2023-08-11

SIDE-CHANNEL LEAKAGE FROM THE ADVANCED ENCRYPTION STANDARD ON FIELD PROGRAMMABLE GATE ARRAYS

Kaden Makechnie-Hardy

Honors Thesis

SIDE-CHANNEL LEAKAGE FROM THE ADVANCED ENCRYPTION STANDARD ON FIELD PROGRAMMABLE GATE ARRAYS

By
Kaden Makechnie-Hardy

Submitted to Brigham Young University in partial fulfillment
of graduation requirements for University Honors

Electrical and Computer Engineering
Brigham Young University
August 2023

Advisor: Jeff Goaders

Reader: Scott Lloyd

Honors Coordinator: Derek Hansen

ABSTRACT

SIDE-CHANNEL LEAKAGE FROM THE ADVANCED ENCRYPTION STANDARD ON FIELD PROGRAMMABLE GATE ARRAYS

Kaden Makechnie-Hardy

Electrical and Computer Engineering

Bachelor of Science

As field programmable gate arrays (FPGAs) perform increasingly critical cryptography, researchers study the safety of these computations. One such compromise on the safety of cryptography is the side-channel attack. These attacks exploit sources of information called leakage that escape a cryptographic system in ways the original design did not intend. One dimension of a successful side-channel attack is the collection of leakage. This research configures the Advanced Encryption Standard (AES) on an FPGA and demonstrates strategies for collecting three types of side-channel leakage: power consumption, temperature emissions, and ring oscillations.

ACKNOWLEDGMENTS

Thank you to Dr. Scott Lloyd for diligently coaching and mentoring my capstone team. Thank you to Dr. Albert Tay for reading my drafts and helping me clarify my writing. Thank you to Jacob Burtenshaw, Sara Chapman, Kepa Zubeldia, Alek Farmer, and Noah Hanks for their persistence in developing tools, putting the cryptography core onto the programmable logic, and helping me dive into the world of hardware. Thank you to Dr. Derek Hansen for mentoring and guiding my undergraduate experience. Thank you to Dr. Jeff Goaders for advising this project.

Table of Contents

List of Tables and Figures

I. Introduction

Cryptography is the study of secret communication, often in the presence of adversaries. Cryptography is a foundation for trust that underpins critical systems and day-to-day activities. Things from global financial markets to text messaging depend on trusted cryptography. Anciently, cryptography shifted letters in war messages to preclude enemy spying [1]. These ciphers are simple enough that kids use them for fun. The algorithmic specification of this kind of cryptography can be as simple as "shift each letter three spots forward in the alphabet." For example, the input message "abc" becomes the encoded message "def". A more complex input like "attack at dawn" becomes "dwwdfn dw gdzq". This cryptographic algorithm is called Caesar's Cipher. While cryptography has historically aided war efforts, modern cryptography propagates into healthcare, education, transportation, and other corners of our lives. Increasingly, we trust cryptography to secure our communication, navigation, diaries, photos, medical histories, banking information, and digital lives.

Today, cryptography requires the expertise of mathematicians, computer scientists, and computer engineers. Indeed, most cryptography is performed by computers at speeds an ancient general would envy. While the specification of Caesar's cipher takes one sentence, modern cryptographic algorithm specifications can take dozens of pages and require years of research. These algorithms are formal specifications of the precise steps necessary to convert plaintext into ciphertext (encryption) and ciphertext into plaintext (decryption). While mathematicians have verified these cryptographic algorithms, the formal specification is different from the implementation. Consider the relationship between a recipe and the product of the recipe. A

culinary expert may specify exact instructions for combining and preparing ingredients with a well-tested recipe. However, the recipe guarantees little about what happens in the kitchen.

Real implementations of cryptographic algorithms often introduce vulnerabilities unrelated to the formal specification. These vulnerabilities exploit side channels like sound, temperature, electromagnetic emanations, power consumption, and others. “A side channel is a potential source of information flow from a physical system to an adversary, beyond what is available via its abstract model” [2]. Another word for side-channel leakage is physical leakage information (PLI) [3]. Sometimes, these side channels even reveal secret details about the algorithm’s state. That may be enough to compromise the system’s security, giving adversaries access to otherwise encrypted secrets. A side-channel attack happens when adversaries exploit flaws in the implementation (not the original algorithm) and reveal secrets.

Rightfully, researchers study, verify, and validate cryptographic algorithms. Mathematicians and cryptographers at universities, research institutes, governments, and businesses work together to formally specify correct cryptographic algorithms (recipes). Then, implementors create software and hardware (kitchens) where the steps in the algorithm to produce plaintext and ciphertext are followed. Implementors put cryptography into television remotes, social messaging apps, keyless door locks, digital signature software, and many other places. In some places, packaged software implements the steps of cryptographic algorithms. In other places, dedicated instructions on a CPU implement the steps of cryptographic algorithms.

This research investigates side-channel leakage from one implementation of the Advanced Encryption Standard (AES) on a Field Programmable Gate Array (FPGA). This implementation is an open-source contribution by Homer Hsing [4]. Other researchers have

demonstrated side-channel attacks and analysis on FPGAs against other implementations of AES [5], [6], [7], [8]. The novel contribution of our research is in studying the unique combination of this AES implementation and the PYNQ-Z2 FPGA. Our findings indicate the importance of AES implementation selection and the influence implementation details have on side channels.

II. Definitions

We will discuss what AES and FPGA mean in turn. In 2001, the National Institute of Standards and Technology specified AES in FIPS 197 [9]. AES is a symmetric cipher, which means the key for encryption is the same as the key for decryption. To start encryption, AES requires two inputs: a plain text message and a key. At the end of encryption, the algorithm has produced ciphertext. Symmetrically, AES takes ciphertext and uses the same secret key to calculate the plain text. This key may be 128, 192, or 256 bits long, which requires 10, 12, or 14 rounds for computation, respectively. A round represents a repeatable unit of steps in the algorithm. Longer keys are more secure, and adding more rounds requires more computing time. AES is a block cipher, which means it operates on a fixed length of bits called a block. With AES, the block is always 128 bits long, regardless of key size. Thus, input and output are always 128 bits long. The word size is 32 bits. Word size is another useful grouping of bits like bytes or nibbles. Figure 1 is John Savard's visual representation of AES.

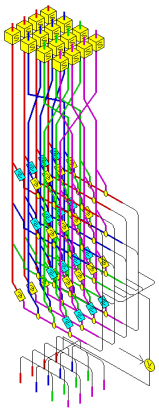

Figure 1
John Savard's Representation of AES

The block, or the state, is where all operations happen. Encryption performs operations called SubBytes, ShiftRows, MixColumns, and AddRoundKey on the state. Figure 1 illustrates how the state changes through each of these four operations. Decryption performs operations called InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey. These methods have different properties in different implementations while producing the same output. These are the steps that side-channel attacks try to reveal. Understanding these methods is important because abstracting these methods helps researchers build models of expected side-channel leakage to compare against real side-channel leakage. This is an essential early step in exploiting side-channel leakage and accomplishing a side-channel attack.

FPGAs are reprogrammable hardware that approach the performance efficiency of dedicated hardware without the upfront design and manufacturing costs of dedicated hardware. We placed AES on the FPGA pictured in Figure 2.

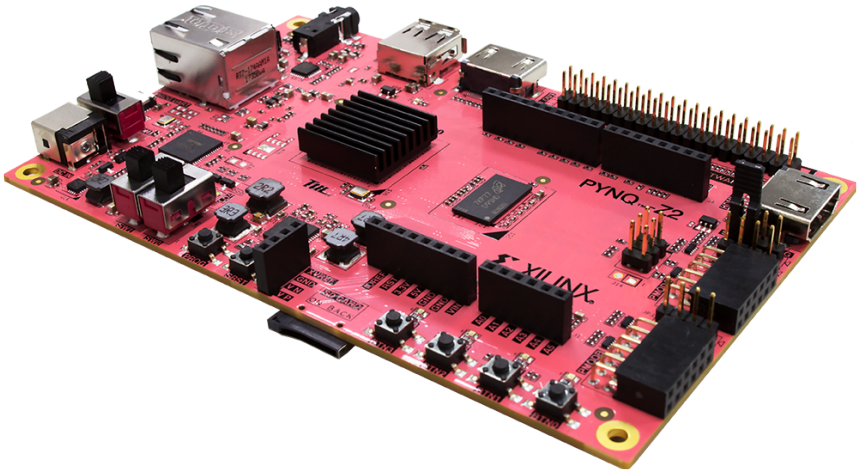

Figure 2
PYNQ-Z2 Board Featuring a ZYNQ Field Programmable Gate Array (FPGA)

FPGAs are relatively inexpensive, easy to reprogram, and power efficient. FPGAs are particularly popular for cryptography because they offer the performance benefits of dedicated hardware and the reprogrammable flexibility of software development. FPGAs can be reprogrammed to implement arbitrarily complex devices, even as complex as microprocessors [10]. The reprogrammable part is called the programmable logic (PL) or the fabric. Figure 3 shows one example of programmable fabric.

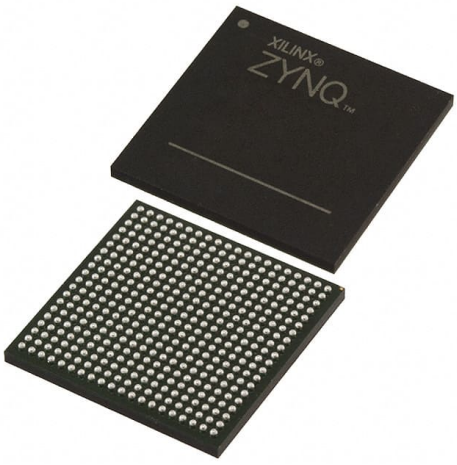

Figure 3
ZYNQ XC7Z020-1CLG400C SoC Featuring Dual ARM® Cortex®-A9 MPCore™ and Zynq®-7000 Artix™-7 FPGA

When a designer takes a high-level representation of behavior, like a cryptography algorithm, and FPGA tooling converts it to low-level functionality, we call that synthesis [10]. Synthesis creates a netlist, representing connections between logic gates [11]. On the fabric, those logic gates are represented by lookup tables, which map certain inputs to certain outputs [10]. These are the logical operations that side-channel analysis attempts to observe. These operations each consume certain amounts of power, emit certain amounts of heat, and cause certain electromagnetic emanations. A careful understanding of how these gates represent the high-level steps of the algorithm is the first step toward performing a bona fide side-channel attack.

Many computers are sequential instruction executors. In contrast, FPGAs have no concept of sequential instruction execution [10]. This makes precise control of input and output simpler than with other types of computers [12]. Implicit assumptions in FPGA specification may inadvertently influence implementation [11]. Without sequential instruction execution, information leaked through the side channels comes from combinatorial logic—not sequential, individual instructions.

Why would a designer work with FPGAs instead of other embedded computers like Application-specific Integrated Circuits (ASICs), as shown in Figure 4? On one hand, ASICs are great for specialty computing needs. However, ASICs are not reprogrammable. Per unit, ASICs are cheaper to fabricate than FPGAs but the upfront costs of ASICs can be large compared to FPGAs. While the per-unit cost of an FPGA is higher than an ASIC, an individual can get started with FPGA development for the cost of a board. It can also be costly to mass-produce ASICs and later realize a flaw in the design. These flaws are trivial to fix on FPGAs and impossible to fix on

ASICs. FPGAs are great for prototype and production embedded systems where ASICs are too expensive [10]. However, FPGAs have expanded to mainstream production because FPGAs deliver faster time-to-market and easy reprogrammability [11]. Figure 5 shows the cost-to-scale relationship between FPGAs and ASICs.

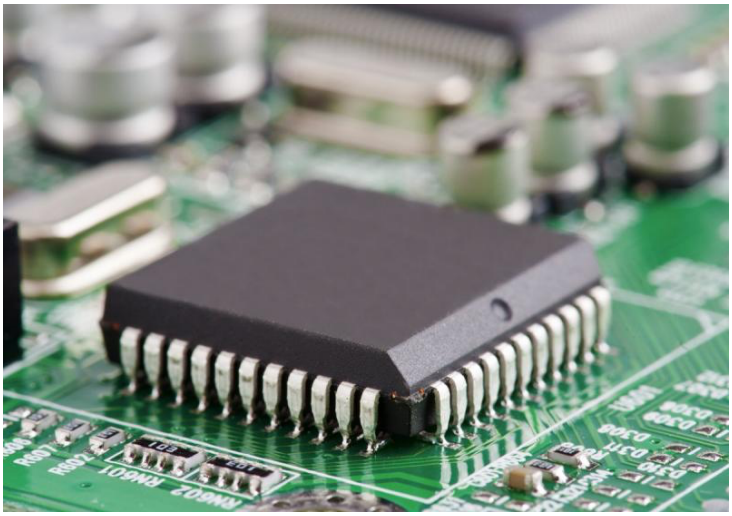

Figure 4
Application Specific Integrated Circuit (ASIC)

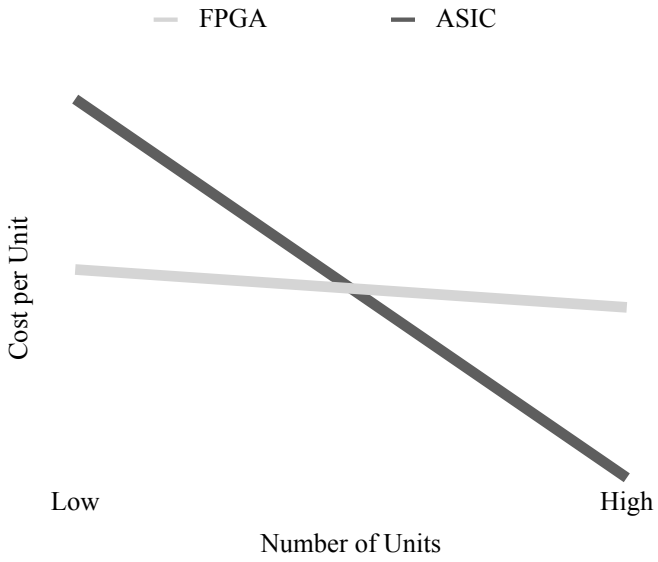

Figure 5
Cost Vs. Scale of FPGAs and ASICs

III. Literature Review

One of the earliest discovered and simplest forms of side-channel leakage comes from measuring how long an operation takes. In other words, the adversary extracts secret information about the state of the cryptographic algorithm by precisely measuring the timing of known computations. Adversaries measure how long certain inputs take to calculate certain outputs. With enough repetition, adversaries build models correlating mere time measurements with secrets, then they perform a timing attack. Before it was discovered that timing information could reveal secrets, cryptography designers tried to make their algorithms operate as efficiently as possible [2]. However, well-defined algorithms with clear timing expectations exacerbate vulnerabilities to timing attacks. That led to one defense against timing attacks, which inserts random or defined idle time (padding) into the execution of the steps of the algorithm [2]. If the adversary can only view the timing of input and output operations, padding is enough to stop the adversary from deducing what is being computed. However, delaying output consumes less power, which appears on a power trace [13]. This led security researchers to explore measuring power and timing information together. This led to the simple power analysis side-channel attack [2], pioneered by Paul Kocher.

Simple Power Analysis (SPA) is a devastating, strong side-channel attack [2]. SPA derives information about the secret key from the shape of the power signal during cryptographic execution [2]. This is achieved by sampling the power signal 10 to 100 times per clock cycle [2]. This sampling produces a power trace, which represents the side-channel leakage. Why are precisely timed, high-resolution power measurements useful? Different operations have different power consumptions. For example, a floating-point operation may consume more power than an

integer operation or a processor stall from a cache miss may consume less power than active computation [13]. Adding 3 and 4 consumes slightly different power than adding 3 and 3. Most revealing, "[t]he power consumption of a cryptographic operation often depends on a secret key value" [13]. In summary, SPA captures and exploits the unique power consumption of atomic steps in a cryptographic algorithm. Paul Kocher developed this kind of attack in 1998. Figure 6 shows one of Kocher's original power traces.

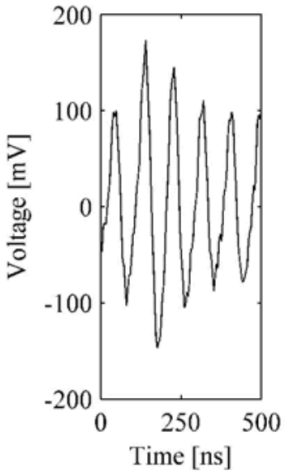

Figure 6
Sample Power Trace

Mangard, Oswald, and Popp wrote, "power analysis attacks exploit the fact that the instantaneous power consumption of a cryptographic device depends on the data it processes and on the operation it performs" [14]. In other words, the peaks and troughs in Figure 6 represent information an attacker can use to reveal information about data and operations [13]. These traces show the power drawn by boards like those pictured in Figure 2.

The next level of sophistication is Differential Power Analysis (DPA), which statistically analyses many power traces instead of just one [2]. Hypothesis testing against side-channel signals is a core step in this type of side-channel attack [2].

Physical access is not necessary for side-channel attacks. Traditional side-channel attacks connect a low-impedance resistor in series with the power supply and measure power consumption as the voltage drops across the resistor. However, researchers have recently built on-fabric power monitors from ring oscillators [13]. These ring oscillators have enough resolution and sensitivity to measure the power consumption of other modules on the FPGA fabric (including other software programs running on the microprocessor) [13]. The ring oscillator frequency is sensitive to variations in voltage, temperature, and even what process is running. In other words, the time it takes for the signal to propagate around the loop directly depends on what is running nearby. It is no longer necessary to have physical access to a board to analyze its side-channel leakage. One impetus for this research is the problem of multi-tenant FPGAs in the cloud, where one tenant may run a ring oscillator near another tenant's legitimate computing. However, many cloud providers prohibit these kinds of combinatorial circuits from running on multi-tenant FPGAs.

Researchers have long exploited side-channel attacks to reveal cryptographic secrets. Researchers are even beginning to exploit side-channel attacks to reverse engineer the architecture of confidential intellectual property (IP) [15]. (IP is a catch-all term for pre-packaged designs available for licensing.) Some of the most important research in side-channel attacks happens in developing countermeasures that obscure the inner workings of cryptographic IP cores [3].

Side-channel attacks are a popular research area because they present a difficult security problem for accelerated computing. How can one secure what does not exist in the "abstract model" of a computing system [2]? Two broad types of countermeasures include masking and

hiding. Masking obscures the PLI by introducing randomness to prevent an attacker from associating PLI with the models of the computation they are attacking [16]. Hiding renders the PLI uniform or random to make useful information indistinguishable [3]. Within hiding, there are also two types. Vertical hiding influences the amplitude of PLI data and horizontal hiding influences the timing of PLI data [3]. The graph of PLI is called a leakage model or leakage function [17].

Early developments in side-channel attacks were effective at cracking small keys of 8 or 16 bits [17]. However, with larger keys like AES's 128, 192, or 256-bit key, side-channel attacks require more sophisticated approaches. The divide and conquer strategy divides the key into sub-keys and recovers sub-keys before combining the sub-keys to reveal the key. The primary enemies of key recovery with non-trivial key sizes are noise and error. Therefore, reliable key recovery requires large sample sizes (where individual samples are traces containing thousands of trace measurements) and data-driven statistical methods [17].

Another way to categorize side-channel attacks is into profiled and non-profiled. Attackers calculate the Hamming Weight and Hamming Distance to perform non-profiled attacks. These are called non-profiled because the attacker has no chance to train on the target device. Conversely, profiled attacks exploit data-driven model training to divide and conquer. Profiling is expensive, and it cannot reasonably recover an AES key without building precise leakage models [17].

The essence of side-channel analysis is measuring leakage over an interesting period, like during a cryptographic operation [18].

IV. Methods

Our research investigates an AES-256 implementation running on the ZYNQ SoC on a PYNQ-Z2 board. This board is popular for educational development because it is an open-source project that exposes a Python API for the Xilinx platform [19]. Most useful for this research, this board offers general-purpose input and output (GPIO) pins, options for SD card booting, and a USB connection (used to open a teletype).

For our AES implementation, we chose tiny_aes [4]. Homer Hsing wrote tiny_aes in 2012 with an emphasis on high speed as demonstrated by his highly parallel, synchronous implementation. His design sends one 128-bit block through one round of encryption every clock cycle, for all 14 rounds of AES-256. This core encrypts constantly, so we built a wrapper around it that flags when valid encrypted data has reached the end of round 14. One implication of this design choice is that it seems impossible to deduce which discrete step of encryption the state of the side channels represents when each round of encryption is running each clock cycle (working on 128 bits at a time). Compared with a sequential instruction executor that might step from round to round, our core is constantly performing all steps of all 14 rounds, continuously. At the beginning of our research, we did not realize how powerful this side-channel leakage obfuscation method would turn out to be.

As for resolution, let us describe some of the tooling we used. We strived to match or exceed the quality of data found in the literature review. The PYNQ-Z2's system clock is configured to run at 125 megahertz. To reach the minimum 10 samples per clock cycle, we must have an oscilloscope capable of reading 1.25 gigasamples per second. Our oscilloscope, the Keysight InfiniiVision DSOX2012A, recorded 2 gigasamples per second.

To facilitate easier side-channel leakage collection, we developed a series of helper methods accessible via the teletype to initiate and control encryption. Users of the cryptographic core may run the various methods pictured in Figure 7. These allow users to run tests on the core, run the core for a set amount of time, or run an on / off pattern (for example, 1 second on and 1 second off repeated 20 times).

```
Starting AES Application
Which tests would you like to run?
1: Hardcoded Test Vectors
            Secworks
            Ting AES HW
            Tiny AES Generated
            NIST Test Suite (KAT, MMT, MCT) NIST Known Answer Test
            NIST Multi-block Message Test
            NIST Monte Carlo Test
2: Repeated run (chose how long you would like to run for)
3: Duty cycle test (chose how long you want it on and off)
4: Duty cycle temperature test (record the temperature during duty
cycle)

Enter q to quit at any time

Input a number between 1 and 4 for the test you would like to run
Make sure to send a new line (Ctrl + j on putty)
```

Figure 7
Controller Application for Delivering Commands to Cryptography Core

We investigated three side channels: power, temperature, and a ring oscillator frequency. We selected these three side channels because they appeared commonly among recent side-channel analysis studies. These data represent individual traces, not the many traces necessary for DPA.

First, we accomplished power analysis by building the circuit pictured in Figure 8, which was patterned after the diagram in Figure 9. This circuit breaks open the power supply connection and inserts a shunt resistor between the board and the power source. To take measurements, we connected oscilloscope probes to both ends of the resistor. We patterned our design after a common exploit used on smart card readers developed by Paul Kocher and

explained by Mangard, Oswald, and Popp [14]. Each measurement takes place about ground truth, which is the definitive start of valid encryption. The ground truth signal is high during valid encryption and low when the core is encrypting whatever nonsense is left over. That ground truth signal is programmed to come out of the general-purpose input and output pins, as connected by the blue wire in Figure 8. With a two-channel oscilloscope, one may measure the ground truth and the power consumption concurrently.

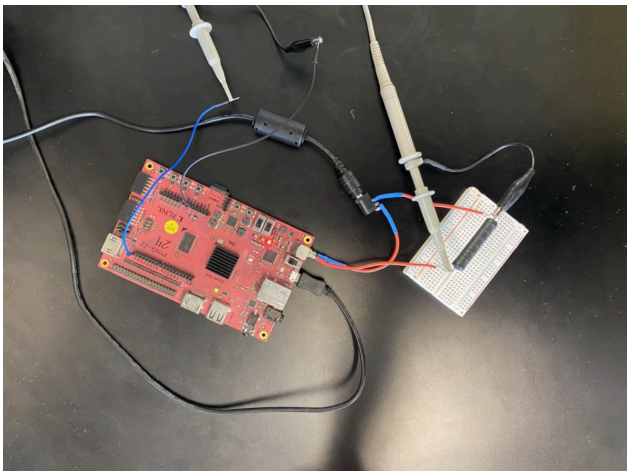

Figure 8
Oscilloscope Connected to Power Analysis Circuit and Ground Truth from FPGA

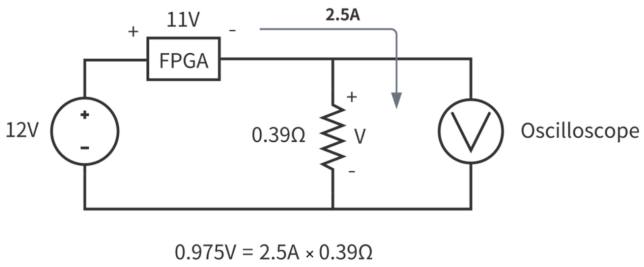

Figure 9
Power Analysis Circuit

One advantage of this circuit is its price performance. All of the parts can be purchased for less than a few American dollars. One disadvantage of this circuit is the quality of the resistor

and the precision of our calculations based on that resistor. However, with power analysis, the variation is what matters—not the absolute values. Therefore, while more precision may be desirable for the sake of high-precision data, it is not necessary. In addition, while Kocher measured and worked with millivolts, we converted the measured volts to amps based on the shunt resistor's resistance.

Second, we accomplished temperature collection through the XADC (Xilinx Analog-to-Digital Converter) wizard. This wizard is an interface for System Monitor, which sits on the ZYNQ system-on-a-chip (SoC). The SoC is divided into programmable logic (PL) and a processing system (PS). The processing system is the Arm core and the programmable logic is the FPGA fabric. In ZYNQ series FPGAs, the System Monitor sits on the PL side of the SoC. This means our temperature readings are coming from close to the cryptography core, which is also located on the PL. (In contrast, power consumption information is measured and generated spatially far from the cryptography core while being temporally sensitive to the slightest change in computation on the PL. Ring oscillations offer the benefits of both emanating directly from the PL and keeping close temporal sensitivity to minuscule changes on the PL. While power consumption and ring oscillation are highly sensitive to slight changes, temperature is a broad and imprecise measure.) The XADC Wizard, as shown in Figure 10, shows a representation of the IP Block responsible for transmitting temperature readings from the programmable logic.

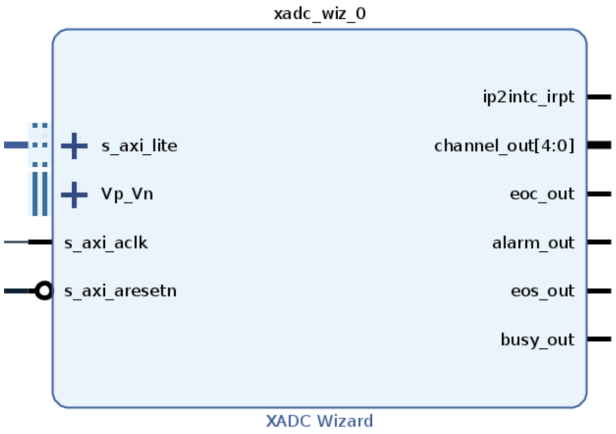

Figure 10
Xilinx Analog-To-Digital Converter Intellectual Property Block

Third, we accomplished ring oscillator analysis by adding a ring oscillator to the fabric. Ring oscillators are loops of inverters, also called NOT gates. Figure 11 shows what this loop of inverters may be conceptualized as. The ring oscillator implemented on the PL has 7 inverters. At this stage in the development of a ring-oscillation-based side-channel analysis, the only important consideration when deciding how many inverters to place in a ring oscillator is that the number must be odd. This is to ensure the inverters uniformly alternate from high to low and back again. Future researchers may increase or decrease the number of inverters to tune the frequency of oscillation. These looped inversions are sensitive to many kinds of changes, even as subtle as moving the board or changes in air movement. We found that jostling the board during measurements was enough to introduce changes in the ring oscillator period. The measurements taken below were recorded on a classroom-style lab table. For more precise ring oscillation measurements, it may be beneficial to experiment with dampening small structural vibrations from the building or even preventing regular HVAC airflow.

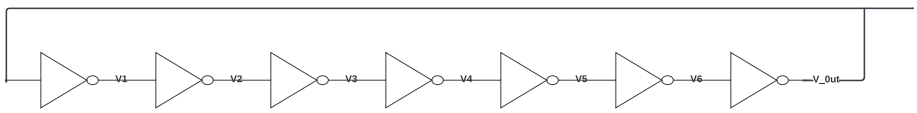

Figure 11
Ring Oscillator Diagram

Why do combinatorial loops produce ring oscillations in the first place? In a perfect computer, combinatorial logic (also called time-independent logic) instantly reflects the output of its inputs. With real computers, there is a slight delay as the input signals propagate to the outputs. A combinatorial loop then feeds those outputs back into the inputs, causing an infinite loop. This loop produces interesting electromagnetic emanations as the signal infinitely oscillates. It should be noted that combinatorial loops like this are not allowed by most cloud FPGA providers because the emanations from these loops are so useful in compromising the security of other tenants on shared hardware. FPGA development tools like Xilinx also prevent developers from placing a combinatorial loop on the PL. Therefore, it is important to override some of the default Xilinx restrictions by including a line that says `(*ALLOW_COMBINATIONAL_LOOPS = "TRUE" *)` in the ring oscillator module.

Once the ring oscillator was on the fabric, we sent its emanations out through one of the GPIO pins. We recorded the ring oscillations as a voltage trace with the oscilloscope, similar to the power trace. The ring oscillation data has a distinct period, so for ease of analysis, we converted the oscillating voltage data into periodic data. That process takes the sinusoidal oscillation trace and finds the time stamps of the zero-crossings.

V. Results

For the ring oscillator (RO), we found the period to change between 5 and 6 nanoseconds. Figure 12 shows this change. This oscillation is caused by a wide variety of electromagnetic emanations. It is difficult to correlate changes in the RO period with specific steps or computations in AES without more advanced analysis. For now, this research demonstrates the viability of placing a RO on the fabric near a cryptographic core and taking traces from the RO.

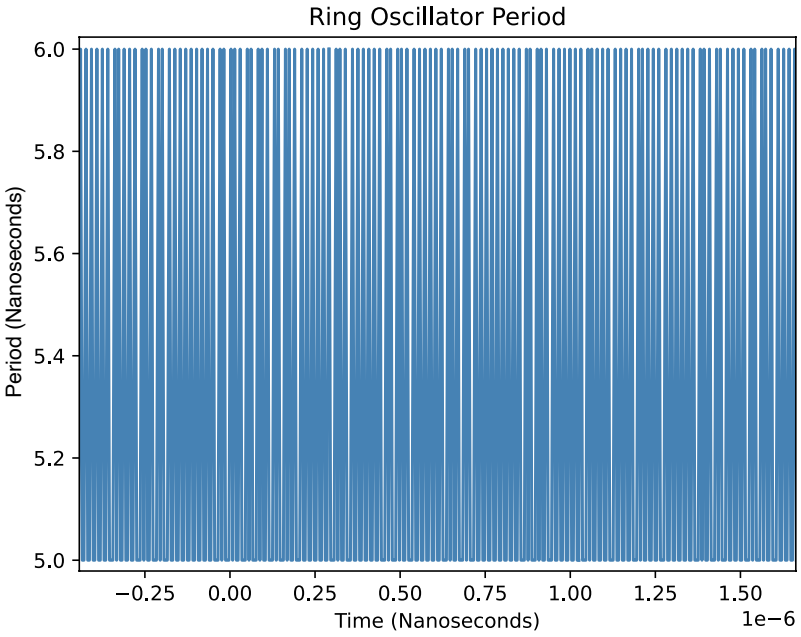

Figure 12
Ring Oscillator Trace

For the temperature, we found steadily rising temperatures as encryption continued (shown in Figure 13). However, the temperature trace is more imprecise and less sensitive than the power trace or the ring oscillator trace. Although it would be more formidable to steal keys or

text from even many temperature traces, this trace may give insights into how intensely the core is working or how long the core has been running.

One serious risk with access to temperature data happens when an attacker tries to force the board to run outside its safe operating temperature range and freeze the board [20]. This is one way to cause a denial-of-service attack, where legitimate computing is stopped.

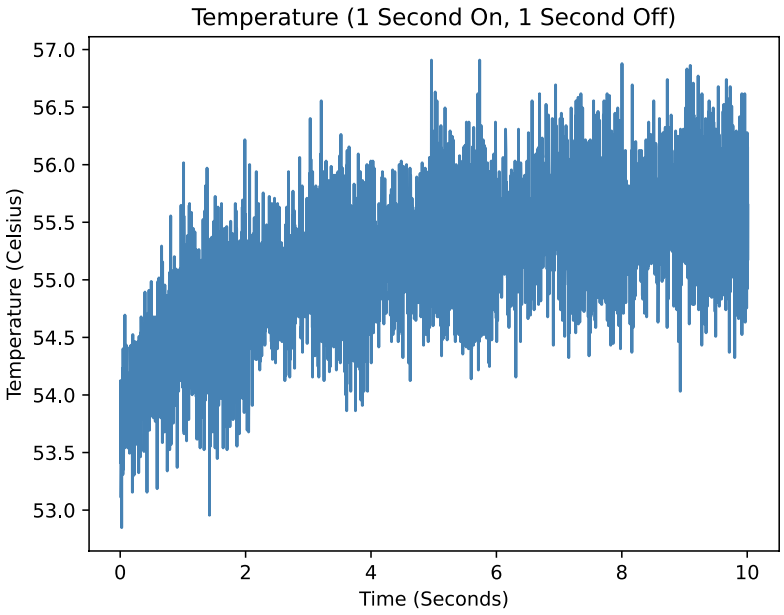

Figure 13
Temperature Trace

For power, we found increasing frequency in the current oscillations. The same data is shown in Figures 14 and 15 at different zoom levels. The darker color represents the power trace and the lighter color represents a ground truth marker of when valid encryption has begun. Because the core is constantly encrypting (ground truth only indicates valid data going into the core), we can only correlate the increased frequency with the valid inputs and outputs—not encryption generally. Some researchers have noticed a dip in power at the beginning of each AES round similar to the dips we see in these traces [17].

Additionally, it is difficult to separate this consumed power from the power consumed by other hungry devices. Other power-consumption-heavy logic components like the direct memory access (DMA) engine exist spatially near the cryptography core and add noise to the side-channel leakage. From the current data, it is too early to say if the increasing frequency we see is from the cryptography core, the DMA engine, or even the Arm microprocessor.

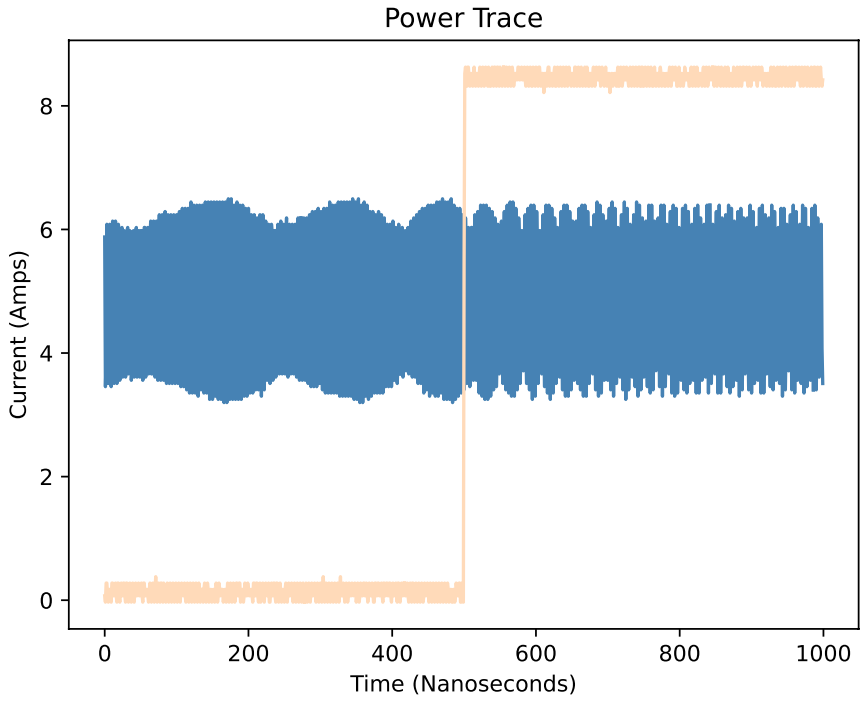

Figure 14
Power Trace

These ambiguities demonstrate the importance of differential power analysis (DPA), where statistical methods are employed to analyze a large sample size of traces. These methods allow researchers to separate the change in power caused by the cryptography core from the changes in power caused by other components on the board.

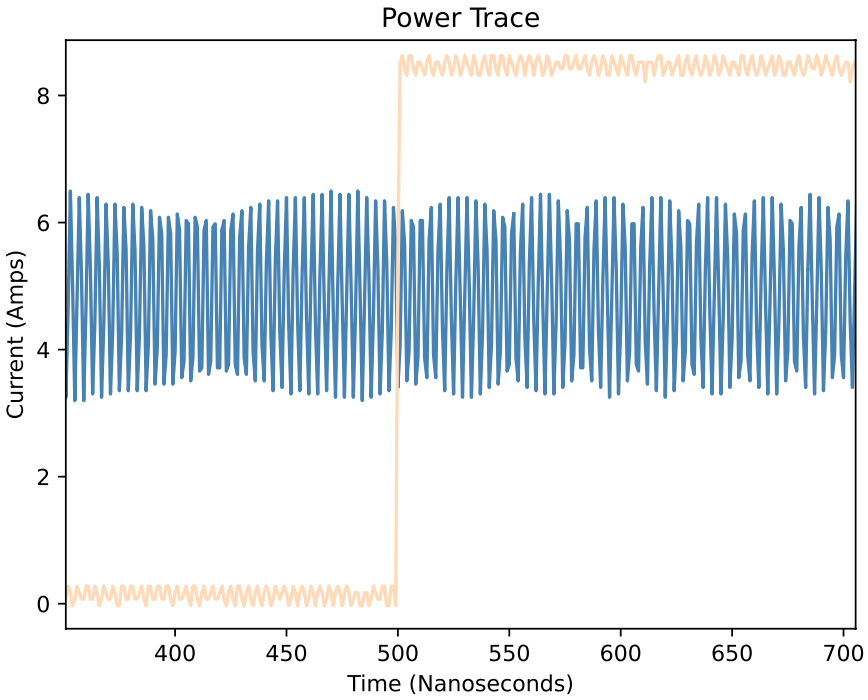

Figure 15
Power Trace (Zoomed)

VI. Key Findings

Our research has shown three strategies for collecting side-channel leakage from an AES cryptographic core running on a Pynq-Z2 FPGA. Collecting leakage information from the power, temperature, and ring oscillation side channels demonstrates the necessity of countermeasures and one perspective on the state of hardware security.

We found the resolution in our measurements to match that of other researchers—the fidelity was there. For example, for the two oscilloscope measurements (power and ring oscillator), our sample rate satisfied the literature expectation of 10 to 100 times per clock cycle.

We discovered certain dimensions of security that rely on the implementation details of the IP that are hard to know in advance of actually collecting that implementation's side-channel

leakage. For example, there are no obvious steps to encryption because this implementation does not use sequential instruction execution. Many researchers study side-channel leakage on sequential instruction executors. Our research is novel because it studies highly parallelized combinatorial logic. (The implication of this fundamental feature is that if one side-channel attacks a sequential instruction executor, one finds a different trace of side-channel leakage than if one attacks a combinatorial logic device. The traces collected in this research all demonstrate the latter characteristics of combinatorial logic gates changing state—not a sequential stream of instructions.)

We also learned that efforts to interface with IP (like the DMA engine or the program for controlling encryption) contribute their own leakage and it is difficult to discern the cryptographic leakage from the noise contributed by the other modules. These two traits of our system contribute to a higher level of security against side-channel attacks by introducing noise and a logic wall around the sensitive, cryptographic IP.

VII. Next Steps

One weakness of our ring oscillator collection strategy is that the GPIO may not be fast enough to reflect high-frequency oscillations. Future researchers may simply write oscillation data to a file.

An original aspiration of this research was to perform a real side-channel attack and recover cryptographic secrets. The two main types of DPA, correlation power analysis (CPA) and differential frequency analysis (DFA) have proven successful in many other research scenarios [21] and may be able to bring statistical rigor to the next steps in researching what is possible in side-channel analysis against combinatorial cryptography.

Another possible next step that takes us closer toward other research and away from studying the combinatorial logic would be selecting a cryptographic core that sequentially steps through each step in each round of encryption without performing a step in parallel. Although this is much slower and less secure, future research may benefit from collecting side-channel leakage from a cryptographic core that is less parallel than tiny_aes.

Any efforts to quiet the side-channel leakage of other components would also help researchers study the purer leakage of the IP under test. For example, our DMA controller is made up of 10 to 15 times more logic gates than the AES implementation. Side-channel leakage is just as much a product of these memory management routines as it is of the AES routines. The side-channel analysis would be a different story if we were collecting from an 8-bit microcontroller with a clock speed of 12 megahertz as opposed to our highly-parallelized 128-bits per clock cycle implementation running at 125 megahertz. The more isolated the steps for computation are, the easier it is to correlate leaked traces with expected values in the model. Our cryptography core works on 128-bit blocks in parallel each clock cycle.

Additionally, future researchers would ideally build a system to control the core such that the core can sit idly, consuming little power while not encrypting, and jump to higher power consumptions when valid encryption has begun. They should then build leakage models based on steps in the algorithm like SubBytes, ShiftRows, MixColumns, and AddRoundKey. Comparing these models with actual leakage models would be the beginning of the aforementioned differential side-channel analysis.

VIII. Conclusion

In many cases, preventing attackers from stealing side-channel leakage can be as simple as implementing physical security measures like placing the computer behind locked doors. In other cases, where physical security is lowered, hardware-level measures may be taken. Two popular ways to prevent side-channel leakage from spilling out of cloud FPGAs include disallowing combinatorial loops (to prevent someone from planting a ring oscillator) and building 'logic walls' around sensitive computation cores. These logic walls add noise and entropy to any trace a side channel may leak. However, with multi-tenant cloud FPGAs, it is still difficult to detect suspicious behavior, gauge the accuracy of suspicious behavior detection strategies, and even assign responsibility for scanning against suspicious behavior [22].

Some researchers have proposed other countermeasures like inserting dummy cycles (where the cryptography core performs no calculations) and false cycles (where the cryptography core operates on falsified data) between legitimate computations [23]. Other researchers have proposed countermeasures that, where possible, randomize the order of computation. This is called shuffling [24]. Both proposed countermeasures (and those discussed in the literature review) vitiate the leakage models attackers build for comparison against real trace data.

Broadly, we observed that countermeasures can be divided into two categories: ones that disturb the timing of trace measurements and ones that disturb the instantaneous trace measurements themselves. Our research found that having increased combinatorial logic executing in parallel within a clock cycle makes the instantaneous trace measurements difficult to tie back to any model for encryption. In addition to the parallelism within the cryptography

core, the direct memory access (DMA) controller helps obscure power consumption by noisily consuming power in parallel with the cryptography core.

Broadly, as programmers and computer scientists grow distant from the hardware that runs their software and algorithms, they risk exposing their otherwise good designs to hardware-level vulnerabilities. In other words, working at a higher level of abstraction makes it easy to forget what is happening in the hardware. It is not enough to find a good recipe. The kitchen must be sanitary and staffed by cleanliness-minded cooks. Just as humans are safer eating food from a kitchen that understands how germs spread and prevents that spread, computation is safer when its designers understand how hardware can be spied on through its side channels.

We recommend that cryptographers do not merely validate their algorithms but consider the implementations as well. For an algorithm to be safe, its hardware must be safe too. As Alan Kay said, “People who are really serious about software should make their own hardware” [25]. For those of us who cannot make our own hardware, writing good software requires mindfulness of best practices at the hardware level.

References

[1] S.-P. Oriyano, *Cryptography* (no. Book, Whole). McGraw-Hill Education, New York (in English), 2013.

[2] *Handbook of information security* (no. Book, Whole). John Wiley, Hoboken, N.J. (in English), 2006.

[3] J. S. Ng, J. Chen, K. S. Chong, J. S. Chang, and B. H. Gwee, "A Highly Secure FPGA-Based Dual-Hiding Asynchronous-Logic AES Accelerator Against Side-Channel Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 30, no. 9, pp. 1144-1157, 2022, doi: 10.1109/TVLSI.2022.3175180.

[4] *tiny_aes*. (2012). [Online]. Available: https://opencores.org/projects/tiny_aes

[5] H. Wang and E. Dubrova, "Tandem Deep Learning Side-Channel Attack Against FPGA Implementation of AES," in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 14-16 Dec. 2020 2020, pp. 147-150, doi: 10.1109/iSES50453.2020.00041.

[6] J. S. Ng *et al.*, "An Asynchronous-Logic Masked Advanced Encryption Standard (AES) Accelerator and its Side-Channel Attack Evaluations," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 27 May-1 June 2022 2022, pp. 2256-2260, doi: 10.1109/ISCAS48785.2022.9937684.

[7] M. Moraitis, M. Brisfors, E. Dubrova, N. Lindskog, and H. Englund, "A side-channel resistant implementation of AES combining clock randomization with duplication," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 21-25 May 2023 2023, pp. 1-5, doi: 10.1109/ISCAS46773.2023.10181621.

[8] R. Menicocci, A. Trifiletti, and F. Trotta, "A logic level countermeasure against CPA side channel attacks on AES," in *Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2013*, 20-22 June 2013 2013, pp. 403-407.

[9] L. Information Technology, *Announcing the Advanced Encryption Standard (AES)* (no. Book, Whole). Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD (in English), 2001.

[10] J. Ledin, *Architecting High-Performance Embedded Systems* (no. Book, Whole). Packt Publishing, Limited, Birmingham (in English), 2021.

[11] T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, Berlin, Heidelberg, J. R. Rao and B. Sunar, Eds., 2005// 2005: Springer Berlin Heidelberg, pp. 427-440.

[12] J. Rajewski, *Learning FPGAs*, First edition. ed. (no. Book, Whole). O'Reilly Media, Sebastopol, CA (in English), 2017.

[13] M. Zhao and G. E. Suh, "FPGA-Based Remote Power Side-Channel Attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, 20-24 May 2018 2018, pp. 229-244, doi: 10.1109/SP.2018.00049.

[14] S. Mangard, *Power analysis attacks* (no. Book, Whole). Springer, New York, N.Y. (in English), 2007.

[15] S. S. Ensan, K. Nagarajan, M. N. I. Khan, and S. Ghosh, "SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 29, no. 12, pp. 2040-2051, 2021, doi: 10.1109/TVLSI.2021.3110744.

[16] C. H. Gebotys, "A table masking countermeasure for low-energy secure embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 14, no. 7, pp. 740-753, 2006, doi: 10.1109/TVLSI.2006.878344.

[17] S. Jin and R. Bettati, "Efficient side-channel attacks beyond divide-and-conquer strategy," *Computer Networks,* vol. 198, p. 108409, 2021/10/24/ 2021, doi: https://doi.org/10.1016/j.comnet.2021.108409.

[18] J. Assaeedi and H. Alsuwat, "Side-Channel Attacks Detection Methods: A Survey," *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND NETWORK SECURITY,* vol. 22, no. 6, pp. 288-296, JUN 30 2022, doi: 10.22937/IJCSNS.2022.22.6.38.

[19] *Experience of PYNQ*, 1st edition 2023. ed. (no. Book, Whole). Springer Nature Singapore : Imprint: Springer, Singapore (in English), 2023.

[20] D. C. Vasile, P. Svasta, and M. Pantazică, "Preventing the Temperature Side Channel Attacks on Security Circuits," in *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 23-26 Oct. 2019 2019, pp. 244-247, doi: 10.1109/SIITME47687.2019.8990788.

[21] Y. Lu, K. H. Boey, M. O. Neill, and J. V. McCanny, "Practical comparison of differential power analysis techniques on an ASIC implementation of the AES algorithm," in *IET Irish Signals and Systems Conference (ISSC 2009)*, 10-11 June 2009 2009, pp. 1-6, doi: 10.1049/cp.2009.1734.

[22] A. Albalawi, V. Vassilakis, and R. Calinescu, "Side-channel Attacks and Countermeasures in Cloud Services and Infrastructures," in *NOMS 2022-2022 IEEE/IFIP Network Operations and*

Management Symposium, 25-29 April 2022 2022, pp. 1-4, doi: 10.1109/NOMS54207.2022.9789783.

[23] M. Korona, T. Wojciechowski, M. Rawski, and P. Tomaszewicz, "Cryptographic Coprocessor with Modular Architecture for Research and Development of Countermeasures Against Power-Based Side-Channel Attacks," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, 27-29 June 2019 2019, pp. 190-195, doi: 10.23919/MIXDES.2019.8787062.

[24] Y. Nozaki and M. Yoshikawa, "Shuffling Countermeasure against Power Side-Channel Attack for MLP with Software Implementation," in *2021 IEEE 4th International Conference on Electronics and Communication Engineering (ICECE)*, 17-19 Dec. 2021 2021, pp. 39-42, doi: 10.1109/ICECE54449.2021.9674668.

[25] A. Hertzfeld. "Creative Think." Folklore. https://www.folklore.org/StoryView.py?project=Macintosh&story=Creative_Think.txt (accessed 2023).