

Santa Clara University

Scholar Commons

Electrical and Computer Engineering Senior
Theses

Engineering Senior Theses

Spring 2021

Machine Learning-Based Side-Channel Analysis on the Advanced Encryption Standard

Jack Edmonds

Tyler Moon

Follow this and additional works at: https://scholarcommons.scu.edu/elec_senior



Part of the [Electrical and Computer Engineering Commons](#)

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Jack Edmonds, Tyler Moon

ENTITLED

**MACHINE LEARNING-BASED SIDE-CHANNEL
ANALYSIS ON THE ADVANCED ENCRYPTION
STANDARD**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE

IN

ELECTRICAL ENGINEERING

Fatemeh Tehranipoor *Sara Tehranipoor*

06/02/2021

Thesis Advisor(s)

date

Shoba Krishnan

June 4th 2021

Department Chair(s)

date

MACHINE LEARNING-BASED SIDE-CHANNEL
ANALYSIS ON THE ADVANCED ENCRYPTION
STANDARD

Jack Edmonds, Tyler Moon

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical Engineering

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical Engineering

Santa Clara, California

Spring 2021

Machine Learning-Based Side-Channel Analysis on the Advanced Encryption Standard

Jack Edmonds, Tyler Moon

Department of Electrical Engineering
Santa Clara University

2021

Abstract

Hardware security is essential in keeping sensitive information private. Because of this, it's imperative that we evaluate the ability of cryptosystems to withstand cutting edge attacks. Doing so encourages the development of countermeasures and new methods of data protection as needed. In this thesis, we present our findings of an evaluation of the Advanced Encryption Standard, particularly unmasked and masked AES-128, implemented in software on an STM32F415 microcontroller unit (MCU), against machine learning-based side-channel analysis (MLSCA). 12 machine learning classifiers were used in combination with a side-channel leakage model in the context of four scenarios: profiling one device and key and attacking the same device with the same key, profiling one device and key and attacking a different device with the same key, profiling one device and key and attacking the same device with a different key, and profiling one device and key and attacking a different device with a different key. We found that unmasked AES-128 can be very vulnerable to this form of attack and that masking can be applied as a countermeasure to successfully prevent attacks in 2 out of the 4 tested scenarios. In addition to providing our experimental results on the following pages, we also plan to release a public GitHub repository with all of our collected side-channel data along with sample analysis code shortly after the time of writing this. We hope that doing so will allow for complete reproducibility of our results and encourage future research without the need for purchasing hardware equipment.

Keywords: Side-channel analysis, machine learning, AES-128, microcontroller

Acknowledgements

We would like to thank our project advisor, Dr. Fatemeh (Sara) Tehranipoor, and our senior design professor, Dr. Cary Yang, for all of their support and guidance over the past year. We would also like to express our appreciation for the help we received via NewAE's online forum regarding the ChipWhisperer toolchain, particularly from user Alex_Dewar. Finally, we would like to thank our families for their love, unwavering encouragement, and support throughout our academic careers.

Table of Contents

Chapter 1 - Introduction	1
1.1 Motivation	1
1.2 Contributions	1
Chapter 2 - Background	3
2.1 Side-Channel Analysis	3
2.2 Machine Learning	3
2.3 Advanced Encryption Standard (AES-128)	5
Chapter 3 - Methodology	9
3.1 Experimental Setup	9
3.2 Evaluation Procedure	10
3.3 Rank Metric	14
Chapter 4 - Results	17
4.1 Unmasked AES-128	17
4.2 Masked AES-128	18
4.3 Classifier Comparison	18
Chapter 5 - Professional Issues and Constraints	20
5.1 Ethical, Science, Technology, and Society	20
5.2 Civic Engagement	20
5.3 Economic	20
5.4 Health and Safety	20
5.5 Manufacturability	20
5.6 Usability	21
5.7 Sustainability	21
5.8 Environmental Impact	21
Chapter 6 - Conclusions	22
6.1 Summary	22
6.2 Future Work	22
References	23
Appendix A: Senior Design Conference Slides	A-1
Appendix B: Hardware and Software	B-1
Appendix C: GitHub Repository	C-1

List of Figures

Figure 1: Example power trace from an STM32F415 MCU, consisting of 2,000 samples of the voltage measured across a shunt resistor in the device’s power supply rail

Figure 2: AES-128 encryption flow (decryption is a similar process with inverse operations)

Figure 3: Evaluation setup with a MacBook Pro, ChipWhisperer-Lite, and STM32F415 targets with a CW308 UFO Board

Figure 4: Flow of the “add round key” and “substitute bytes” operations in the first round of unmasked AES-128

Figure 5: Result of a leakage model SNR calculation on the unmasked software AES-128 profiling set, revealing samples associated with the processing of the 16 S-box output bytes

Figure 6: Result of a leakage model SNR calculation on the masked software AES-128 profiling set, revealing samples associated with the processing of the 16 S-box output bytes

Figure 7: Byte rank plot showing a successful attack, as all 16 bytes converge to a rank of 0 before running out of attack traces

Figure 8: Byte rank plot showing a failed attack, as all 16 bytes do not converge to a rank of 0 before running out of attack traces

List of Tables

Table 1: Attack results for unmasked and masked software AES-128 on an STM32F415 MCU

Table 2: Average byte rank 0 convergence for different classifiers

Chapter 1 - Introduction

1.1 Motivation

With hundreds of billions of dollars lost each year due to hackers and the likelihood that attacks will continue to get stronger being high, staying on top of what is secure and what may leave data at risk is of utmost importance. Anything from login details and private messages to credit card and bank account numbers has the potential of being exposed to hackers if not properly secured, which is why cryptanalysis, or “the art or process of deciphering coded messages without being told the key,” can often be necessary [1]. When employed by an ethical hacker instead of someone who uses it maliciously, attempting to break cryptosystems with methods of cryptanalysis can be a positive endeavor in the sense that it can reveal vulnerabilities that can then be accounted for, even though it’s ultimately what we’d like to protect against.

All cryptographic algorithms, while ideally mathematically secure, are almost certain to have flaws when implemented in hardware, and an important question is whether or not the effort required to break them is worth the reward an attacker would gain in doing so. Attempting to break them non-maliciously allows us to gain a sense of this trade-off and decide whether or not a method of data protection will be secure enough for a proposed use case, along with potentially resulting in the future development and implementation of preventative countermeasures to various forms of attack.

1.2 Contributions

To summarize our work, we evaluated the resistance of AES-128 implemented on an STM32F415 MCU in four different scenarios against MLSCA. 12 machine learning classifiers including both classical and deep learning algorithms were used in combination with a side-channel leakage model to do this. We found that without implementing SCA-specific countermeasures (such as masking in our case), secret encryption keys can be vulnerable to this form of attack. In addition to providing the details of our process and results in this paper, we plan to publish a GitHub repository with all of our collected and analyzed power traces from two microcontrollers along with a Jupyter Notebook to allow for reproducibility of our results and encourage expansion upon our work.

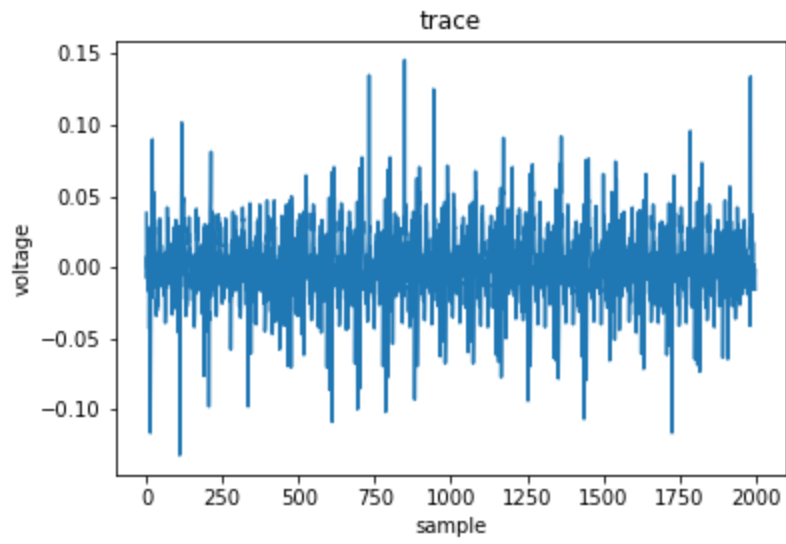


Figure 1: Example power trace from an STM32F415 MCU, consisting of 2,000 samples of the voltage measured across a shunt resistor in the device's power supply rail

Chapter 2 - Background

2.1 Side-Channel Analysis

One method of attacking cryptographic algorithms by attempting to exploit hardware vulnerabilities is side-channel analysis (SCA), which involves analyzing device information associated with the internal workings of an algorithm. More specifically, side-channel analysis entails learning information about a system through analyzing things like device power consumption, electromagnetic radiation, timing, and sound, and in doing so attempting to reveal information that was intended to remain unknown to unauthorized parties. In a sense, side-channel information can be thought of as the side effects of a device's operation. While SCA has been around since before the start of the 21st century, only in the past couple decades has a surge of interest been seen in its potential as a cheap, non-invasive or semi-invasive method of attack. It's pairing with machine learning is even more recent, and has been the focus of our research, with device power consumption being our chosen type of side-channel information (as seen in Figure 1).

2.2 Machine Learning

Machine learning is an area of computer science that involves using algorithms to learn from and identify patterns in datasets - the goal being to make a computer learn something on its own without any (or with little) human intervention. In our project, we used 12 different types of machine learning classification algorithms:

AdaBoost: An ensemble learning method that improves or “boosts” the performance of weak binary classifiers such as Decision Stumps. This algorithm uses an iterative approach when making the classifiers more efficient [2].

Convolutional Neural Network: A model created using multiple layers of neurons which are mathematical functions that output a value based on the calculated weights of each neuron. This mathematical process is also known as convolution [3].

Decision Tree: A prediction model that uses statistics to predict the end result given a set number of inputs. Decision trees have branches as the observational values and leaves as the targeted values [4].

Gaussian Naive Bayes: An algorithm that assumes that the data fits a Gaussian distribution and makes predictions of the output value using the probability of appearing in the distribution. The closer to the tail end of the distribution, the less likely [5].

K-Nearest Neighbors: Out of a set of data points, this algorithm classifies the test set based on the nearest neighbors where “K” represents the amount of data points the algorithm considers that are closest to the test point [6].

Linear Discriminant Analysis: This statistical analysis algorithm uses the dataset given and separates the data into one or more classes using a linear combination of the features [7].

Logistic Regression: An algorithm that uses a logistic model to calculate the probability of a result occurring and makes a prediction based on its probability [8].

Long Short-Term Memory Network: A model with a neural network architecture that utilizes feedback connections which allows for information to persist throughout the training process [9].

Multilayer Perceptron Network: A model with a neural network architecture where every node in one layer connects to every node in the previous and next layer. The activation function at each node is used to calculate the weights for each layer that influence the output [10].

Restricted Boltzmann Machine: A two-layered network where all nodes in the input layer are connected to all the nodes in the hidden layer. Once an output is obtained, it tries to reconstruct the input data from the output and then compares the divergence of the input differences. It makes adjustments based on how well the input is reconstructed [11].

Random Forest: An algorithm that uses multiple deep decision trees and trains all of them on the dataset and then takes the average of all the tree predictions as the final output [12].

Support Vector Machine: This algorithm distinguishes between two or more classes within the dataset by using a linear combination of the features. The way it optimizes this is by using the points of different classes that are closest to each other as support vectors then making a linear boundary at the center of those support vectors [13].

2.3 Advanced Encryption Standard (AES-128)

AES, or the Advanced Encryption Standard, known originally as Rijndael, is one of the most widely used symmetric-key algorithms (where the same key is used for both encryption and decryption) today. Along with being popular in private industry, it serves as the U.S. government's primary method of encrypting and decrypting sensitive data (initially approved for use in 2001) [14]. AES can operate with 128, 192, or 256-bit keys, which is a significant step up from what is commonly regarded as its predecessor, DES (the Data Encryption Standard), which was developed by IBM in the 1970s [15]. AES became a replacement for DES because the effective key size of 56 bits for DES eventually became too susceptible to brute force (or exhaustive search) attacks as computer processing power improved [15]. The smallest AES key size offers $2^{128} = 3.4e+38$ possible combinations, which is considerably larger than the $2^{56} = 7.2e+16$ key space offered by DES.

AES is a block cipher, which means that chunks of bits are operated on together instead of each bit being dealt with separately. It operates on 16 byte-sized chunks (128 total bits) of data regardless of key size, and a series of operations is performed on the 4x4 state array of this data a certain number of times depending on the key size. AES-128 has 10 rounds, AES-192 has 12, and AES-256 has 14. AES-128 was the variant we focused on, and its general flow of encryption operations can be seen in Figure 2.

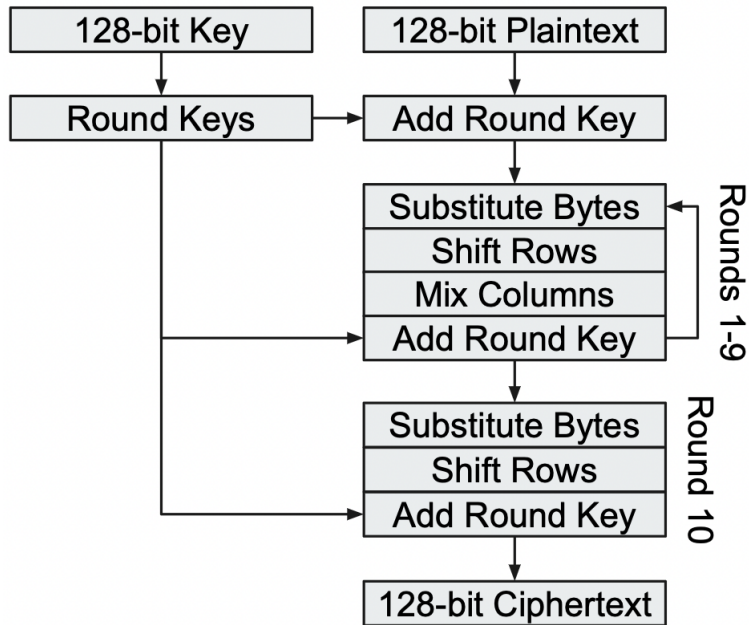


Figure 2: AES-128 encryption flow (decryption is a similar process with inverse operations)

The “add round key” operation takes the XOR of the state array bytes (which are initially loaded with the plaintext byte values along with any necessary padding) and the corresponding round key bytes, which will be discussed more later [15]. It should be noted that the XOR operation is common in symmetric-key cryptography because it’s invertible. As a simple example, with 1100 as our plaintext and 0101 as our key, $\text{plaintext} \oplus \text{key} = 1100 \oplus 0101$ results in the ciphertext 1001, and $\text{ciphertext} \oplus \text{key} = 1001 \oplus 0101$ gets us back to our plaintext of 1100.

The “substitute bytes” or S-box operation substitutes each of the 16 bytes in the state array with a value in the Rijndael S-box [15]. The values of the bytes prior to the substitution are used as indices to determine what values to substitute them with from the S-box lookup table [16]. The table itself is static across AES implementations and there are 256 values in it because a byte has $2^8 = 256$ possible values, thus allowing for indices from 0 to 255. One countermeasure commonly used to protect cryptographic algorithms against SCA is masking, and this technique was applied in one of the two AES-128 implementations we evaluated. Specifically, the masked AES-128 implementation XORed each of the S-box output bytes with an additional mask value in an attempt to disrupt the first-order relationship between the power consumption spike

resulting from this operation and the S-box output byte values [16]. As stated previously, XOR is invertible, and the mask can later be removed so long as its value can be recalled.

The “shift rows” operation applies circular left shifts to the rows of the state array. The first row is not shifted, the second row is shifted once to the left, the third row twice to the left, and the fourth row three times to the left [15]. Because it is circular, a value on the far left of a given row will become the value on the far right after the next shift.

The “mix columns” operation consists of taking the matrix-vector product of a Maximum Distance Separable (MDS) circulant matrix and each of the columns of the state array [15]. An MDS circulant matrix is specially designed to hide the relationship between the plaintext and ciphertext - the details of which we won’t discuss here [17]. It should be noted that this operation does not show up in the final round, as it was determined when the algorithm was developed that including it would not significantly add to the algorithm’s security in the final round [15].

To generate “round keys,” an AES key schedule routine is used to expand the primary key into 10 additional 128-bit subkeys. To obtain the first column of the first round key’s array, you take the far right column of the original key array, upward circular shift it by one, perform an S-box substitution of the bytes, XOR the column with first column of the key array, and XOR the result of this with a 4x1 vector whose first value is a predefined constant dependent on the round number (which can be found in the AES round constant table), and whose second, third, and fourth values are zero [15]. The result of the second XOR (note that XOR is commutative and associative, so the order doesn’t actually matter) makes up the first column of the first round key’s array and can be visualized as being placed in a new fifth column of the key array [15]. The next three columns that will make up the rest of the first round key’s array are found in a similar manner, but the role that first column of the original array played in the generation of the first column of the first round key’s array will be played by the column to its right in the generation of the second column of the first round key’s array, and the newly generated column will play the role that the far right column of the original key array played in its generation [15]. In short, the index of the columns used in the subkey column generation routine will continue to increment by one until all four columns of all round key arrays have been generated. Note that

there are actually 11 instances of the “add round key” operation in AES-128 as shown in Figure 2, or the number of rounds plus one in general [15]. The first of these “add round key” operations typically uses the original key itself, which is often referred to as the zeroth round key.

Chapter 3 - Methodology

3.1 Experimental Setup

To perform our evaluation of the MLSCA resistance of AES-128 implemented on an STM32F415 MCU, we first needed to collect power consumption data associated with encryptions performed by two STM32F415 targets. More specifically, we measured the voltage across a shunt resistor inserted in the power supply rail of a target with the help of a CW1173 ChipWhisperer-Lite. As stated on their site, “The ChipWhisperer® ecosystem presents the first open-source, low-cost solution to expose weaknesses that exist in embedded systems all around us” [18]. The ChipWhisperer-Lite essentially operates like an oscilloscope, while also allowing for communication between a laptop and a target board using an open-source Python package for control. Our evaluation setup consisting of a MacBook Pro, ChipWhisperer-Lite, and two STM32F415 MCUs capable of being mounted on a CW308 UFO Board to facilitate connections can be seen in Figure 3. On the software side, we used Python packages including but not limited to the ChipWhisperer, Keras/TensorFlow, Scikit-learn, NumPy, Pandas, and Matplotlib libraries in combination with a Jupyter Notebook to carry out our data processing and analysis.

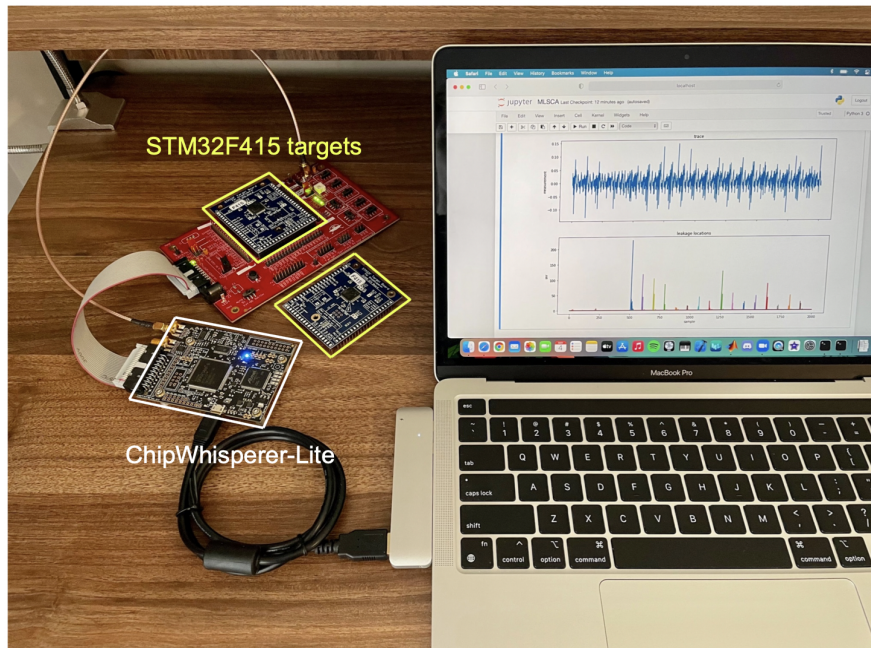


Figure 3: Evaluation setup with a MacBook Pro, ChipWhisperer-Lite, and STM32F415 targets with a CW308 UFO Board

3.2 Evaluation Procedure

Beginning with a standard unmasked software implementation of AES-128, we downloaded the program to the first of our two MCUs and triggered a series of encryptions in order to capture enough power consumption traces for a profiling set for our machine learning classifiers. We decided to collect 60,000 traces consisting of 2,000 samples each, corresponding to 60,000 different encryptions of randomly generated 128-bit plaintext with a static 128-bit key. Through trial and error, 2,000 samples turned out to be enough to capture the operation of interest to us in the first round of an encryption. We also collected four additional sets of 10,000 traces each to serve as our attack sets. The first two of these four sets were collected from the same device that was profiled, with one set of encryptions using the same key that was used in the profiling set and the other using a different key. The second two attack sets were collected from the second STM32F415 MCU, with one set again using the same key that was profiled on the first device and the second using a different key. This same general process was used when collecting traces for datasets associated with a masked software implementation of AES-128 on the STM32F415 MCUs. We saved the total 100,000 collected traces for each implementation along with the associated 128-bit key, plaintext, and ciphertext values from each encryption for later analysis.

After collecting our datasets for the evaluation, we then applied a side-channel leakage model to a profiling set in an attempt to determine which samples in our collected traces may correspond to the processing of an operation of interest to us as attackers. For attacking both the unmasked and masked software implementations of AES-128 on the MCUs, the operation of interest to us was the “substitute bytes” or S-box operation in the first round. Only a few operations occur prior to this in the encryption process - those being the generation of the round keys derived from the original key and an “add round key” operation that consists of an XOR between the bytes of the zeroth round key (or the key itself) and the input plaintext. It’s been shown prior to us that the power consumption corresponding to the processing of the S-box operation may be correlated to its output values, and that if you’re able to predict what those values are, it’s possible to work backwards to the secret key used. This is assuming that as an attacker you kept track of what random plaintext you used for each encryption, as there is only one value out of 256 possible values for the first key byte that when XORed with the first plaintext byte will result in the predicted first output byte of the S-box operation and so forth. A quick for loop can be used to

figure out what the value is for each of the 16 key bytes. Figure 4 shows a general flow of the operations involved in this leakage model.

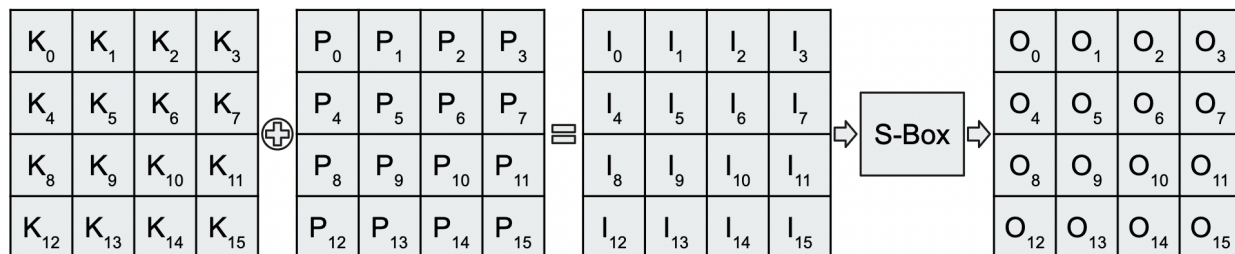


Figure 4: Flow of the “add round key” and “substitute bytes” operations in the first round of unmasked AES-128

A signal-to-noise ratio (SNR) calculation was performed on information from a profiling set in order to actually locate the samples of interest to us that corresponded to the processing of the 16 bytes. While we won’t go into detail on all of the steps of that calculation in this thesis, the full code used to do this will be provided in the Jupyter Notebook hosted on our GitHub repository. Figure 5 shows the result of the SNR calculation for the unmasked software profiling set in plot form. As seen by the peaks in the bottom plot, it’s clear what samples are related to the 16 S-box output bytes in the first round, with the top plot being one of the 60,000 collected power consumption traces. Figure 6 shows the same thing for the masked implementation.

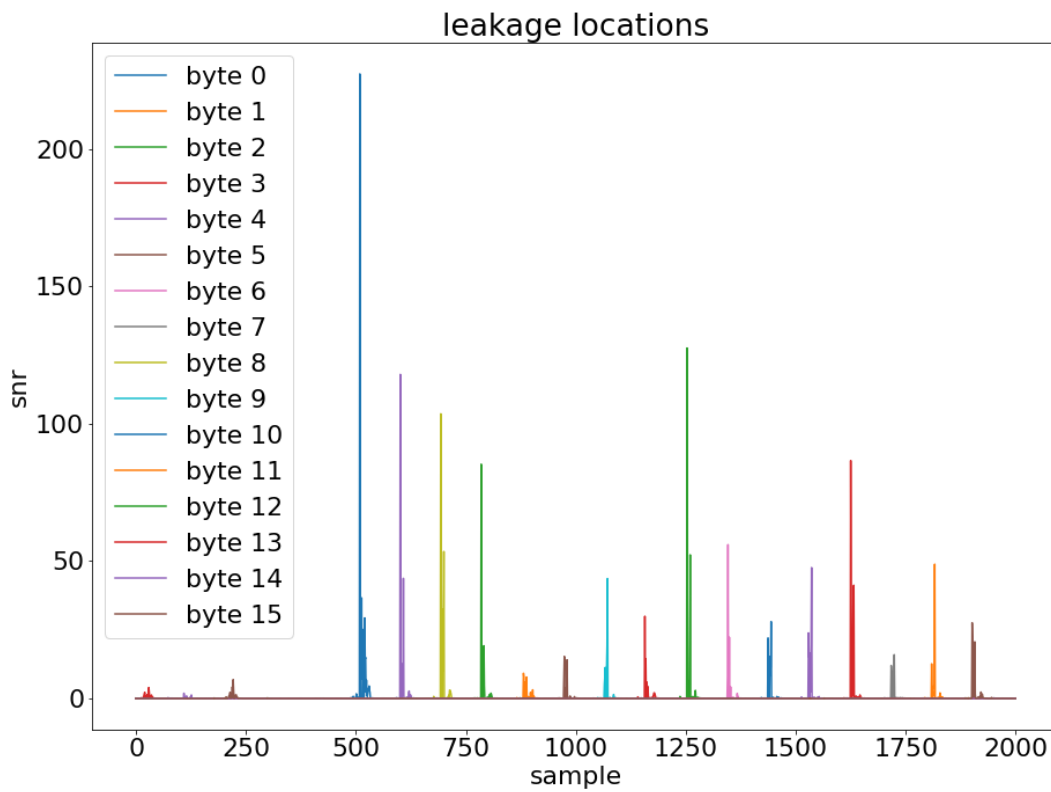
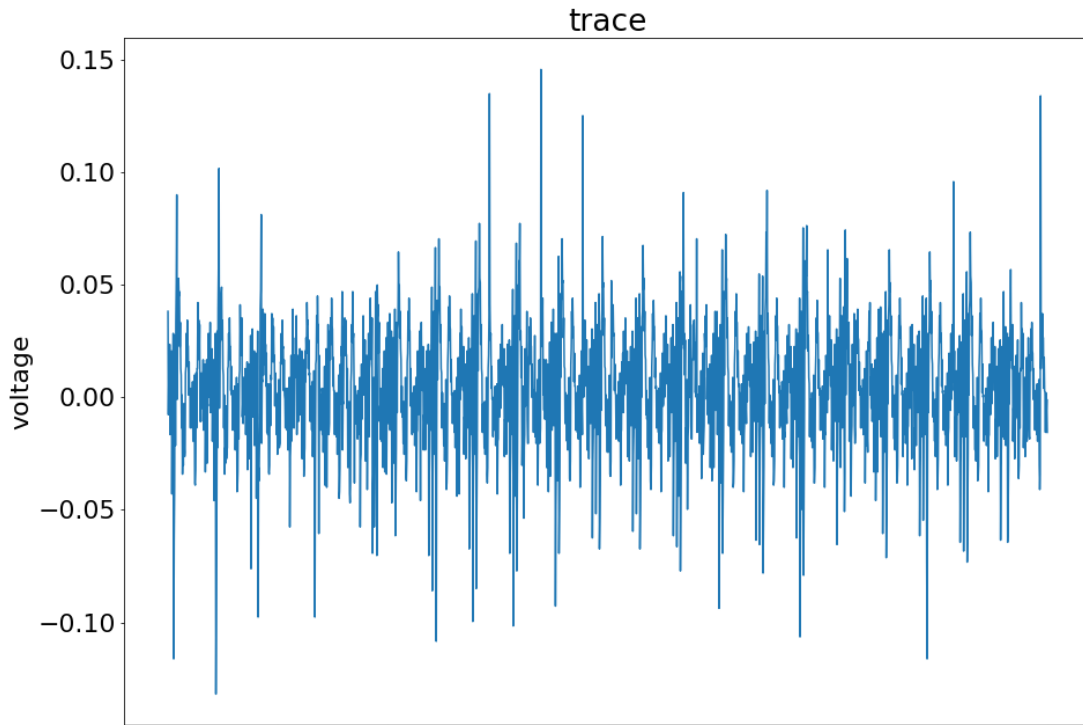


Figure 5: Result of a leakage model SNR calculation on the unmasked software AES-128 profiling set, revealing samples associated with the processing of the 16 S-box output bytes

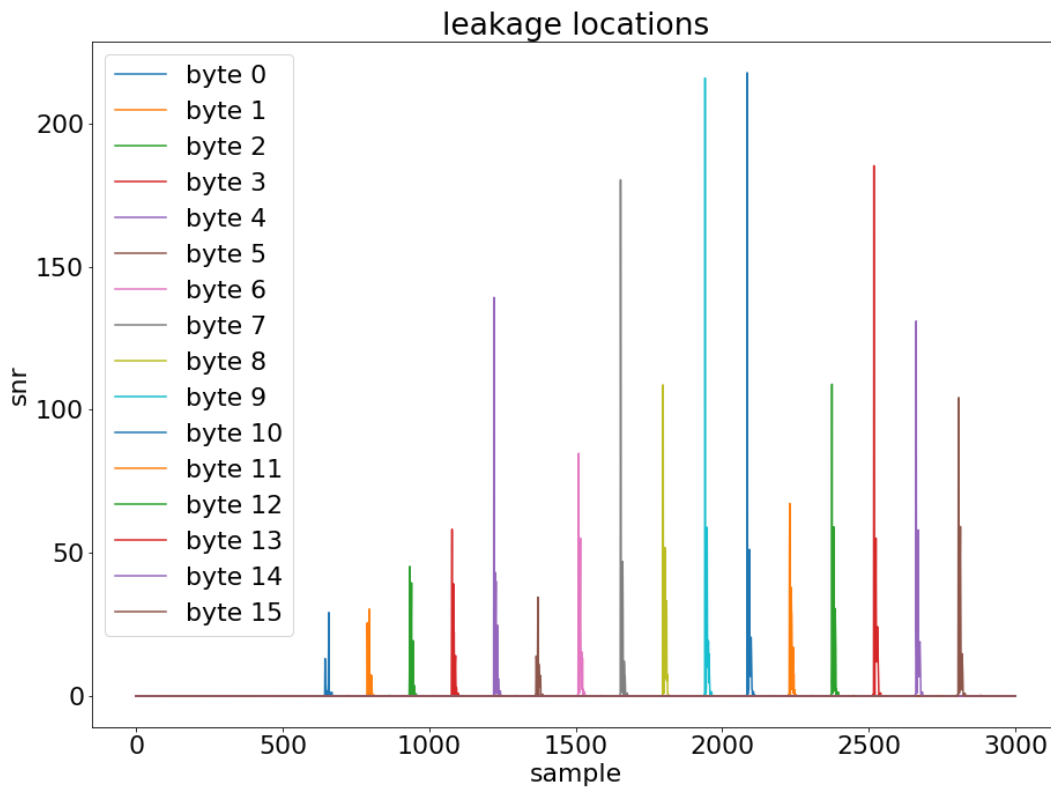
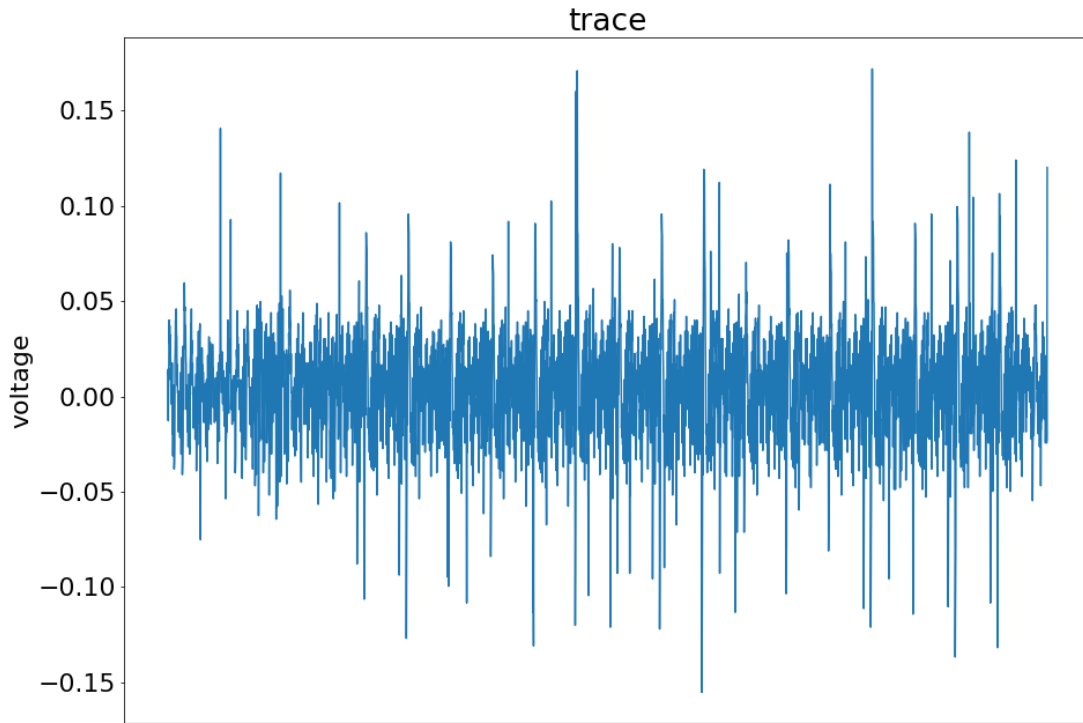


Figure 6: Result of a leakage model SNR calculation on the masked software AES-128 profiling set, revealing samples associated with the processing of the 16 S-box output bytes

Having identified 16 sample values of interest based on the peaks found with our leakage model's SNR calculation, we then created 16 different training and test sets from our profiling and attack sets that would be examined by 12 different machine learning classifiers. For the first byte's training set, we zoomed in on the sample of interest in each of the 60,000 profiling traces, or in some cases several samples centered around the peak SNR index, and used that voltage value or group of voltage values as the predictor in the training set for each observation. Each predictor was labeled with the correct S-box output value we hoped a machine learning classifier would learn to predict. For the corresponding test set, the predictor was also the voltage value(s) associated with the first S-box output byte in each of the 10,000 attack traces, and while we labeled each of them as well in order to check our classifiers' prediction accuracy for the sake of evaluation purposes, in reality we would assume an attack set would not be labeled, as we wouldn't actually know the key ahead of time to check our test predictions against.

For the masked software implementation of AES-128, there are only two real differences in the process described above. The first is that 3,000 samples were collected per trace in order to capture all S-box substitutions in the first round. The second is that an additional mask value was XORed with the S-box output bytes themselves in an attempt to throw off a first-order relationship between the measured voltage value corresponding to the S-box operation and the values we trained our machine learning classifiers to predict. Again, Figure 6 shows the result of the SNR ratio calculation for the masked software AES-128 profiling set in plot form.

3.3 Rank Metric

The method we used to determine whether or not a machine learning classifier was successful in predicting the key we were looking for was by using a metric known in the side-channel analysis research community as "rank" or "byte rank." As stated previously, our training and test sets for the 16 bytes were labeled with the correct S-box output bytes for each encryption, but the accuracy of correct prediction based on a single trace was found to be unreliable. Rather, the predictions from many traces needed to be taken into account in order to have a fighting chance at determining the correct key.

In the case of a byte, there are 256 values it can take, so we asked our machine learning classifiers to return an array of 256 probabilities corresponding to each of the potential S-box output values (0-255). Then, we looped over all of the traces' returned probability arrays, and for each one we determined what key byte value was associated with each of the possible S-box values using the plaintext associated with each trace. Once we figured out what S-box value probability was associated with each of the 256 possible key byte values, we added the probabilities to an array of sums corresponding to key byte value probabilities. As we looped over all of the traces' probability arrays and continued to add up probabilities, we hoped (as an attacker) that eventually the key byte with the highest total value after summing up all probabilities would be the correct one. We also kept track of where the correct value stood in relation to the 255 incorrect values over the course of this process. That way we could plot how its rank changed over time, and how many trace predictions it took before reaching the top rank or highest likelihood. In our case, we chose 0 to represent the highest rank and 255 the lowest. Figures 7 and 8 display example results of a rank calculation for a successful and failed attack in plot form. As with all other steps discussed, the code for computing rank will be provided in our project's GitHub repository.

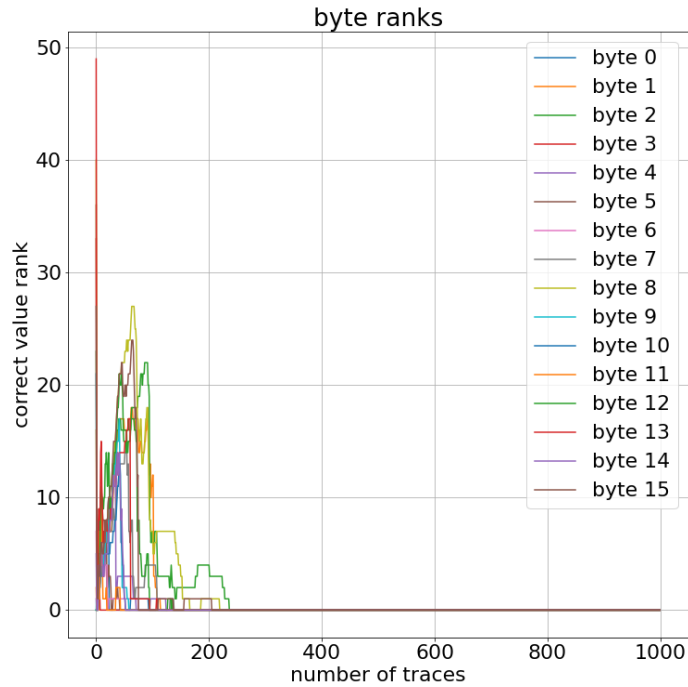


Figure 7: Byte rank plot showing a successful attack, as all 16 bytes converge to a rank of 0 before running out of attack traces

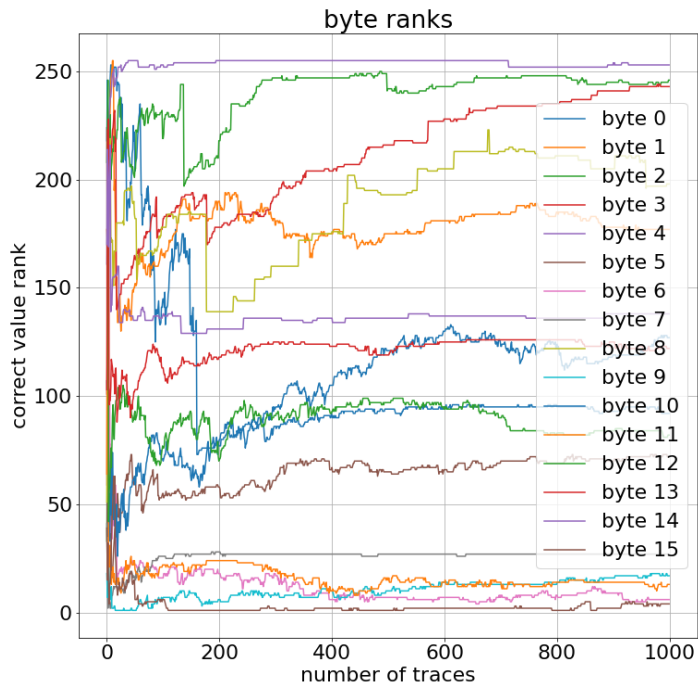


Figure 8: Byte rank plot showing a failed attack, as all 16 bytes do not converge to a rank of 0 before running out of attack traces

Chapter 4 - Results

The following section discusses what we found in performing our evaluation of the MLSCA resistance of AES-128 implemented on an STM32F415 MCU. A summary of our results can be found in Table 1.

Table 1: Attack results for unmasked and masked software AES-128 on an STM32F415 MCU

Attack Success on the STM32F415								
Machine Learning Classifier	Unmasked Software AES-128				Masked Software AES-128			
	SD & SK	SD & DK	DD & SK	DD & DK	SD & SK	SD & DK	DD & SK	DD & DK
AdaBoost	✓	✓	✓	✓	✓	☐	✓	☐
Convolutional Neural Network	✓	✓	✓	✓	✓	☐	✓	☐
Decision Tree	✓	✓	✓	✓	✓	☐	✓	☐
Gaussian Naive Bayes	✓	✓	✓	✓	✓	☐	✓	☐
K-Nearest Neighbors	✓	✓	✓	✓	✓	☐	✓	☐
Linear Discriminant Analysis	✓	✓	✓	✓	✓	☐	✓	☐
Logistic Regression	✓	✓	✓	✓	✓	☐	✓	☐
Long Short-Term Memory Network	✓	✓	✓	✓	✓	☐	✓	☐
Multilayer Perceptron Network	✓	✓	✓	✓	✓	☐	✓	☐
Restricted Boltzmann Machine	✓	✓	✓	✓	✓	☐	✓	☐
Random Forest	✓	✓	✓	✓	✓	☐	✓	☐
Support Vector Machine	✓	✓	✓	✓	✓	☐	✓	☐
Abbreviations								
SD = same device that was profiled								
SK = same key that was profiled								
DD = different device than what was profiled								
DK = different key than what was profiled								

4.1 Unmasked AES-128

To begin with the unmasked software implementation of AES-128, we found that in all four attack scenarios previously discussed (involving two devices and two keys), we were able to recover the full encryption key with all 12 of our chosen machine learning classifiers in combination with a first round S-box leakage model. The classifiers were easily able to do this with little to no tuning, rendering this combination of device and AES variant very vulnerable to this form of attack.

4.2 Masked AES-128

Concerning the masked software implementation of AES-128, we found that in two out of the four attack scenarios, the machine learning classifiers in combination with the same leakage model were again easily able to recover the secret key. However, in the two scenarios involving a different attack key than profiling key, all classifiers failed at predicting it. In rare cases we were able to recover partial keys, but never the full 16 bytes. Given that these two scenarios (attacking the same device that was profiled with a different key and attacking a different device than what was profiled with a different key) are likely the most realistic scenarios out of the four, we think that masking can indeed be an effective countermeasure against MLSCA.

4.3 Classifier Comparison

In terms of how the 12 machine learning classifiers compared to each other performance-wise, Table 2 displays the average number of attack traces required per byte before reaching a rank of 0 for all 12 classifiers in the various scenarios. It should be noted that for the results in this table we used reduced profiling and attack sets (20,000 and 1,000 traces, respectively). This table should not be considered an end-all for determining which classifier is the best for MLSCA in general, but in context of evaluating AES-128 on the STM32F415, our long short-term memory, k-nearest neighbor, and support-vector machine classifiers tended to require the least number of traces to rank the correct key byte values as most-probable. The support-vector machine classifier did however take considerably longer to run than any of the others, so while it did produce a low rank convergence, it wasn't nearly as time-efficient as the rest.

Table 2: Average byte rank 0 convergence for classifiers

Classifier Rank Comparison (20,000 Profiling and 1,000 Attack Traces From an MCU)								
Machine Learning Classifier	Unmasked Software AES-128 Average Number of Attack Traces to Rank 0				Masked Software AES-128 Average Number of Attack Traces to Rank 0			
	SD & SK	SD & DK	DD & SK	DD & DK	SD & SK	SD & DK	DD & SK	DD & DK
AdaBoost	69	49	10	12	8	N/A	9	N/A
Convolutional Neural Network	83	83	7	8	13	N/A	17	N/A
Decision Tree	90	105	40	28	31	N/A	44	N/A
Gaussian Naive Bayes	88	129	15	13	20	N/A	37	N/A
K-Nearest Neighbors	8	7	3	4	5	N/A	7	N/A
Linear Discriminant Analysis	118	136	6	4	8	N/A	19	N/A
Logistic Regression	109	98	6	5	7	N/A	15	N/A
Long Short-Term Memory Network	15	13	6	5	7	N/A	11	N/A
Multilayer Perceptron Network	110	133	12	8	11	N/A	26	N/A
Restricted Boltzmann Machine	156	149	48	57	38	N/A	41	N/A
Random Forest	33	52	25	24	30	N/A	36	N/A
Support Vector Machine	7	9	5	4	4	N/A	16	N/A
Abbreviations								
SD = same device that was profiled								
SK = same key that was profiled								
DD = different device than what was profiled								
DK = different key than what was profiled								

Interestingly, for the successful attacks on masked software AES-128, the average rank 0 convergence was higher when the attacked device was not the same as what was profiled, as one might expect, but this was not the case for unmasked software AES-128. Instead, for all 12 classifiers the average rank 0 convergence was lower when the attacked device was not the same as the one that was profiled. Why this is, we are unsure at the time of writing this thesis, but it may be a topic that's worth investigating in future work. Had the same thing occurred for both unmasked and masked implementations, we would be inclined to believe that the second device just had more obvious leakage (a stronger correlation between the voltage measurements and the S-box output values), but because this was not the case, there is likely a different reason.

Chapter 5 - Professional Issues and Constraints

5.1 Ethical, Science, Technology, and Society

At first glance, it may seem that our project is unethical because we successfully proved that AES-128 can be vulnerable to MLSCA. However, this research provides greater understanding of the need for precautions against such attacks. Without research like this, there would be no improvements to the hardware security field. With knowledge gained by doing this sort of work, we become aware of where the issues and vulnerabilities lie in cryptosystems and are able to focus on making the necessary changes to protect against attacks that target them in the future.

5.2 Civic Engagement

Our project wasn't a product, but rather involved research in the realm of hardware security. Since there are so many microcontrollers out there running AES, this may pose a challenge for implementing widespread hardware countermeasures against MLSCA. Instead improvements may have to remain purely software-based (such as masking) in the short term if people are willing to use them. This is because it would be nearly impossible to change all of the microcontrollers currently in the hands of customers. There's no law out there that would require people to use a device with new countermeasures.

5.3 Economic

The cost of carrying out our research was relatively low. We only needed to buy the equipment discussed previously, which is also listed in Appendix B.

5.4 Health and Safety

There were no serious safety precautions that we needed to take in completing our project due to voltage and current levels being relatively low.

5.5 Manufacturability

As our project is not a product, there's nothing to manufacture on our end. All of the equipment required for our evaluation setup was not manufactured by us.

5.6 Usability

Our research shows that MLSCA is a viable form of attack when no SCA-specific countermeasures are taken, and can be performed without the need for extremely high-end laboratory equipment.

5.7 Sustainability

Our work will be relevant in the future for guiding other research in hardware security for microcontrollers and embedded systems. We're sure that there will be further research and development within this area that will prove to be beneficial in helping with preventing MLSCA going forward.

5.8 Environmental Impact

There's no serious environmental impact from our research besides us using some electricity to power the equipment needed to conduct our experiments, which was no more than what would be needed to power an everyday laptop.

Chapter 6 - Conclusions

6.1 Summary

With hardware security attacks being prevalent in our digital age, it's important to look for vulnerabilities in cryptosystems used to protect sensitive information. In keeping with this idea, the goal of our senior design project was to evaluate the machine learning-based side-channel attack resistance of unmasked and masked software AES-128 implemented on an STM32F415 MCU. We did this using 12 machine learning classifiers in combination with a side-channel leakage model that targeted the first round S-box operation of the encryption algorithm. In completing our evaluation, we found that unmasked software AES-128 on the MCU is very vulnerable to this form of attack, but that masking can protect against it when the attacked key is different than what was profiled. We also found that while all 12 of the machine learning classifiers used were capable of predicting the secret key when at least one of them was able to, some required more power consumption traces than others to do so.

6.2 Future Work

Regarding future work, we hope that providing our collected power consumption side-channel data on GitHub along with a Jupyter Notebook to get started with will more easily allow others along with ourselves to conduct further research in this area. Topics of further investigation could include in-depth optimization of the machine learning classifiers and attempting to better understand why some generally performed better than others in this context. As noted previously, it could also involve investigating why the average rank 0 convergence for unmasked AES-128 was lower when attacking a second device that was not profiled as compared to attacking the same device that was profiled. And one other topic of exploration could be to try out alternate forms of attack not discussed in this thesis on the power consumption data collected. In addition to the MCU data, our research group also plans to provide side-channel data collected from FPGAs on GitHub. A link to our future public repository can be found in Appendix C.

References

- [1] US English. (2020). Retrieved from <https://www.lexico.com/en/definition>
- [2] Kurama, V. (2021, April 9). A Guide To Understanding AdaBoost. Paperspace Blog. Retrieved from <https://blog.paperspace.com/adaboost-optimizer/#:~:text=AdaBoost%20is%20an%20ensemble%20learning,turn%20them%20into%20strong%20ones>
- [3] Saha, S. (2018, December 17). A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way. Medium. Retrieved from <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [4] Decision Trees for Classification: A Machine Learning Algorithm. Xoriant. (n.d.). Retrieved from <https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html#:~:text=%7C-,Blog,namely%20decision%20nodes%20and%20leaves>
- [5] Brownlee, J. (2020, August 14). Naive Bayes for Machine Learning. Machine Learning Mastery. Retrieved from <https://machinelearningmastery.com/naive-bayes-for-machine-learning/#:~:text=This%20extension%20of%20naive%20Bayes,deviation%20from%20your%20training%20data>
- [6] Harrison, O. (2019, July 14). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Medium. Retrieved from <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [7] Brownlee, J. (2020, August 14). Linear Discriminant Analysis for Machine Learning. Machine Learning Mastery. Retrieved from <https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/>
- [8] Swaminathan, S. (2019, January 18). Logistic Regression - Detailed Overview. Medium. Retrieved from <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>
- [9] Understanding LSTM Networks. Understanding LSTM Networks -- colah's blog. (n.d.). Retrieved from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [10] Brownlee, J. (2020, August 14). Crash Course On Multi-Layer Perceptron Neural Networks. Machine Learning Mastery. Retrieved from <https://machinelearningmastery.com/neural-networks-crash-course/>

- [11] Sharma, A. (2018, December 6). Restricted Boltzmann Machines-Simplified. Medium. Retrieved from <https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976>
- [12] Yiu, T. (2019, August 14). Understanding Random Forest. Medium. Retrieved from <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [13] Gandhi, R. (2018, July 5). Support Vector Machine - Introduction to Machine Learning Algorithms. Medium. Retrieved from <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- [14] NIST's Encryption Standard Has Minimum \$250 Billion Economic Benefit, According to New Study. (2018). National Institute of Standards and Technology. Retrieved from <https://www.nist.gov/news-events/news/2018/09/nists-encryption-standard-has-minimum-250-billion-economic-benefit#:~:text=AES%20is%20a%20cryptographic%20algorithm,widely%20adopted%20by%20private%20industry>
- [15] What is AES? - Step by Step. (2019). Medium. Retrieved from <https://medium.com/@14wnrkim/what-is-aes-step-by-step-fcb2ba41bb20>
- [16] Benadjila, R., Prouff, E., Strullu, R., Cagli, E., & Dumas, C. (2018). Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter <https://eprint.iacr.org/2018/053>. pdf, zuletzt geprüft am, 22, 2018.
- [17] Confusion and diffusion. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Confusion_and_diffusion
- [18] ChipWhisperer. (2020). NewAE Technology Inc. Retrieved from <https://www.newae.com/chipwhisperer>
- [19] Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A., & Sen, S. (2019, June). X-DeepSCA: Cross-device deep learning side channel attack. In Proceedings of the 56th Annual Design Automation Conference 2019 (pp. 1-6).
- [20] Golder, A., Das, D., Danial, J., Ghosh, S., Sen, S., & Raychowdhury, A. (2019). Practical approaches toward deep-learning-based cross-device power side-channel attack. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(12), 2720-2733.
- [21] Maghrebi, H., Portigliatti, T., & Prouff, E. (2016, December). Breaking cryptographic implementations using deep learning techniques. In International Conference on Security, Privacy, and Applied Cryptography Engineering (pp. 3-26). Springer, Cham.

- [22] Wei, L., Luo, B., Li, Y., Liu, Y., & Xu, Q. (2018, December). I know what you see: Power side-channel attack on convolutional neural network accelerators. In Proceedings of the 34th Annual Computer Security Applications Conference (pp. 393-406).
- [23] Chawla, N., Singh, A., Kar, M., & Mukhopadhyay, S. (2019, July). Application inference using machine learning based side channel analysis. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.
- [24] Timon, B. (2019). Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems, 107-131.
- [25] Maghrebi, H. (2019). Deep Learning based Side Channel Attacks in Practice. IACR Cryptol. ePrint Arch., 2019, 578.
- [26] Song, S., Chen, K., & Zhang, Y. (2019, June). Overview of Side Channel Cipher Analysis Based on Deep Learning. In Journal of Physics: Conference Series (Vol. 1213, No. 2, p. 022013). IOP Publishing.
- [27] Mukhtar, N., Mehrabi, M. A., Kong, Y., & Anjum, A. (2019). Machine-learning-based side-channel evaluation of elliptic-curve cryptographic fpga processor. Applied Sciences, 9(1), 64.
- [28] Sönmez, B., Sarıkaya, A. A., & Bahtiyar, Ş. (2019, September). Machine Learning based Side Channel Selection for Time-Driven Cache Attacks on AES. In 2019 4th International Conference on Computer Science and Engineering (UBMK) (pp. 1-5). IEEE.
- [29] Jin, S., Kim, S., Kim, H., & Hong, S. (2020). Recent advances in deep learning-based side-channel analysis. ETRI Journal, 42(2), 292-304.
- [30] Carbone, M., Conin, V., Cornelié, M. A., Dassance, F., Dufresne, G., Dumas, C., ... & Venelli, A. (2019). Deep learning to evaluate secure RSA implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 132-161.

Appendix A: Senior Design Conference Slides

SANTA CLARA UNIVERSITY
School of Engineering

Machine Learning-Based Side-Channel Analysis on the Advanced Encryption Standard

Team: Jack Edmonds & Tyler Moon
Advisor: Dr. Sara Tehranipoor
Date: 5-13-2021
Electrical and Computer Engineering

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Outline

- Objectives
- Motivation
- Background & Plan
- Capture & Analysis Setup
- Project Outcomes
- Project Schedule
- Summary & Conclusions
- References

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Objectives

- To evaluate the machine learning-based side-channel attack (MLSCA) resistance of the AES-128 encryption algorithm implemented on an STM32F415 microcontroller (MCU)
- And to provide the SCA research community with a database of our collected side-channel data and sample analysis code on GitHub to allow for expansion upon our work and reproducibility of our results

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Motivation

- Hundreds of billions of dollars are lost each year to hacking and these attacks will continue to get stronger
- "White hat" (ethical) hacking can be employed to examine potential vulnerabilities in today's security standards and encourage the development of countermeasures and improved methods of protection





Image from <https://www.tbrn.co.uk/hacking/00282/what-is-ethical-hacking-white-hat-hackers-explained>

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Side-Channel Analysis (SCA)

- A method of cryptanalysis, or "the art or process of deciphering coded messages without being told the key," that involves monitoring device signals that can unintentionally reveal secret information
- Side-channel information can come in various forms, such as power consumption, electromagnetic radiation, timing, or sound



www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Machine Learning (ML)

- An artificial intelligence method of data analysis that can learn from and identify patterns in datasets
- Our project has involved using machine learning to predict secret data encryption keys based on power consumption side-channel information




Image from <https://xkcd.com/1638/>

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

AES-128

The Advanced Encryption Standard is a symmetric-key block cipher:

Encryption: $C = \text{AES}(K, P)$
 Decryption: $P = \text{InvAES}(K, C)$

(K, P, and C represent the key, plaintext, and ciphertext, respectively)

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

AES-128

Consists of 10 (almost) identical rounds of operations on 16 bytes:

S_0	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	S_{15}

www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

S-box Leakage Model

A correlation may exist between $O_b = \text{S-box}[I_b]$ and the measured voltage V_b across a shunt resistor, where $I_b = K_b \wedge P_b$, and b represents the byte number

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}

P_0	P_1	P_2	P_3
P_4	P_5	P_6	P_7
P_8	P_9	P_{10}	P_{11}
P_{12}	P_{13}	P_{14}	P_{15}

I_0	I_1	I_2	I_3
I_4	I_5	I_6	I_7
I_8	I_9	I_{10}	I_{11}
I_{12}	I_{13}	I_{14}	I_{15}

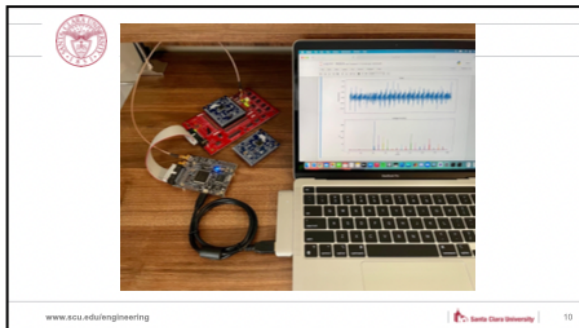
www.scu.edu/engineering

SANTA CLARA UNIVERSITY
School of Engineering

Capture & Analysis Setup

Using the ChipWhisperer open-source toolchain, we were able to trigger encryptions performed by a microcontroller and record the corresponding power consumption for later analysis with ML.

www.scu.edu/engineering



SANTA CLARA UNIVERSITY
School of Engineering

Tested Scenarios

- Unmasked Software AES-128**
 - Same MCU with the same key used for profiling and attacking
 - Same MCU with different keys used for profiling and attacking
 - Different MCUs with the same key used for profiling and attacking
 - Different MCUs with different keys used for profiling and attacking
- Masked Software AES-128**
 - Same MCU with the same key used for profiling and attacking
 - Same MCU with different keys used for profiling and attacking
 - Different MCUs with the same key used for profiling and attacking
 - Different MCUs with different keys used for profiling and attacking

www.scu.edu/engineering

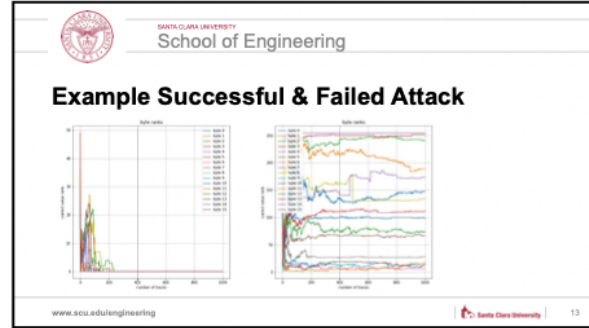
SANTA CLARA UNIVERSITY
School of Engineering

Attack Success on the STM32F415

Machine Learning Classifier	Unmasked Software AES-128				Masked Software AES-128			
	SD & SK	SD & DK	DD & SK	DD & DK	SD & SK	SD & DK	DD & SK	DD & DK
AdaBoost	✓	✓	✓	✓	✓	✓	✓	✓
Convolutional Neural Network	✓	✓	✓	✓	✓	✓	✓	✓
Decision Tree	✓	✓	✓	✓	✓	✓	✓	✓
Gaussian Naive Bayes	✓	✓	✓	✓	✓	✓	✓	✓
K-Nearest Neighbors	✓	✓	✓	✓	✓	✓	✓	✓
Linear Discriminant Analysis	✓	✓	✓	✓	✓	✓	✓	✓
Logistic Regression	✓	✓	✓	✓	✓	✓	✓	✓
Long Short-Term Memory Network	✓	✓	✓	✓	✓	✓	✓	✓
Multilayer Perceptron Network	✓	✓	✓	✓	✓	✓	✓	✓
Restricted Boltzmann Machine	✓	✓	✓	✓	✓	✓	✓	✓
Random Forest	✓	✓	✓	✓	✓	✓	✓	✓
Support Vector Machine	✓	✓	✓	✓	✓	✓	✓	✓

SD = same device | SK = same key | DD = different device | DK = different key

www.scu.edu/engineering | Santa Clara University | 12



SANTA CLARA UNIVERSITY
School of Engineering

Project Schedule

Summer 2020	Fall 2020	Winter 2021	Spring 2021
Survey related research and decide on an encryption algorithm and microcontroller to evaluate	Learn about AES-128 and get it running on an STM32F415 MCU	Run a complete evaluation of an unmasked AES-128 software implementation	Create necessary submissions on materials including final report
Learn about machine learning and deep learning algorithms that could be used for classifying side-channel information	Investigate and try out various leakage models in combination with machine learning classifiers for AES-128	Investigate more secure forms of AES-128 and experiment with using EM radiation as side-channel data instead of power consumption	Prepare for and give final presentation
Learn how to use a ChipWhisperer for side-channel analysis			

www.scu.edu/engineering | Santa Clara University | 14

- SANTA CLARA UNIVERSITY
School of Engineering
- ### Summary & Conclusions
- In completing our evaluation of the MLSCA resistance of AES-128 on an STM32F415 MCU, we found that side-channel vulnerabilities can be exploited when proper countermeasures are not taken
 - We also found that MLSCA does have limitations and can be hindered by taking steps to hide the correlation between side-channel information and encryption calculations
 - Finally, we plan to release our collected datasets and code on GitHub soon after additional parallel research is completed
- www.scu.edu/engineering | Santa Clara University | 15

SANTA CLARA UNIVERSITY
School of Engineering

References

- Paper Notes
- ChipWhisperer — ChipWhisperer 5.5.0 documentation
- CW1173: ChipWhisperer Lite
- STM32F415xx/STM32F417xx
- ANSSI-FR/ASCAD: Side Channels Analysis and Deep Learning
- Keras: the Python deep learning API
- scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation
- What is ethical hacking? White hat hackers explained
- Machine Learning: What it is and why it matters
- xkcd: Machine Learning

www.scu.edu/engineering | Santa Clara University | 16

SANTA CLARA UNIVERSITY
School of Engineering

Thank you!

- A special thanks to Dr. Tehraniipoor and Dr. Yang for their guidance and support over this past year

www.scu.edu/engineering | Santa Clara University | 17

Appendix B: Hardware and Software

- Apple MacBook Pro
- CW1173 ChipWhisperer-Lite
- CW308 UFO Board
- 2x STM32F415 MCU targets
- Jupyter Notebook
- Python with packages including
 - ChipWhisperer
 - Keras/TensorFlow
 - Scikit-learn
 - NumPy
 - Pandas
 - Matplotlib

Appendix C: GitHub Repository

A GitHub repository with the side-channel data and code associated with our thesis will be found at this link once public: <https://github.com/jsedmonds/cosi>