engineering department at WPI for his knowledge and help with the technical aspect of our project. Our team would also like to thank WPI for the ability to complete a Major Qualifying Project to gain a deeper understanding of our majors.

## Table of Contents

<b>Introduction</b> .....	8
<b>Literature Review</b> .....	9
<b>Side Channel Attacks</b> .....	10
<b>Attacks</b> .....	10
<b>Levels of Accuracy</b> .....	12
<b>Power Models</b> .....	13
<b>Differential Power Analysis Side Channel Attacks</b> .....	14
<b>Methodology</b> .....	22
<b>Simulator Development</b> .....	22
<b>Results</b> .....	26
<b>Hardware Simulation Results</b> .....	26
<b>Python DPA Attack</b> .....	29
<b>Vulnerability Demonstration</b> .....	32
<b>Conclusion</b> .....	33
<b>Results</b> .....	33
<b>Stakeholders</b> .....	34
<b>Limitations</b> .....	36
<b>Future Research</b> .....	38
<b>Conclusion</b> .....	39
<b>References</b> .....	40

## Executive Summary

Currently, there are not a lot of reliable simulators that help detect a software systems vulnerability to differential power analysis side channel attacks. Side channel attacks are a type of cryptographic vulnerability of a system where key information can be learned by an attacker through information other than the inputs and outputs of the system. When delving deeper into our project to gain a better understanding of what we wanted to accomplish we needed to select the type of side channel leakage we wanted to tackle. When we were deciding what to attack and what we wanted to accomplish we had to decide if we wanted to do offensive or defensive technique to protect against the attacks. As a team we decided to do a more offensive approach to side channel attacks. After thoughtful consideration as a team, we decided to use Differential Power Analysis (DPA) as our side-channel attack technique. Our reason for picking this type of side channel attack is not only is it something that interests both of us, but it allows us to take an offensive approach to the side channel attack.

Differential power analysis uses multiple power traces and uses comparisons between the actual measurements and the hypothetical measurement based on a power model. The power model is created using a partial cryptographic key guess. A successful DPA attack involves knowing the power consumption, what algorithm is computed, and need plaintext or ciphertext (Skorobogatov, 2011, p.8). For our project to work we will need a deep understanding of DPA attacks. Differential power analysis attacks are conducted by observing several cryptographic operations and measuring power traces during the operation, then, depending on the encryption implementation mathematical analysis is performed on the set of power traces. All differential power analysis attacks use a “divide and conquer” strategy (Randolph, 2020, p.2). That is, they

define a function that maps each key into a subkey, allowing them to make a series of subkey guesses using their mathematical analysis on the power trace sets.

Our team started our project by familiarizing ourselves with Verilator. We then used Modelsim for Verilog implementation and tested it with a simple test bench to verify that the encryption works. We also used a value change dump to run the DPA attack algorithm. Our team found that we would get more accurate results if we run 100 encryptions instead of ten. After that we used Python with Verilog VCD library to parse the VCD file and write our attack. Our DPA algorithm generates the most likely key guesses for each encryption so our next step is to lay out the data in a series of graphs that we will provide to the user. For best results we decided to use the IPython and the Pandas library to present the data for the user. To confirm our methods, we made sure to test along the way instead of running into an issue at the end.

Our project was used to help meet a management engineering requirement which allowed us to gain a deeper understanding of the business implications of our project. To do this we aimed to make the tools and simulator as user friendly as possible to allow more people to use the tools without issues. The most important part of our project in a business view is to understand the main stakeholders of our project. This allowed us to develop our product in a way that was best for the stake holders. Understanding the stake holders also helps us show the results in the most efficient way.

As our team gained a deeper understand of our project and started producing final results for the project, we were able to solidify who the main stakeholders of our project. Beforehand our team focused on understanding the main groups of stakeholders. Our team decided to focus mainly on four stakeholders we would be interested in our project. The stakeholders we choose

to focus on are electrical engineering business firms, engineering professors, electrical computer engineering professionals, and electronic hobbyists.

Our main goal was to create a tool to help users mitigate side channel leakage that might occur in their technology. Though there are so many diverse types of side channel attacks, we chose to focus on differential power analysis (DPA). This choice allowed us to create solutions from an offensive view instead of from a defensive view.

## Introduction

Every device is vulnerable to attacks by hackers. When creating new devices companies attempt to develop devices that are safe and free of security vulnerabilities. Another concern is when a company is attacked, they lose money during the process of having to recall hardware projects. The more reliable of a device a company makes the more likely customers will return to the company. Currently, there are not a lot of reliable simulators that help detect a software systems vulnerability to differential power analysis side channel attacks. Electrical companies want to protect their devices against hardware attacks to keep their business running. If devices from one company tends to be vulnerable to attacks consumers will no longer go to that company slowly shutting down the company. To help protect companies from shutting down due to their devices being vulnerable to attacks we set a main goal of our project. Our goal is to use ModelSim, an environment used for hardware simulation, to help us develop a tool to simulate power analysis attacks on a system. Our team completed testing our simulator and recommended strategies to mitigate the possibility of side channel attacks. Our project can also be used to help companies guarantee that their devices are safer and more protected against DPA attacks than competitors' products.

Looking into this project you might be thinking “what is a side channel attack?” Side channel attack are a type of attack that “takes advantage of patterns in the information exhausted that computers constantly give off: the electric emissions from a computer monitor or hard drive” (Greenberg, 2020). Our project aims to develop a tool that will help protect against side channel attacks it also is able to be used in electrical businesses to allow their devices to be protected against these attacks. Some characteristics of side channel leakage are power consumption, timing, acoustic and electromagnetic emissions. With side channel attacks there are a variety of



different types of attacks. The main categories being cache-based attacks, thermal attacks, electromagnetic attacks, and power-based attacks.

When delving deeper into our project to gain a better understanding of what we wanted to accomplish we needed to select the type of side channel leakage we wanted to tackle. This is due to there being so many types of side channel leakage we would run out of time before we were to accomplish methods for protecting against these attacks. Our team believed that power-based side-channel leakage attacks are important to protect against so our product will allow business to help protect their products. When we were deciding what to attack and what we wanted to accomplish we had to decide if we wanted to do offensive or defensive technique to protect against the attacks. As a team we decided to do a more offensive approach to side channel attacks. This allowed us to think a step ahead of attackers. After thoughtful consideration as a team, we decided to attack differential power analysis side channel attacks. Our reason for picking this type of side channel attack is not only is it something that interests both of us, but it allows us to take an offensive approach to the side channel attack. We also wanted to focus on this type of attack because it allows companies to save money if they are able to protect against attacks.

### **Literature Review**

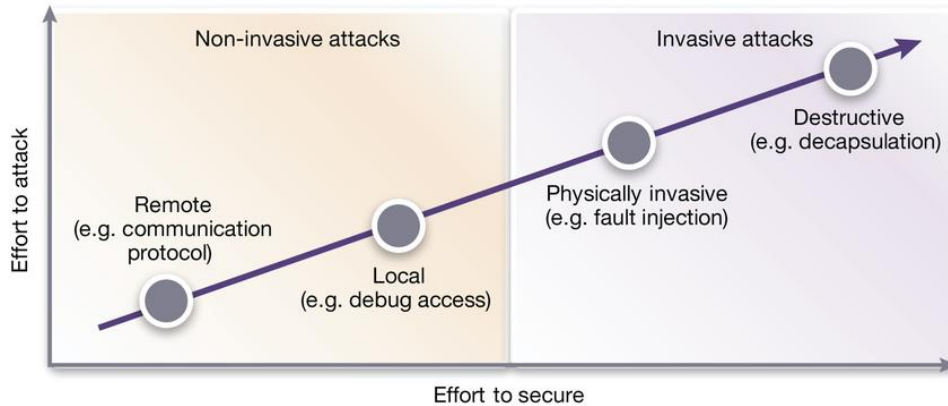
This chapter presents more information about different type of side channel attacks. This chapter is meant to give a better understanding of all types of side channel attacks, but it focuses on differential power analysis side channel attacks. We also provide a deeper context of why we chose differential power analysis side channel attacks.

## Side Channel Attacks

Side channel leakage are a type of cryptographic vulnerability of a system where key information can be learned by an attacker through information other than the inputs and outputs of the system. Side channel attacks are considered the “most powerful attack that can be mounted by an adversary. This is because the adversary is assumed to have physical access to the physical device where the crypto algorithm is implemented and executed” (Ye, 2015, p. 28). These kinds of attacks are new, first being proposed by Paul Kocher *et al* in their 1999 paper, *Differential Power Analysis*. Before that, researchers focused more effort on the development of mathematical encryption and security algorithms. More recently, researchers have focused more on implementation and countermeasures (Chen, 2006, p. 76). With side channel attacks there are a variety of different types of attacks. The main categories being cache-based attacks, thermal attacks, electromagnetic attacks, and power-based attacks. In cache-based attacks, there are two subcategories, access-based and timing-based, where access attacks observe the vector of the cache lines accessed during operation to gain information, and timing attacks observe the number of cache misses. Electromagnetic attacks observe the fluctuations in electromagnetic fields to gain information. However, our project will focus on power-based attacks, because they are a type of attack that can be protected against if users take an offensive approach.

**Attacks** When it comes to side channel attacks there are many different types. These types of attacks can be broken into invasive attacks and noninvasive attacks. Figure 1 shows a chart of different types of attacks. Some of the attacks shown are possible attacks on chips not solely on side-channel attacks. This chart is useful because it shows the line between invasive attacks vs non-invasive attacks. Examples of invasive attacks are micro-probing, reverse engineering, and fault attacks. Invasive attacks are important for businesses to protect against because it attacks the hardware making their device defective. The more vulnerable their devices

are to these types attacks the less customers will trust them and the business will slowly lose customers. Micro-probing is a method where you directly probe an integrated circuit after package decapsulation and possible removal of the chip's upper passivation layer. The next type of invasive attack is reverse engineering attacks. The last type of invasive attacks are fault attacks.



*Figure 1.* Hardware Attacks (Neustadter, 2018)

This is where the environmental conditions of the circuit are manipulated to generate faults (Skorobogatov, 2011, p. 9). Unlike micro-probing and reverse engineering attack's fault attacks are only considered semi-invasive attacks. The other category of attacks are noninvasive attacks such as timing attacks, simple and differential power analysis, and electromagnetic analysis. Noninvasive attacks are useful because it does not require the device to be opened. (Skorobogatov, 2011, p. 5). Noninvasive attacks are important for the businesses to protect against because customers want a reliable device. These types of attacks are important to protect against allowing the businesses to test multiple times before the product is released to allow them to guarantee that their devices are protected. Another reason companies and businesses should be concerned against protecting against noninvasive attacks is because if their own devices are attacked within the company the entire system might shut down. There are different outcomes

that could happen if the company is attacked such as system shutting down, data leakage, and files lost. This is not good for a company because it slows down operations for the day and also makes customers not trust their data with the company. Another issue comes when a company needs to recall devices due to how vulnerable to attacks, they are. The first type of noninvasive attacks are timing attacks. These types of attacks were founded in 1996 and they “exploits the observation that computations performed in some of the cryptographic algorithms often take different amounts of time on different inputs” (Peeters, 2013, p. 13). Another type of noninvasive attack would be simple and differential power analysis attacks. These types of attack record the power traces that are leaked by a device and analysis it. The last type of noninvasive attack is electromagnetic analysis. This type of attack can recover information from a device by exploiting the electromagnetic emanations due to the current flowing through the device (Skorobogatov, 2011, p. 8). Our project focus on differential power analysis side channel attacks which is a type of power-based attack. In Figure 2 below it shows different types of attacks that can occur on a device. Our team is focusing on the power aspect of attacks allowing us to take an offensive approach to the attacks.

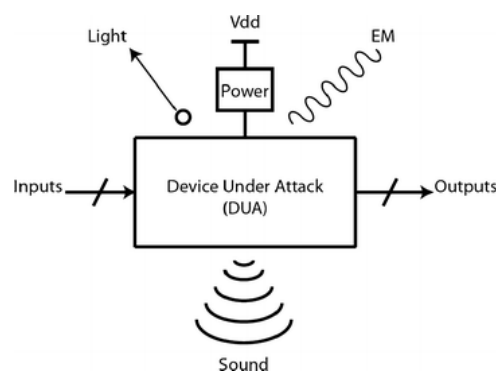


Figure 2. Attacks (Mai, 2012, p. 176)

**Levels of Accuracy.** When looking into power consumption one would find that there are three modeling abstraction levels: analog, logic, and behavioral. The most accurate power

consumption simulation is the analog level. “The basis of such a simulation is a transistor netlist of the circuit that lists all transistors of the circuit and the connections between them” (Mangard, 2007, p. 34). The next level of power simulation is the logic level which has a lower precision than an analog level, but it does require fewer data points. “The basis of a power simulation at the logic level is a netlist of the cells of the circuit” (Mangard, 2007, p. 35). The netlist is important for attacks because it shows the connection between different cells within a circuit. The last type of level is the behavioral level this level has a fast power simulation. Though the power simulation is fast it tends not to be an accurate method to simulate the power consumption. “The basis of a power simulation at this level is a high-level description of the digital circuit. This high-level description contains the major components of the digital circuit (microcontrollers, memories, dedicated hardware modules, etc.) and some high-level power models of these components” (Mangard, 2007, p. 37). The information found within the behavioral level is important to power analysis attacks because they are the only power simulation level that is important. This is because they have information on “data-dependent and the operation-dependent portions of the power consumption” (Mangard, 2007, p. 37).

**Power Models.** In power analysis attacks there are multiple different types of power models that attackers will look at; these attacks are either under the categories of Hamming-Distance models, Hamming-Weight models, or other. The Hamming-Distance model is a way to compare binary data which is normally two strings. In this model “attackers commonly use the Hamming-Distance model to describe the power consumption of buses and registers” (Mangard, 2007, p. 40). This model is used to simulate power consumption when the attacker knows consecutive values in the netlist. Another type of model is the Hamming-Weight model. This model is easier to use than the Hamming-Distance model. Unlike the Hamming-Distance model

this model can be used if the attacker has no information on the netlist. “In the case of the Hamming-Weight model, the attacker assumes that the power consumption is proportional to the number of bits that are set in the processed data value” (Mangard, 2007, p. 40). The attackers prefer looking at the Hamming-Distance model but will look at a Hamming-Weight model if they do not have the required information for a Hamming-Distance model. Sometimes there are other types of models, but it varies device to device. These models tend to be a subcategory of Hamming-Distance models.

### **Differential Power Analysis Side Channel Attacks**

Power based attacks have two main types of attacks, simple power analysis (SPA) and differential power analysis (DPA). Simple power analysis uses one power trace to learn the key to a system, usually by attempting to learn all the bits of a cryptographic variable. Differential power analysis uses multiple power traces and uses comparisons between a hypothetical key passed through the system with one measured power stat and the actual measurement. Simple Power Analysis (SPA) “examines a chip’s current consumption over a period of time” (Side-channel attacks, 2017). This type of attack allows an attacker to understand what function is being performed at a specific period of time. In this type of attack, it is important to evaluate the ratio of 1 vs 0 and that can be shown within the power. In Figure 3 this is an example of a SPA but shown in a simple way.

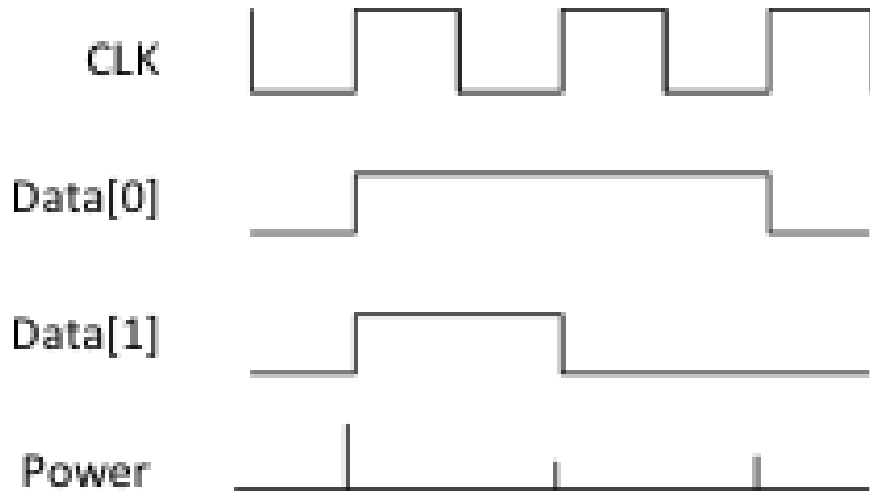


Figure 3. SPA Attack (Side-channel attacks, 2017)

SPA is best used when the power traces are known or are obvious. If there is noise in the system, it would not be an efficient method and that is when it is useful to look into a DPA.

A successful DPA attack involves knowing the power consumption, what algorithm is computed, and need plaintext or ciphertext (Skorobogatov, 2011).

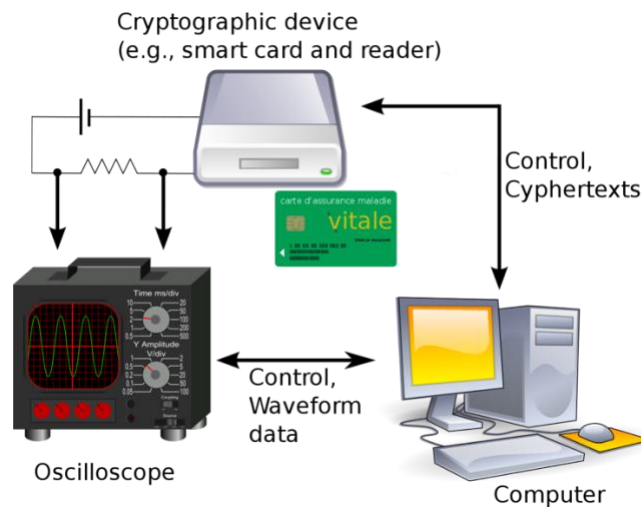


Figure 4. Measurement Setup for Differential Power Analysis (Pellegrini, 2009)

Differential Power Analysis (DPA) attacks are one of the most popular and powerful types of power attacks. DPA attacks were originally considered the “cheap” way to recover secret information from side channel leakages” (Batina, 2014, p. 148). This paper was based on

the concept “security faults often involve unanticipated interactions between components designed by different people.” (Velegalati, 2008, p. 1) “In the classic Differential Power Analysis, the first moment or mean is first used to reduce all the traces in each class down to a master trace. The class master traces are then compared at each point in the trace, to determine if those points are significantly different from each other.” (Velegalati, 2008, p. 1) This information will be used to help us understand and create our simulator to protect against DPA attacks.

Differential power analysis attacks are conducted by observing several cryptographic operations and measuring power traces during the operation, then, depending on the encryption implementation mathematical analysis is performed on the set of power traces. A company can help protect their customers if they are able to protect their devices from people being able to measure the power traces during operation. All differential power analysis attacks use a “divide and conquer” strategy (Randolph, 2020, p. 2). That is, they define a function that maps each key into a subkey, allowing them to make a series of subkey guesses using their mathematical analysis on the power trace sets. This increases the efficiency of improving the guesses by giving each subkey guess more defined inputs. The attacker then uses the mapping function to find the original key. This process can be used to attack many different encryption algorithms and has been proving effective and powerful repeatedly.

### **DPA on Different Devices**

In this section, we will discuss DPA attacks on different devices, including examples of DPA attacks on SIMON and LED Block Ciphers. The first attack we are going to dive into is the attack on SIMON.



According to Dillibabu Shanmugam, Ravikumar Selvam, and Suganya Annadurai “SIMON is based on Feistel structure and the algorithm supports various block and key sizes.” (Shanmugam, 2014, p. 112) Below we show an example of the power consumption of SIMON according to the paper “Differential Power Analysis Attack on SIMON and LED Block Ciphers.” This graph is important because it helps show the power difference within the encryption period. This graph has time in nanoseconds as the X values. As the Y values it shows the voltage during the encryption cycle.

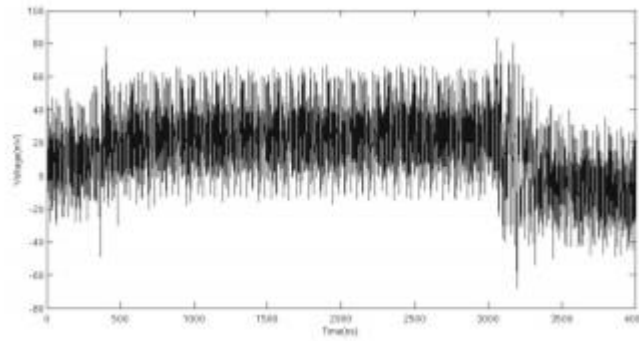


Figure 5. Power Consumption Of SIMON (Shanmugam, 2014, p. 112)

*Register value after First round*

$$L^2 = \underbrace{(K_{16}^1 \oplus R_{16}^1 \oplus L_{14}^1 \oplus (L_{15}^1 \& L_8^1))}_{L_{16}^2},$$

$$\underbrace{(K_{15}^1 \oplus R_{15}^1 \oplus L_{13}^1 \oplus (L_{14}^1 \& L_7^1))}_{L_{15}^2}, \dots, \underbrace{(K_1^1 \oplus R_1^1 \oplus L_{15}^1 \oplus (L_{16}^1 \& L_9^1))}_{L_1^2} \quad (8)$$

Figure 6. Register Value After First Round (Shanumgam, 2014, p. 114)

*Register value after Second round*

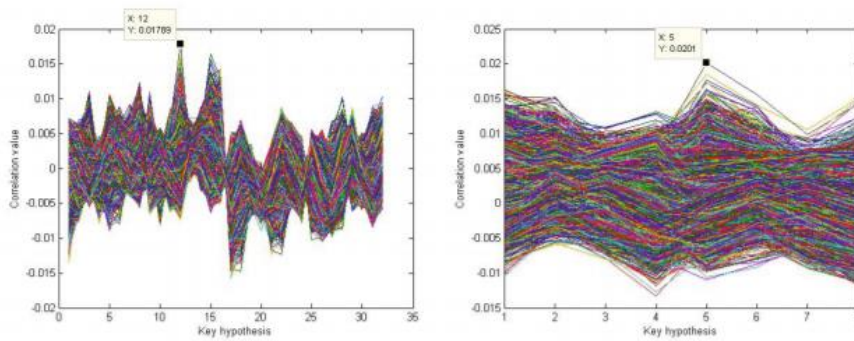
$$L^3 = \underbrace{(K_{16}^2 \oplus R_{16}^2 \oplus L_{14}^2 \oplus (L_{15}^2 \& L_8^2))}_{L_{16}^3},$$

$$\underbrace{(K_{15}^2 \oplus R_{15}^2 \oplus L_{13}^2 \oplus (L_{14}^2 \& L_7^2))}_{L_{15}^3}, \dots, \underbrace{(K_1^2 \oplus R_1^2 \oplus L_{15}^2 \oplus (L_{16}^2 \& L_9^2))}_{L_1^3} \quad (9)$$

Figure 7. Register Value After Second Round (Shanumgam, 2014, p. 114)

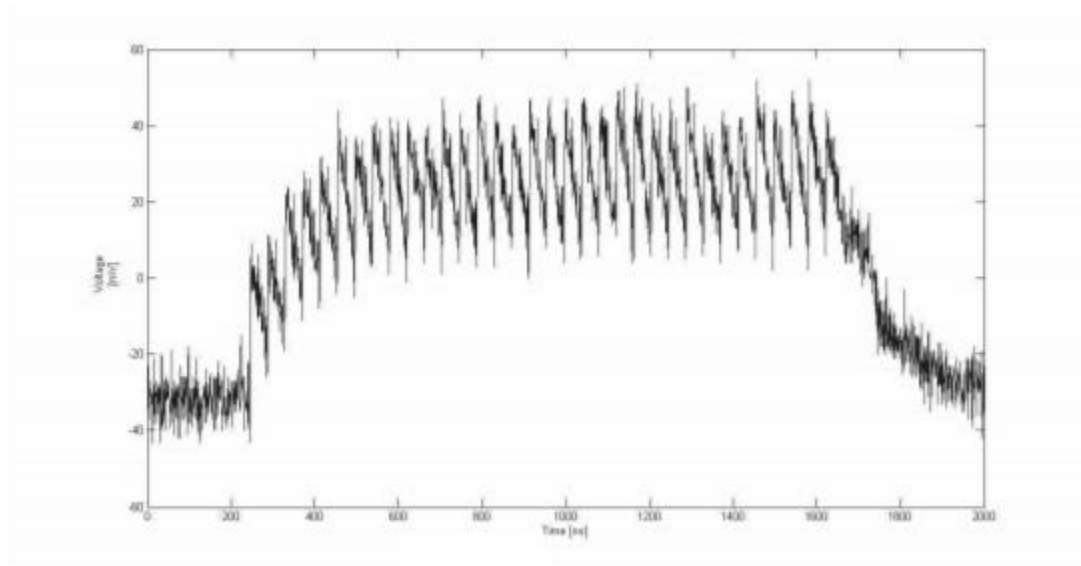
As stated earlier, attackers look into different types of models to structure their attack. For this attack the attackers focused on a Hamming Distance model. A Hamming distance model is used when the attacker knows consecutive values in the net list. In the paper "Differential Power Analysis Attack on SIMON and LED Block Ciphers" (Shanmugam, 2014, p. 110) they chose to focus on SIMON32/64 in order to show an attack. Above in Figure 6 and 7 are their register values after the first and second round. They chose  $L^3$  because it is considered the intermediate result and can be used to perform a DPA attack. Figure 8 is the DPA attack results in the form of a graph. When understanding the results of a DPA attack you will notice that there is one value that looks best in the graphs. In the case below you can see that it is (x 12, y 0.01789) and (x 5, 0.0201). On these graphs the X value are always the key hypothesis where the Y values are the correlation value. Figure 8 shows graphs where the left graph is the first bit correlation, and the right is the second bit correlation.

When we are looking to structure our results, we will be looking into the importance of a graph. The graph is important to prove that the DPA attack showed one value that was more likely than the rest of the values. The next device that the paper *Differential Power Analysis Attack on SIMON and LED Block Ciphers* investigates is LED Block Ciphers.



*Figure 8. Results of DPA Attack On SIMON (Shanmugam, 2014, p. 116)*

Our team found it important to look at both examples that this paper showed because it will help us structure our attack and gain a better understanding of how to format our results. “LED is based on a design principles of Advanced Encryption Standard (AES)” (Shanmugam, 2014, p. 117). Figure 9 shows the Power consumption of LED-64 according to the paper. This is important to show again because it looks different from the one above. It shows that the power consumptions change from device to device. Again, the X value is time in nanoseconds and the Y value is the voltage.



*Figure 9. Power Consumption of LED (Shanmugam, 2014, p. 117)*

Just like the model above they used the Hamming Distance model to perform the attack. Unlike SIMON the LED-64 runs on a row major matrix. This is important to know because it will change what the attack looks like. Below in figure 10 we have shown the power model for the first attack according to the paper *Differential Power Analysis Attack on SIMON and LED Block Ciphers*.

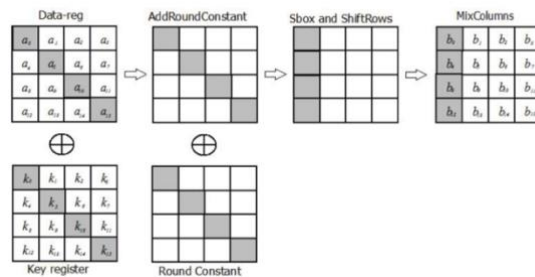
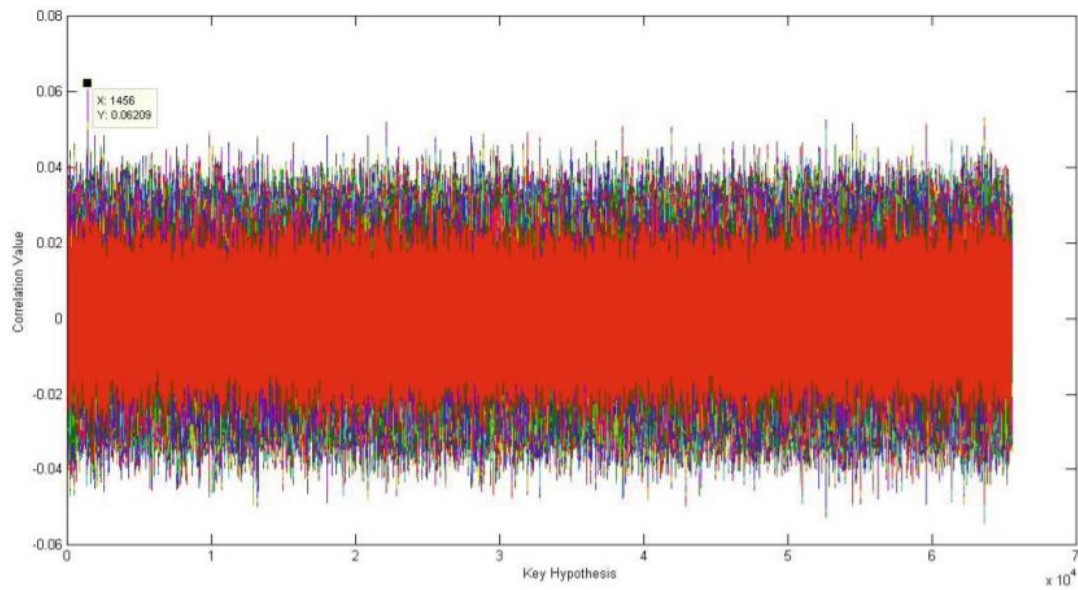


Figure 10. Attack Matrix of LED (Shanmugam, 2014, p. 118)

The attack looks like this because of the way that LEDs run, and the attack needs to be in the format of a matrix in order to be a successful attack. Finally, we wanted to understand their results of the attack. In Figure 11 you will see that the attack has more noise than the attacks before and the graph looks a bit different. This is because the graph varies device to device another factor that varies how results look is how many keys you give it. If you have only five it will be difficult to fully understand what one is better than the other. But in the LED attack you

can see that the X value ends up being 1456 and the Y value ends up being 0.06209. Again the x-axis shows the key hypothesis and the y-axis shows the correlation value.



*Figure 11.* Results of LED DPA Attack (Shanmugam, 2014, p. 120)

## Methodology

The purpose of our project is to create a simulator to detect the vulnerabilities in a user's hardware. To achieve this, we will start by attempting to use simple power analysis to attack the systems. Our methods include having testing phases in between each step allowing us to fix any problems that might arise. Another valuable tool when it comes to completing this project is having code that everyone on the team can look at. We are achieving this using GitHub and sharing latest versions of the code on there.

## Simulator Development

We started this project by familiarizing ourselves with Verilator and how to read the information we receive from it. However, we through this process we discovered that ModelSim was better suited for our needs. The ability to use the built in \$dumpvars command to generate a VCD file in line with our test bench code perfectly suited our needs. With that knowledge we used ModelSim for a Verilog implementation of AES encryption and we made sure to test it with a simple test bench to verify that it works. We wrote our encryption implementation and tested its functionality, finding that it functioned but did not have enough clock cycles per encryption for DPA to work. So, we set out to modify our design to use 10 cycles per encryption. To achieve this, we added an enable signal and a flip flop that would only pass new input into the modules at the positive edge of our clock signal when that enable signal was high. Then, in our top-level module, we created a state machine that updated at the negative edge of the clock signal. The enables of each individual round were tied to their own unique state in the state machine. At each negative edge, the state machine moved to the next state, triggering the subsequent round. After the final round, the state machine moved into a “ready” state which simply set a “ready” signal to high, indicating that a new encryption was finished. Then the

encryption would return to the initial state, load more data, and start the process anew. Our next step was to begin simulating power traces for our differential power attack.

We approached this by adding the \$dumpfile and \$dumpvars commands in our test bench module to generate a value change dump(VCD) file that records each bit flip during encryption.

```

46      +
46 47      $dumpfile("AES_VCD.vcd");
47 48      $dumpvars(0, uut);
48 49      $dumpvars;
50      +
51      +      $display("Variable dump commenced");
52      +      simReady=1;
53      +      memidx=0;
53      +      encrypts=0;
49 54      #10000;
50      -      $dumpoff;
51      -      $finish;
52 55      end

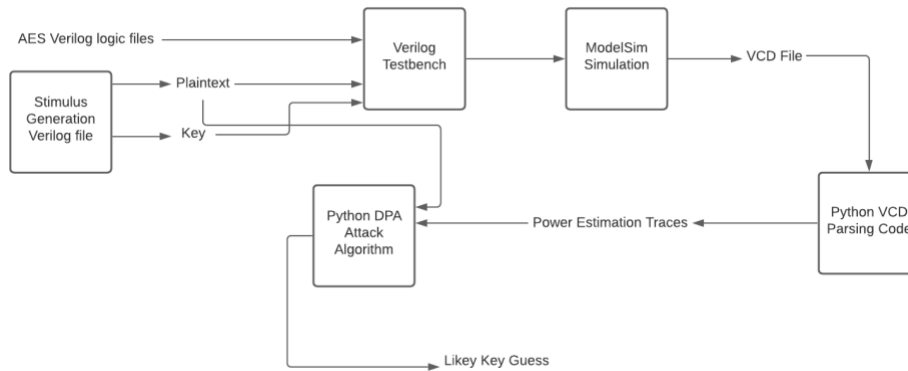
```

*Figure 11.* ModelSim \$dumpfile Example

We then wrote an algorithm that counted each time a bit went high by parsing this data. This gave us a one-to-one simulation of power draw during operation. Once we had our VCD file it was time to run our actual DPA attack algorithm. After much testing and advisement, we determined we would need at least 50 rounds of encryption to gather enough data for a successful attack. We decided to go with 100 encryptions for the sake of robustness. We used Python with the Verilog VCD library to parse the VCD file and write our attack.

The DPA algorithm generates likely key guesses for each encryption, by analyzing the correlation between possible key guesses for each byte and producing a graph for each byte. Our next step was to lay out the data in a series of graphs that the program provides to the user. We experimented with the Python libraries IPython and pandas to process and present the data. Then, we used this data to provide the first, second, and third most likely keys to the user. This process

will constitute a completed DPA attack. Figure 12 is a block diagram showing the flow of our overall process.



*Figure 12.* Block Diagram of Attack Process

The development of our methods required various testing between each step. This testing allowed us to confirm that each step works along the way. Another important part to each step is research. When we researched between each step or before we did different steps allowing us to see different challenges others might have ran into when attempting to do something similar. For example, we found in our research on the Verilog \$dumpvars command, that we could specify the level of abstraction we wanted included in our VCD. Looking at our research on DPA, we decided that every level under the test bench abstraction should be included.

**Importance of Stakeholders.** When starting a business or even creating a new device or product it is important to evaluate stakeholders. Stakeholders help shape the development of services or products because it helps the creator understand how ‘user-friendly’ the tool needs to be. For this element our team made the tool and simulator as user friendly as possible. We also worked hard to gain a deeper understanding of who could be the main stakeholders for a simulation tool. Using a stakeholder map, we evaluated possible stakeholders and their level of interest and influence.



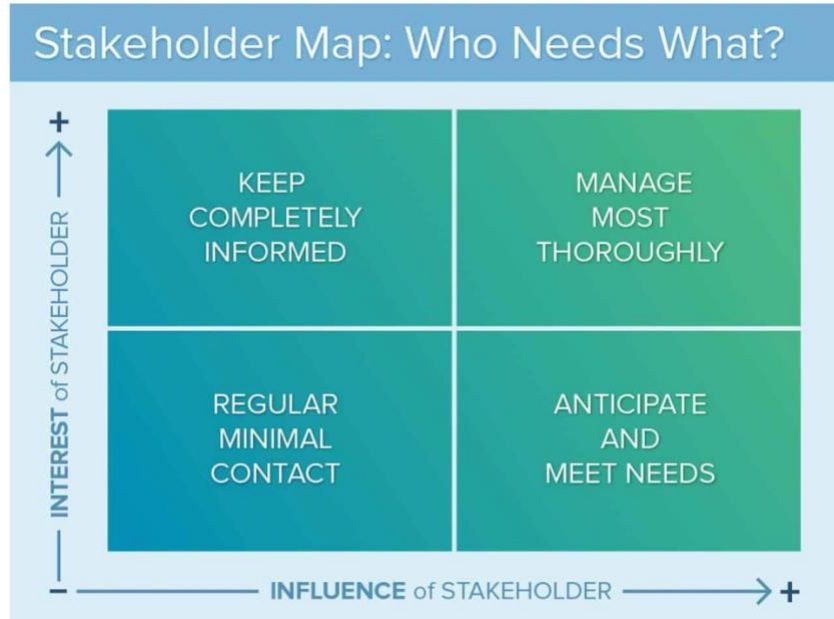


Figure 13. Stakeholder Map Example (Simon, 2016)

In Figure 13 above we show an example of a stakeholder map. These maps are very useful to help understand who to focus the project or the product around. Our team believes just from looking at the map that a majority of the stakeholders will fall around the manage most thoroughly section of the map. Figuring out stakeholders is a crucial step of our process because it will help show us how to develop our product the best we can for our stakeholders. This can be in a sense of marketing our product for the people who will use our product. It is also important to understand the main stakeholders because it will allow us to show our results in the most efficient way for our stakeholders. Stakeholders also are important to building a company because with a well-developed company stakeholder are found in many different forms from employees to people who use the product. Our stakeholders were found by looking into the results of the product and by figuring out how our product could be used in the market. Using the methods stated above allowed us to get the results we show next.

## Results

The purpose of our project is to create a simulator to detect the vulnerabilities in a user's hardware. This allows business to create reliable hardware and allows business to know they are safe from attacks. We accomplished this task by creating a simulator using ModelSim to understand DPA attacks. The results our team was able to simulate are discussed within this section.

### Hardware Simulation Results

In our first attempt at AES encryption simulation, we took a simple combinational approach. We created modules for the substitution box, key generation, and column mixing steps of AES, as well as a module each for the initial rounds and for the final round of encryption. Our top module simply passed the output from each round directly into the output for the next round. Therefore, our encryption occurred in one clock cycle. This proved to be problematic when we reached the DPA attack step, as there was nowhere near enough data per encryption to run a proper correlation attack because the generated power trace was too noisy for a successful DPA attack. In addition, we were running the encryption on the same data repeatedly.

As such, we reworked our Verilog modules. The modules for individual encryption steps remained the same. However, for the round modules, we added an enable signal and a flip flop that would only pass new input into the modules at the positive edge of our clock signal when that enable signal was high. Then, in our top-level module, we created a state machine that updated at the negative edge of the clock signal. The enables of each individual round were tied to their own unique state in the state machine. At each negative edge, the state machine moved to the next state, triggering the subsequent round. After the final round, the state machine moved into a "ready" state which simply set a "ready" signal to high, indicating that a new encryption

was finished. Then the encryption would return to the initial state, load more data, and start the process anew.

In addition, we added a stimulus generation module to generate a random and random plaintext data for our encryption. To achieve this, we simply used the System Verilog commands \$random and \$writememh to iteratively generate random bytes, then write those bytes into files to be read by our Verilog test bench.

We then had to add logic to our testbench to load in this data using the \$readmemh command. We first added logic to load our key, byte by byte, from the key file we generated using our stimulus generation module. Next, we added an always loop that triggered at each positive clock edge. This loop checks the ready signal from our top-level module as well as a “sim\_ready” signal we added and set in our test bench to indicate that the initialization steps of our simulation were complete. If both were high, the loop would load the next 4 lines of our plaintext file, with each line being 4 bytes long, thus giving our simulation a new 128-bit data input to be encrypted.

This process led to a more robust encryption process that was now 12 clock cycles long instead of one, giving us more data on which to run our attack. Figure 14 is an image of waveforms from the simulation as well as excerpts from our VCD file.

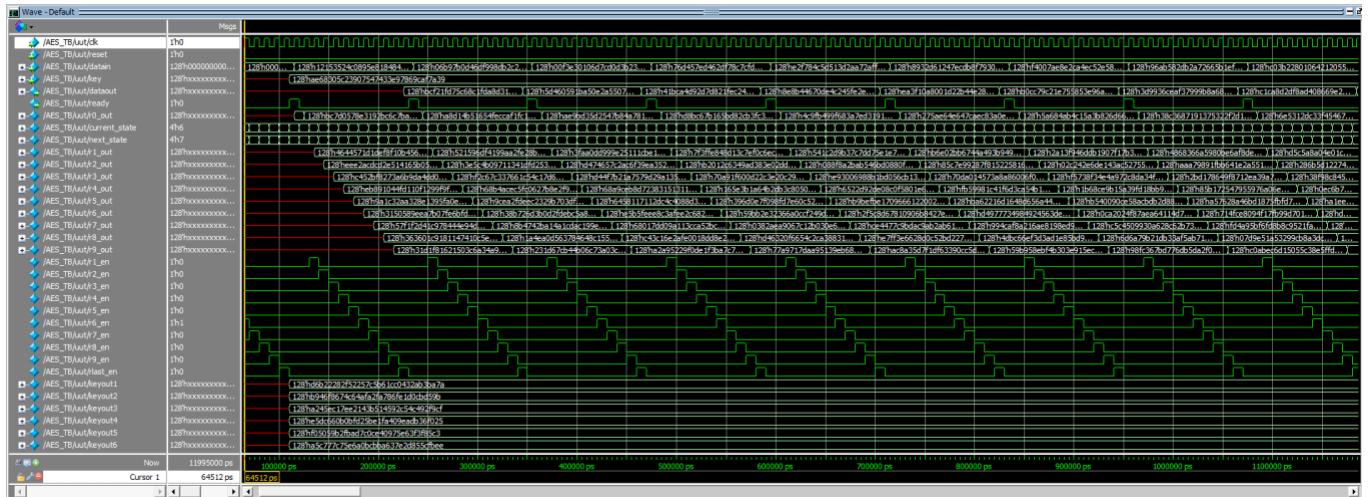


Figure 14. AES Encryption Simulation Waveform

```
$scope module uut $end #925000
$var wire 1 ! clk $end 1!
$var wire 1 " reset $end b110110110110110011110011011001000011101101100110011101101101101101101101101101101100011110001111101000110100111101101010
$var wire 1 # datain [127] $end 1#
$var wire 1 $ datain [126] $end 1$
$var wire 1 % datain [125] $end 0%-
$var wire 1 & datain [124] $end 1).
$var wire 1 ' datain [123] $end 1'.
$var wire 1 ( datain [122] $end 1(.
$var wire 1 ) datain [121] $end 0).
$var wire 1 * datain [120] $end 1*.
$var wire 1 + datain [119] $end 1k.
$var wire 1 , datain [118] $end 0j.
$var wire 1 - datain [117] $end 0h.
$var wire 1 . datain [116] $end 0a.
$var wire 1 / datain [115] $end 1'.
$var wire 1 0 datain [114] $end 1).
$var wire 1 1 datain [113] $end 1\..
$var wire 1 2 datain [112] $end 1|.
$var wire 1 3 datain [111] $end 12..
$var wire 1 4 datain [110] $end 0Y.
$var wire 1 5 datain [109] $end 1V.
$var wire 1 6 datain [108] $end 1U.
$var wire 1 7 datain [107] $end 00.
$var wire 1 8 datain [106] $end 1M.
$var wire 1 9 datain [105] $end 0K.
$var wire 1 : datain [104] $end 0F.
$var wire 1 ; datain [103] $end 1E.
$var wire 1 < datain [102] $end 0D.
$var wire 1 = datain [101] $end 1C.
$var wire 1 > datain [100] $end 1B.
$var wire 1 ? datain [99] $end 1@.
$var wire 1 @ datain [98] $end 0?.
$var wire 1 A datain [97] $end 1>.
$var wire 1 B datain [96] $end 0:..
$var wire 1 C datain [95] $end 19.
$var wire 1 D datain [94] $end 17.
$var wire 1 E datain [93] $end 15.
$var wire 1 F datain [92] $end 04.
$var wire 1 G datain [91] $end 03.
$var wire 1 H datain [90] $end 12..
$var wire 1 I datain [89] $end 0/.
$var wire 1 J datain [88] $end 1+.
$var wire 1 K datain [87] $end 0%.
$var wire 1 L datain [86] $end 0".
$var wire 1 M datain [85] $end 0~.
$var wire 1 N datain [84] $end 0j.
$var wire 1 O datain [83] $end 1j.
$var wire 1 P datain [82] $end 0z.
$var wire 1 Q datain [81] $end 1x.
$var wire 1 R datain [80] $end 0v.
$var wire 1 S datain [79] $end 0v.
$var wire 1 T datain [78] $end 1u.
```

Figure 15. VCD Variable Definition and Value Change Section Excerpts

After analysis and testing, it also became clear that our encryption was actually not vulnerable to a DPA attack. This was because our implementation used separate registers for each round of

encryption, preventing the algorithm from ever getting a lock on the correlation and providing a likely key guess. This implementation, under further review would not be viable due to the high memory cost.

**Python VCD Parsing.** In order to determine whether our encryption was vulnerable to a DPA attack, we wrote a Python script to load the data from the VCD file and run our attack. Our parsing method used the Python function “open” to read the VCD file line by line. Each line in a VCD file starts with a unique string character depending on the purpose of that line, so all we had to do was use the Python function strip to look at the first string in the array it returned. There were only a few we cared about so we could simply check for the correct strings or character and move on if none of them were present. The first thing we checked was whether the string was “\$timescale” indicating that this line was setting the timescale for the VCD file. In this case, we ran our calc\_mult function that calculated and set the time scale for the data. The next possibility was the string starting with a ‘#’, indicating the start of a new timestep. In this case we created a new time entry in our trace and initialized the entry to 0. Next, if the line started with ‘x’, ‘X’, ‘1’, ‘0’, ‘z’, or ‘Z’, that line was showing a value change. We would then check whether that value was ‘1’. If this was the case, this was a positive bit flip and we incremented our value for the current time step.

After all lines were parsed, we went through the trace data structure to reformat it, counting the timesteps as clock edges. We would store each value in a new entry for the current sample. Then once we reached 24 edges, indicating a new encryption, we would create a new sample and begin the process over again.

**Python DPA Attack.** Finally, it was time to run our actual DPA attack. The attack starts by going through our plaintext file line by line, reading each byte from the line and storing it in

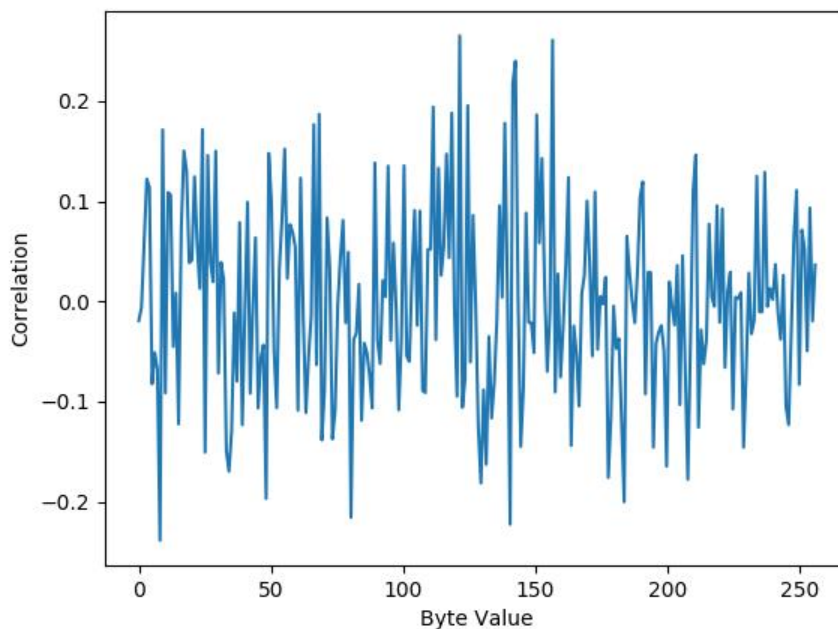
the “pbytes” array before moving on to the next line. When 4 lines have been stored, indicating a complete data set, it moves on to the next entry in the array and repeats the process until the entire file has been parsed. The next step is to populate a three-dimensional array with correlation numbers. For each encryption, the attack calculates a hamming weight value based on substitution box output of a plaintext byte and stores it in an array. The function then runs through each sample in the trace for that encryption and calculates the correlation between that sample and the estimate, before storing that entry in the array. This process is then repeated for every possible guess for that byte of the key, then in turn for every byte in the key. The correlation results are then dumped into separate files for each byte. Finally, the program goes through each byte correlation and looks for the biggest peaks in correlation for that byte, giving a most likely guess for that byte. When this process is finished, we have a complete guess for the key as shown in the image below.

```
Key byte 0
30
Position 10
SNRd 0.14413366318714713
Key byte 1
28
Position 18
SNRd 0.0880978947734466
Key byte 2
146
Position 21
SNRd 0.12104560985484523
Key byte 3
28
Position 16
SNRd 0.13380061526562944
Key byte 4
16
Position 12
SNRd 0.17375920502020323
Key byte 5
64
Position 19
SNRd 0.17322385918281577
Key byte 6
5
Position 8
SNRd 0.1665554585441853
Key byte 7
188
Position 0
SNRd 0.1131601513516975
Key byte 8
34
Position 17
SNRd 0.10735774666583878
Key byte 9
23
Position 11
SNRd 0.22119004773047474
Key byte 10
40
Position 16
SNRd 0.1847142278008994
Key byte 11
60
Position 20
SNRd 0.13329017497947912
Key byte 12
17
Position 0
SNRd 0.11367350060266287
Key byte 13
50
Position 20
SNRd 0.228128618481185
Key byte 14
11
Position 19
SNRd 0.13094154662847285
Key byte 15
20
Position 11
SNRd 0.14378872566498535
```

*Figure 16.* DPA Attack Initial Output

For each byte of the key, the program provides two guesses, each consisting of three pieces of information. The first being the actual guess for that byte, followed by the position of that guess in an array of correlation values. The final line is the standard deviation of the correlation of that guess, As you can see from the image, the standard deviation values are all fairly low, never reaching above .25, indicating that these guesses are most likely not accurate and indicative of a successful DPA attack.

The script also provides correlation plots for each byte of the key, an example of which is shown below in Figure 17.



*Figure 17. Correlation Plot*

The data from these plots is analyzed by the program, looking for significant spikes in correlation for specific byte values for the given byte. The values with the biggest spikes are the most likely guesses for that byte. This process is repeated for each byte, generating most likely guess for the value of the key. The data shown in this plot show a fairly noisy correlation signal,

further demonstrating an unsuccessful attack. This proves to be true as none of the combinations of our key guesses match with the key used by our AES simulation

**Vulnerability Demonstration.** Based on the current output of our attack, we are currently unable to prove vulnerability to DPA based on our methods. Our likely key guesses are currently not matching up with the key provided to our example encryption algorithm. This would likely suggest that our algorithm is not vulnerable to a DPA attack. However, since we know this not to be the case, there is something wrong with either our simulation, or our algorithm. To attempt to fix this problem we ran a series of tests to attempt to identify the issue. Our first step was to attempt to identify an issue with our simulation. We reran the simulation first using a constant key, hoping to see correlation spikes at the constant points in each byte guess. However, there was no significant change in our correlation data. Next, we attempted to identify a misalignment between the plaintext data passed to our simulation and the data passed to the attack algorithm. In debugging the alignment, we identified some inconsistencies and eliminated them, however these changes did not create any improvement in our correlation data. Next, we investigated our AES implementation. Our original design used a separate 128 bit register for each round of encryption. We realized that this method would not be feasible in actual implementation for fabrication as each instance of the encryption would require 10 times the amount of memory as the amount of data being encrypted. So, we changed our algorithm to use a single encryption register. Theoretically, this would allow our encryption algorithm to get a better lock on our encryption in order to break it. However, there was still no accurate correlation data. It was at this point that our team ran out of time to make any further changes.



## Conclusion

We set out on this project with the goal of creating a simulator to determine a systems vulnerability to side-channel attacks. We have made significant progress toward that goal however there is still more work to be done, either by us, or by future projects. We successfully created a method for simulating a power trace on a hardware implementation of an encryption method and in addition were able to write a Python script to run a complete DPA attack. Some of the work left to complete could be in further investigation of our power mode. In addition, there is a great deal of potential in development of this simulator as an easy-to-use tool that can just be sent out as a complete software package to clients.

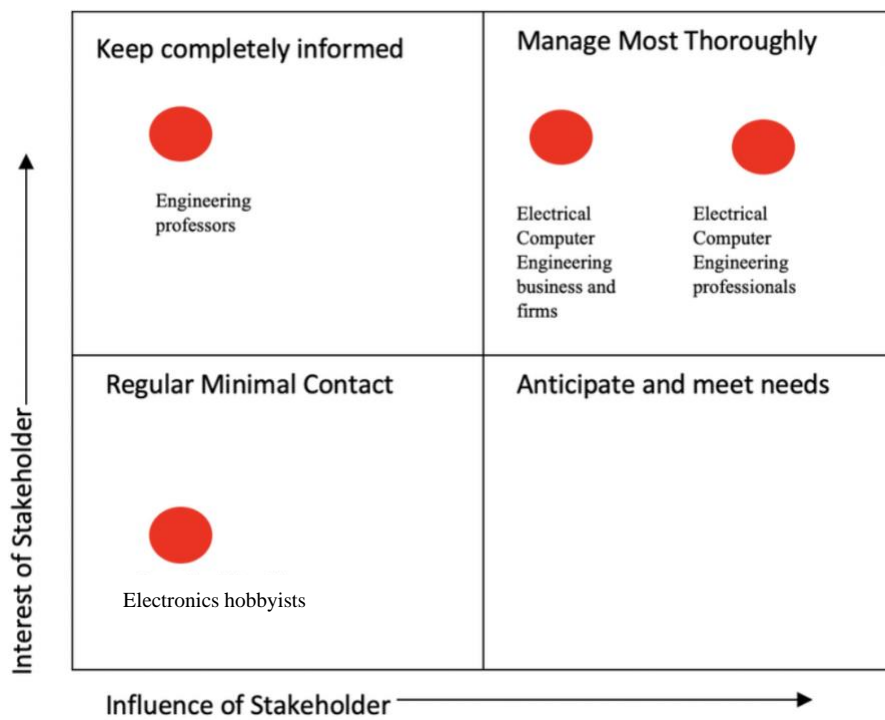
## Results

Although we were able to complete a simulated DPA attack, our results are not quite we had in mind when we started this project. We expected to be able to successfully recover the key used by our simulation, however our power data always proved to be too noisy to accomplish this. Furthermore, as this issue persisted even after we rewrote our AES implementation to use a single data register, and even used an established standard implementation, we were able to determine that the issue does not lie in our implementation of AES. Theoretically, this could mean that DPA attacks do not work on AES encryption, however based on tests we were able to run on a DPA attack on AES with real-time data, we know this to not be the case. What remains is the accuracy of our power model. We believe the issue could lie in the fact that Verilog's `$dumpvars` command also dumps some variables in a binary string instead of purely bit by bit, which could cause some issues with our model, and these issues could have caused our negative result. Of course, more testing would be needed to confirm these hypotheses.

**Stakeholders**

As our team gained a deeper understanding of our project topic and started producing final results, we were able to explore more of who the stakeholders could be. Our team decided to focus on four stakeholders who we think would be the most interested in our project's topic and results. The stakeholders we choose to focus on are electrical engineering business firms, engineering professors, electrical computer engineering professionals, and electronics hobbyists.

Figure 18 shows where we believe each of these falls within the stakeholder map.



*Figure 18.* Stakeholder Map

According to the map our team was able to also fill out a stakeholder analysis matrix to gain a better understanding of all of the stakeholders. In this map and matrix (Table 1), we have evaluated each of the stakeholders to see how the project and product should be aimed around their needs.

Table 1

Stakeholder Table

Stakeholder	Power (ability to stop or change project)	Interest	Type and frequency of communication
Professionals	High power to improve or change project	Very interested	Frequent communication
Professors	No power to change the project	Very interested	Not constant communication
Hobbists	No power to change the project	Main interest is studying it for fun	Do not need constant communication
Firms	High power to improve or change project	Very Interested	Frequent communication

To start our analysis, we examine how professors could teach topics like side channel attacks as part of ECE classes or projects. This project helps explain and show deeper understanding of DPA attacks providing some general content that professors can use to teach students about DPA attack. We think the results of our project could also be used in middle school or high school STEM curriculum.

The next stakeholder we examined was electrical engineering businesses. We believe that businesses that work with devices that could be affected by DPA attacks are an important stakeholder. This is due to the fact that they could use our project and code to test and understand DPA attacks and help protect their products against the attacks. We believe that they have high power to improve and change the project because they can adjust and change the project to work best for their devices. Our project is more like a skeleton for them to use and understand allowing them to adjust to their own projects. We believe they have high interest, and we should have frequent communication with them. This is due to the fact that we want them to understand and use our product for good and not have any trouble using it. If we were not to help them, they

might end up using it wrong and in the long run that will hurt their business which is not a goal of ours.

The next stakeholder we examined was electrical computer engineering professionals. This project can help them gain a better understanding of the attacks. We believe that they are able to adjust how the project works and the code to improve the project and format it to fit their own personal needs. Our team also believes that these people can use our project to dive into an even deeper analysis on DPA attacks than we were able to. The findings from our project can be used as a starting point to a more intense project working on protecting against DPA attacks. Our project will allow them to start with a base knowledge and code of the topic, so they are able to skip the 'set up' steps in the process expediting their research.

The last stakeholder we examined was electronics hobbyists. We believe they have very low power and low interest in our findings. This is due to the fact they might be looking into the project to learn something new. We would love to help them a bit, but we would not see a need for constant communication with them. Our project knowledge might also spark interest in topics they previously were not interested in learning.

### **Limitations**

As with any project, there were some limitations to our findings. Our team experienced several obstacles that impacted our findings and scope of work and ran into many different problems when starting this project. Some of our problems were technical issues and some of our issues had to do with timing and lack of resources. When we created this project, we discovered that simulators of this type do not widely exist. We had to come up with everything on our own and found that we had lots of technical issues.

One of the biggest limitations and issues our team ran into when creating the simulator was the environment, since we were trying to create a perfect simulation. When we developed the simulator, we found that we needed to run ModelSim on a Linux interface. To do this we ended up having to set up a virtual machine to run a Linux set up with a ModelSim and to run and test our simulator. We recommend the best way to adapt our simulator for the most ideal use in businesses is by creating a simulator that does not need such a controlled environment. Not all businesses will have the capability to run and set up a simulator that is so difficult to set up so if that was changed in the future more companies and businesses would be able to use our product.

Another limitation that our team ran into was with technical issues when setting up our simulator. We had issues with the simulations not running correctly due to settings on the simulation not being correct. To fix this our team attempted to remake the project and readd the files. Our team was able to solve this issue by adding files to the project directory.

Another limitation our team ran into was with the VCD generation. We tried the VCD commands and those did not work for us. What happened was the code seemed to function but there was no output given. We could not generate the WLF file from VCD. Our solution to this problem was by using the \$dumpfile instead. Some other issues we ran into when creating the VCD was that ModelSim did not process string literals and we had simple syntax errors we needed to fix within the code.

Despite these limitations, we think that future projects could look into our code and go off of what our team created to avoid running into the problems we found. Our team believes that if others end up looking at our code files, they will be able to avoid the basic set up errors we ran into. This would allow business that wanted to repurpose our product to save some money by using our code to start off instead of having to pay someone to do everything from the start.

## **Future Research**

As our team reflected on our project, we identified different ways to improve this project with future research. One of the main categories of improvement would be on the business front by creating advertisements and guides. Another category of improvement would be improving the technical aspects of the project.

Our team believes there are some important business aspects to add to this project. The first being a way to advertise our product and simulator to companies. An example advertisement is attached in Appendix A. Advertisement is important to get interest in companies making them want to implement our simulator within the company. Without adding an advertising feature to future projects, the research done will not be shown to companies. Another part of the project we think is important to create for future research is a step-by-step guide for companies on how to use the product. A simple layout for one of these guides is attached in Appendix B. This guide allows companies to troubleshoot on why the simulator might not be working for them. This guide would also streamline the implementation process of our simulator within the company. Our team has also created a layout for showing companies how easy the simulation is to use. This is helpful to get companies to realize that the product would be easy to implement within their company.

Our team also believes that there are some ways to improve the technical aspect of our product. The main improvement would involve finding a way to create the simulator to just be ran by the press of a button. Currently the simulator needs a specific environment to run which is not ideal for companies because they might not be able to set up that environment. If this project was created to just be a software package that companies could download and run it would gain

interest in more companies. Another important aspect future projects could look into is adding the ability to test for other hardware vulnerabilities within the software package.

### **Conclusion**

Devices are potentially vulnerable to physical data leakage. Our team developed a simulator, using ModelSim, to simulate the operations of a device then we ran a side-channel-based attack on the data from the simulation. This simulator can be used with business to allow them to save money by making sure none of their devices are vulnerable to attacks. This can also be used within businesses to help them create a guarantee that their devices have minimal vulnerabilities to DPA attacks.

### References

- Batina, L., & Robshaw, M. (2014). Cryptographic hardware and embedded systems 16th international workshop; proceedings. Heidelberg: Springer
- Cai, X., Li, R., Kuang, S., & Tan, J. (2020). An Energy Trace Compression Method for Differential Power Analysis Attack. *IEEE Access*, 8, 89084–89092.  
<https://doi.org/10.1109/ACCESS.2020.2993701>
- Chen Z., Zhou Y. (2006) Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In: Goubin L., Matsui M. (eds) Cryptographic Hardware and Embedded Systems - CHES 2006. CHES 2006. Lecture Notes in Computer Science, vol 4249. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/11894063\\_20](https://doi.org/10.1007/11894063_20)
- Daemen, J., Rijmen, V.: Rijndael for AES. In: AES Candidate Conference, pp. 343–348 (2000)
- Greenberg, A. (2020). What Is a Side Channel Attack? Retrieved October 17, 2020, from <https://www.wired.com/story/what-is-side-channel-attack/>
- Kocher, P., Jaffe, J., Jun, B., & Rohatgi, P. (2011). Introduction to differential power analysis. *Journal Of Cryptographic Engineering*, 1(1), 5-27. doi: 10.1007/s13389-011-0006-y
- Mai K. (2012) Side Channel Attacks and Countermeasures. In: Tehranipoor M., Wang C. (eds) Introduction to Hardware Security and Trust. Springer, New York, NY.  
[https://doi.org/10.1007/978-1-4419-8080-9\\_8](https://doi.org/10.1007/978-1-4419-8080-9_8)
- Mangard, S., Oswald, E., & Popp, T. (2007). Power Analysis Attacks: Revealing the Secrets of Smart Cards (1. Aufl.). Springer-Verlag. <https://doi.org/10.1007/978-0-387-38162-6>
- Neustadter, D. (2018, October 08). Using threat models and risk assessments to define device security requirements. Retrieved May 03, 2021, from



<https://www.techdesignforums.com/practice/technique/using-threat-models-and-risk-assessments-to-define-device-security-requirements/>

Peeters, E. (2013). *Advanced DPA Theory and Practice Towards the Security Limits of Secure Embedded Circuits*. New York, NY: Springer New York.

Pellegrini, M. (2009, March 18). Differential power analysis [Self made diagram of differential power analysis.]. Retrieved from [https://commons.wikimedia.org/wiki/File:Differential\\_power\\_analysis.svg](https://commons.wikimedia.org/wiki/File:Differential_power_analysis.svg)

Randolph, M., Diehl, W. (2020). Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. *Cryptography*, 4(2), 15. doi:10.3390/cryptography4020015

Shanmugam, D., Selvam, R., & Annadurai, S. (2014). Differential Power Analysis Attack on SIMON and LED Block Ciphers. *Security, Privacy, and Applied Cryptography Engineering*, 110–125. [https://doi.org/10.1007/978-3-319-12060-7\\_8](https://doi.org/10.1007/978-3-319-12060-7_8)

Side-channel attacks: How differential power analysis (dpa) and simple power analysis (spa) works. (2017, September 27). Retrieved February 10, 2021, from <https://anysilicon.com/side-channel-attacks-differential-power-analysis-dpa-simple-power-analysis-spa-works/>

Simon, B. (2016, November 26). What is stakeholder analysis and mapping and how do you do it effectively? Retrieved May 03, 2021, from <https://www.smartsheet.com/what-stakeholder-analysis-and-mapping-and-how-do-you-do-it-effectively>

Skorobogatov, S (2011). *Side-channel attacks: new directions and horizons* [https://www.cl.cam.ac.uk/~sps32/ECRYPT2011\\_2.pdf](https://www.cl.cam.ac.uk/~sps32/ECRYPT2011_2.pdf)

S. Mangard, E. Oswald and F. - . Standaert, "One for all - all for one: unifying standard differential power analysis attacks," in IET Information Security, vol. 5, no. 2, pp. 100-110, June 2011, doi: 10.1049/iet-ifs.2010.0096.

Velegalati, R., & Yalla, P. (2008). Differential Power Analysis Attack on FPGA Implementation of AES (Unpublished doctoral dissertation). George Mason University.

Ye, X. (2015). Side Channel Leakage Analysis - Detection, Exploitation and Quantification.

Retrieved from <https://digitalcommons.wpi.edu/etd-dissertations/47>

Appendix A



PROTECT YOUR PRODUCTS FROM BEING ATTACKED

# HARDWARE VULNERABILITY TOOL

---

A simulator that shows you if your device is vulnerable to differential power analysis attacks

## Appendix B

### How To Use The Tool

Page 1: What is this tool?

This tool was created for your use to detect vulnerabilities in your hardware. This simulator uses ModelSim and Python to simulate an attack on hardware to find if it is vulnerable. This attack will show you if you need to improve your hardware, so it is not vulnerable to these types of attacks.

Page 2: How does it work?

In this section you will want to explain exactly how the product works. You do not need to have step by step instructions here, but you should explain what happens during the process. This part will ease people's concern about if the simulator is safe.

## Page 3-4: Steps to implement the tool

## HOW TO USE THE HARDWARE TOOL

---

- 1 How to download**

Our product is created to be downloaded as a package. All you will have to do is download from the website and then move onto the next step.
- 2 How to run**

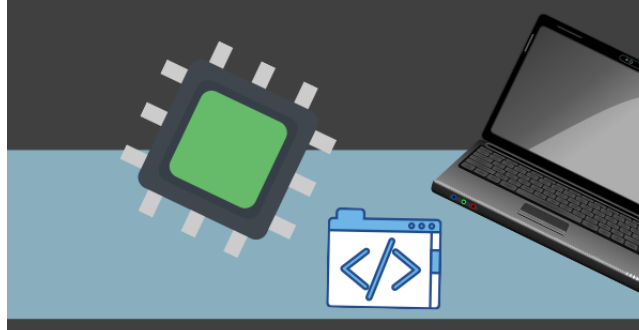
Our product has an easy install process and all you will have to do is follow the instructions that are shown along the way.
- 3 How to create simulation**

When running our program you will have the options to create a simulation of your new hardware device. The nice thing about our product is that it will show you the steps along the way on how to set up a custom simulation.
- 4 How to test your product**

After setting up your simulation you will have the opportunity to test it along the remade attacks. Once your simulation is set up you just need to switch to testing mode and follow those instructions.
- 5 How to read results**

Our product is created to make the results reading easy! Once the device testing is complete our program will show you your results. It will tell you if your device is vulnerable to attacks so you do not have to worry about reading results wrong.
- 6 Next steps**

If your system is vulnerable to attacks we suggest you look into differnt ways to fix the problem. Our product will show some common solutions but each product is unique so we are not able to fully confirm the best ways to make your device not vulnerable.

An illustration at the bottom of the page shows a green microchip with pins, a laptop, and a code editor icon with blue and white symbols.

#### Page 6: Common errors

In this section it is important to discuss what common errors might happen during the set-up process. You will also want to explain different ways to fix these common errors. This allows the customer to know ways to fix the simulator without needing to contact help right away.

#### Page 7: Contact

In this section you would want to put a way to contact you to ask for help if they come across errors that are not listed in the common errors section.

#### Page 8: Resources

In this section you will want to include resources on these types of attacks and ways to protect against them. This section allows the customer to do more research on the topic if interested.

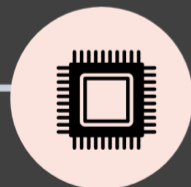
# Easy Implementation



Download  
Our Tool



Follow  
instructions  
to set up  
simulation



Test



Protect

a