

9-13-2012

# Effects of Architecture on Information Leakage of a Hardware Advanced Encryption Standard Implementation

Eric A. Koziel

Follow this and additional works at: <https://scholar.afit.edu/etd>

Part of the [Computer and Systems Architecture Commons](#), and the [Information Security Commons](#)

---

## Recommended Citation

Koziel, Eric A., "Effects of Architecture on Information Leakage of a Hardware Advanced Encryption Standard Implementation" (2012). *Theses and Dissertations*. 1127.  
<https://scholar.afit.edu/etd/1127>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [richard.mansfield@afit.edu](mailto:richard.mansfield@afit.edu).



**EFFECTS OF ARCHITECTURE ON INFORMATION LEAKAGE OF A  
HARDWARE ADVANCED ENCRYPTION STANDARD IMPLEMENTATION**

THESIS

Eric A. Koziel

AFIT/GCO/ENG/12-25

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

---

---

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

**EFFECTS OF ARCHITECTURE ON INFORMATION LEAKAGE OF A  
HARDWARE ADVANCED ENCRYPTION STANDARD IMPLEMENTATION**

**THESIS**

Presented to the Faculty

Department of Electrical & Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Eric A. Koziel, BS

September 2012

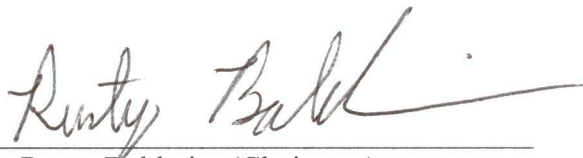
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**EFFECTS OF ARCHITECTURE ON INFORMATION LEAKAGE OF A  
HARDWARE ADVANCED ENCRYPTION STANDARD IMPLEMENTATION**

Eric A. Koziel, BS

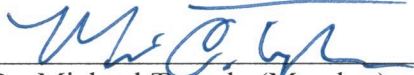
Civilian

Approved:



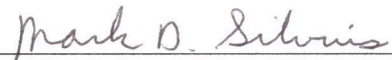
Dr. Rusty Baldwin (Chairman)

4 Sep 12  
Date



Dr. Michael Temple (Member)

45 Sep 12  
Date



Mark Silvius, Major, USAF (Member)

30 Aug 2012  
Date

### **Abstract**

Side-channel analysis (SCA) is a threat to many modern cryptosystems. Many countermeasures exist, but are costly to implement and still do not provide complete protection against SCA. A plausible alternative is to design the cryptosystem using architectures that are known to leak little information about the cryptosystem's operations. This research uses several common primitive architectures for the Advanced Encryption Standard (AES) and assesses the susceptibility of the full AES system to side-channel attack for various primitive configurations. A combined encryption/decryption core is also evaluated to determine if variation of high-level architectures affects leakage characteristics. These different configurations are evaluated under multiple measurement types and leakage models. The results show that different hardware configurations do impact the amount of information leaked by a device, but none of the tested configurations are able to prevent exploitation.

## **Acknowledgments**

I would like to express my sincere appreciation to my faculty advisor, Dr. Rusty Baldwin, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. I would also like to thank all current and former members of the AFIT SCA research group for your insight and guidance throughout my experimentation and analysis. Finally, I would like to thank the Air Force Research Labs for sponsoring this effort.

Eric A. Koziel

## Table of Contents

	Page
Abstract .....	iv
Acknowledgments .....	v
Table of Contents .....	vi
List of Figures .....	viii
List of Tables .....	x
I. Introduction .....	1
II. Literature Review .....	3
2.1 Background.....	3
2.2 Formal Models.....	8
2.3 Modern Attacks and Countermeasures.....	12
2.4 Advanced Encryption Standard .....	17
2.5 Summary.....	25
III. Methodology .....	27
3.1 Problem Definition .....	27
3.3 System Boundaries .....	30
3.4 System Services .....	31
3.5 Workload .....	32
3.6 Performance Metrics .....	33
3.7 System Parameters.....	34
3.8 Factors .....	36
3.9 Evaluation Technique.....	38
3.10 Experimental Design .....	39
3.11 Methodology Summary .....	39



IV. Results and Analysis .....	41
4.1 Effects of Primitive Architecture Variation .....	41
4.2 Effects of High-Level Architecture Variation .....	54
4.3 Conclusions & Reasoning .....	57
4.4 Summary.....	60
V. Conclusions and Future Work.....	61
Appendix A: Bit Maximum Leakage Tables .....	64
Appendix B: Byte Maximum Leakage Tables.....	68
Appendix C: Byte First and Last Round Tables .....	72
Appendix D: <i>t</i> -Test p-Value Tables .....	74
Bibliography .....	77

## List of Figures

	Page
Figure 2.1: Simple Power Analysis Attack on RSA.....	6
Figure 2.2: Theoretical Device for Formal Model.....	10
Figure 2.3: Leakage Map of a Software AES Implementation with High Correlations....	14
Figure 2.4: Leakage Map of a Software AES Implementation with Poor Correlations ...	14
Figure 2.5: SubBytes Stage of AES .....	18
Figure 2.6: ShiftRows Stage of AES .....	18
Figure 2.7: MixColumns Stage of AES .....	19
Figure 2.8: AddRoundKey Stage of AES .....	19
Figure 2.9: General Structure of AES-128 .....	19
Figure 2.10: Iterative Architecture for AES .....	23
Figure 2.11: Pipelined Architecture for AES.....	23
Figure 2.12: Combined EN/DE Core for AES.....	24
Figure 3.1: AES Memory Interface .....	31
Figure 4.1: EM HW Leakage Map .....	43
Figure 4.2: EM HD Leakage Map .....	43
Figure 4.3: Power HD Leakage Map .....	43
Figure 4.4: EM HW Average Maximum Correlations .....	44
Figure 4.5: Power HD Average Maximum Correlations .....	44
Figure 4.6: Power HW Average Maximum Correlations .....	44
Figure 4.7: Initial ARK Average Maximum Correlations .....	47
Figure 4.8: Final ARK Average Maximum Correlations .....	47
Figure 4.9: Leakage Map with Inaccurate Correlation Spikes .....	48
Figure 4.10: Power HD Byte Leakage Map.....	50
Figure 4.11: EM HW Byte Leakage Map.....	50
Figure 4.12: Power HD Byte Average Maximum Correlations .....	51
Figure 4.13: EM HD Byte Average Maximum Correlations.....	51
Figure 4.14: First Round Power HD Average Maximum Correlations .....	52
Figure 4.15: Last Round Power HD Average Maximum Correlations .....	52

Figure 4.16: Mean Bit Correlations .....	55
Figure 4.17: Maximum Bit Correlations.....	55
Figure 4.18: Mean Byte Correlations.....	56
Figure 4.19: LUT-LUT Configuration Round 6 SB .....	58
Figure 4.19: GAL-LUT Configuration Round 6 SB.....	58
Figure 4.21: Round 6 SB Correlation Comparison.....	59

## List of Tables

	Page
Table 2.1: Comparison of Required Correlation Coefficients to Assert DPA Resistance.	14
Table 3.1: Correlation Confidence.....	34
Table 3.2: Factors and Levels .....	37
Table 4.1: Bit Allocation of Variation .....	45
Table 4.2: Average Bit Correlation Differences by Stage .....	46
Table 4.3: Number of Exploitable Bits per Confidence Level .....	49
Table 4.4: First and Last Round Average Maximum Correlations.....	52
Table 4.5: Power HD MC $t$ -Test P-Values .....	53
Table 4.6: Number of Exploitable Bytes per Confidence Level.....	54
Table 4.7: Maximum Correlation Statistics by Architecture .....	55
Table 4.8: High-Level Average Correlation Differences .....	56
Table 4.9: Combined EN/DE Exploitable Bits and Bytes .....	57

# **EFFECTS OF ARCHITECTURE ON INFORMATION LEAKAGE OF A HARDWARE ADVANCED ENCRYPTION STANDARD IMPLEMENTATION**

## **I. Introduction**

Cryptographic algorithms and devices are used in many modern services and devices, including cellular communications, electronic banking, and Internet services. This means that the security of many types of personal and classified information depends on the strength of the algorithm used. Most modern algorithms are peer-reviewed so that as many weaknesses as possible are identified and reinforced. In this way, the algorithms themselves provide a level of security. However, side-channel analysis (SCA) completely bypasses the strength of modern cryptographic algorithms by attacking the implementation of the algorithm instead of the algorithm itself.

SCA observes physical phenomena that emanate from a device during operations that use secret information such as cryptographic keys. Such phenomena include power consumption, electromagnetic emissions, heat, and acoustics. All of these phenomena are considered side channels that indirectly leak information about the internal state of a system. Statistical analysis of such information can determine the secret information.

Consider a two-way radio that can encrypt and decrypt data with a pre-stored secret key. If an enemy were to compromise just one of these radios it could have a localized impact on allied combat capabilities. Copies of the device must be made to have a broader impact on combat operations. But without knowledge of the secret key, creating functionally similar devices is not possible. SCA can be used to attack this compromised device and ascertain the secret key.

Susceptibility to SCA techniques is a problem for many sensitive operations, including the secure storage of data and secure communications. Thus developing devices resistant to these attacks is necessary for maintaining secrecy in many situations. Current countermeasures do not completely protect most devices and require significant development time and resources to implement. Determining secure design practices would lead to a higher degree of resistance to side-channel analysis attacks.

A major difficulty in protecting devices from side channel analysis is most countermeasures require significant development time and resources. Additionally, information leakage is not well understood or even consistent. Simulation and analytical models are not sufficient to assess a given device's weakness to side channel attacks as they do not explicitly model side channel effects. Furthermore, how the internal architecture of a device affects its leakage characteristics is not well understood, so best practices for secure hardware design have not been developed.

The primary goal of this research is to investigate the leakage characteristics of several hardware design styles and propose best practices when designing hardware for security. A secondary goal is to develop a hardware implementation of the Advanced Encryption Standard (AES) that exhibits a high resistance to side channel attacks.

This chapter contains a brief overview of the problem and research goals. Chapter 2 reviews basic concepts and current literature. Chapter 3 describes the experimental approach, system parameters, and evaluation methodology used to accomplish the research goals. Chapter 4 discusses the results of the experiment and analyzes the findings. Chapter 5 summarizes the research findings and outlines several areas for future research and investigations.

## **II. Literature Review**

This chapter contains background information relevant to this effort. It is not meant to be a comprehensive review of related research, nor a tutorial on the basics of cryptography and computing sciences. Each section covers an important aspect of side channel analysis (SCA) that influences the decisions made during construction of the experiment and analysis of the results. Specifically, the role and history of SCA is introduced with reference to standard attacks. A formal model for information leakage is also introduced. Several modern attacks and countermeasures are discussed, along with their relative strengths and weaknesses. Finally, the structures and variants of the Advanced Encryption Standard (AES) are described with respect to the potential for information leakage.

### **2.1 Background**

This section discusses some of the fundamentals and history of SCA. SCA is placed within the cryptanalysis branch of cryptology. Initial techniques are discussed, and examples of classical attacks are given. Several fundamental principles are described.

#### **2.1.1 Cryptology and Side Channel Analysis**

Cryptology is the science of hiding or obscuring messages, often with a secret key or process. In practice, cryptology is broken into two separate fields based on the primary goals: cryptography and cryptanalysis. Cryptography designs secure cryptosystems (ciphers); that is, designs systems that diffuse and confuse inputs in such a way that the output reveals very little about the original message. This can be regarded as the “defensive” side of cryptology. The goal of cryptanalysis, in contrast, is to extract the

original message or other secret information, such as a key, from the cryptosystem. As such, it can be considered the “offensive” side of cryptology [StL07].

Cryptanalysis as a whole refers to any technique, process, observation, or manipulation that enables an attacker to gain insight into secret information being used within the cryptosystem. There are many different approaches to attacking current and theoretical ciphers, but all fit into several distinct categories of weaknesses being exploited.

- Algorithmic weakness: the underlying structure of the cipher does not adequately obscure the information, or the secret key can be determined at a rate better than brute force [Sch00]. Attacks in this category include linear cryptanalysis and differential cryptanalysis. Solutions to certain mathematical problems believed to be hard (such as factoring large semiprimes or discrete logs) also fall into this category.
- Protocol weakness: messages relating to processing, errors, data transfer, or authentication reveal too much information about the secret or the internal state of the cryptosystem. Attacks against these weaknesses are typically observed when a cryptosystem is used remotely. This category also includes any attacks that involve the attacker intruding on some of the communications, such as man-in-the-middle attacks.
- Implementation weakness: the platform on which the cryptosystem is implemented is insecure. This type of weakness stems from oversights in the physical or software structure of a cryptosystem. Some examples include improper generation of random numbers, secret information that is stored non-



securely in memory, and observable activity of the cryptosystem. This extends to side channel analysis attacks and techniques.

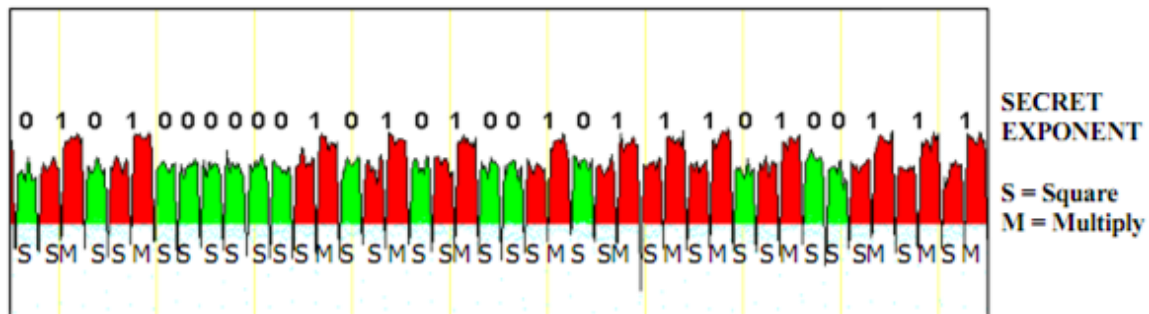
While the overall security of a cryptosystem relies on careful consideration of each of these types of weaknesses, implementation weaknesses have emerged as the major threat to many modern cryptosystems. This is mainly due to the fact that unintended emissions of electronic devices reveal information about the internal state of the cryptosystem. Many cryptographic algorithms are designed with the assumption that no information about its internal state is observable to the outside world; knowledge of intermediate values, then, significantly weakens or trivializes a cryptosystem [KJJ11].

It is most often not possible to directly observe intermediate values, but the physical characteristics of a cryptosystem can be analyzed to determine likely candidates for the intermediate values. The first major exploitation of this type was introduced in [KJJ98], using observations of the current used by a device thereby determining its secret key through statistical analysis. This process was later extended to electromagnetic observations [KJJ99, GMO11]. Many types of countermeasures have been discussed to defeat these and related attacks, but as of yet there is no adequate solution.

### **2.1.2 Power Analysis**

The primary principle behind SCA techniques is a device behaves differently as it performs different operations. This is distinct from a microcontroller performing distinct instructions; it refers to any changes that occur within the cryptosystem. Such activity includes transistors toggling, memory operations, and capacitors charging and discharging. As data is processed, the system behavior is correlated to this data. If this

correlation can be predicted, hypotheses about the secret information can be tested against the observed activity of the device [MOP07].



**Figure 2.1: Simple Power Analysis Attack on RSA**

Power analysis is split into two primary types: simple power analysis (SPA) and differential power analysis (DPA). SPA determines the behavior of a cryptosystem through manual examination of the power traces. A famous example of SPA exploits an implementation of the RSA algorithm (Figure 2.1) [KJJ11]. In this example, the secret exponent bits can be determined as the device performs fast exponentiation. Squaring is part of every operation, however if the exponent bit is a 1 a multiply operation will also occur which uses additional resources and time, both of which are visible in the trace. Most times SPA will not lead to such complete information leakage, but even so it serves as a means of identifying regions with potentially high leakage. This is important since reducing the search space has a significant impact on other more computationally-intensive techniques.

Differential power analysis analyzes differences between traces or sets of traces. The strength of differential analysis comes from its ability to eliminate the random variations of a trace to isolate the core signal of interest by taking the average of the traces for a single execution of a cryptosystem. Given enough traces and assuming the

noise follows a normal Gaussian distribution, the noise will cancel itself out and only the core signal will remain. This is especially useful when comparing sets of known input differences, as the only differences between the sets should be due solely to the differences in input data. This is a somewhat naïve approach to DPA, but it forms the basis of the matching that occurs in correlation-based DPA [KJJ11].

Similar to noting variations in power consumption, electromagnetic emissions also leak information about the switching activity of a cryptosystem. Electromagnetic (EM) analysis also has simple and differential components (SEMA and DEMA, respectively), and the general process is the same as for power. EM signals differ from power signals in that there is no DC offset (low frequency) component and the signal is bi-polar. Electromagnetic emissions also become stronger at higher frequencies, so they can be more easily exploited than power against high-frequency electronics [Ris10]. Another difference is the collection setup: power analysis requires taking measurements from a primary power line to the device while electromagnetic emissions can be captured with a specialized probe placed over the physical location of execution. Thus power measurements often include the activity of other miscellaneous components, while electromagnetic measurements provide more localized activity. Nearly all advanced DPA techniques also apply to DEMA as well, so both are implied in any discussion of DPA throughout the remainder of this document. More advanced attacks using simultaneous power and EM observations substantially reduce error in attack calculations [ARR03].

## 2.2 Formal Models

A formal model for information leakage generalizes the case for why and how information leaks in a device via the proposition of several axioms which describe primary assumptions about information leakage. This formal model is extended to analytical leakage models, which extract information from physically observable phenomena emitted from the device during operation.

### 2.2.1 General Model

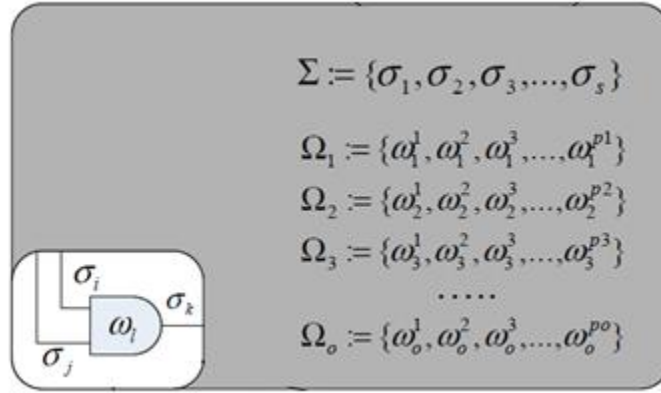
To create attacks effective against many different types of cryptosystems, several underlying assumptions are used to develop appropriate attacks. In these models, cryptosystems are viewed as abstract computers defined as collections of special Turing machines that interact with each other and share a special common memory [SMY06]. The primary hypothesis is these computers leak information during their operation. With this in mind, cryptosystems conform to 5 informal axioms [MiR04]:

1. Computation and only computation leaks information.
2. The same computation leaks different information on different computers.
3. The information leakage depends on the chosen measurement.
4. The information leakage is local.
5. All information leaked through physical observations can be efficiently computed from a target device's internal configuration.

Axiom 1 highlights the difference between data at rest and data in motion; that is, it is assumed that information can only be leaked when it is used in some manner of computation. Secrets stored within memory are assumed to be secure and cannot be

observed if they are not used. Axiom 2 captures the notion that the algorithm is abstract and leaks differently on different platforms. Axioms 3 and 4 state that the collection of measurements is imperfect and introduces additional randomness to the observations. Axiom 4 also implies that leakages are the same for a device each time the same inputs are given. Finally, Axiom 5 states that the information leakage is a computable function of the computer's internal configuration, the chosen measurement, and other external randomness beyond normal control [SMY06]. These axioms do not perfectly capture the operations of realistic systems, but for the most part these are reasonable assumptions and form the basis for attack methodologies.

With these axioms in mind, each Turing machine subcomponent of the computer produces a signal when invoked. The sum of these signals is the total observable signal for a potential attacker. As the overall state of the computer changes, variation occurs in the signals of each Turing machine. These variations constitute the information leakage. Discerning individual variations from the full signal is thus the goal for the attacker. This process is made more difficult in the presence of other subcomponent signal variations in addition to the external noise introduced as part of axioms 3 and 4 [SMY06]. Figure 2.2 graphically depicts the formal model, where  $\Sigma$  represents the set of all signals, and each  $\Omega$  represents the set of basic operations performed to query the system. Each  $\sigma$  is a distinct internal signal of  $s$  total signals.  $\omega_i^j$  is considered primitive element  $j$  of operation  $i$  for  $pj$  total primitive elements. In this case, a primitive element is defined as a logic gate.



**Figure 2.2: Theoretical Device for Formal Model**

### 2.2.2 Leakage Models

Properly attacking many types of cryptosystems depends on the ability of the selected leakage function to correctly identify leakages in the observations by comparing variations in the main signal to expected variations given knowledge of the input/output and a hypothesis for the secret key. From the formal model described in Section 2.2.1, leakage information is present in the signal, but its relative strength depends on the complexity of the rest of the system and the errors introduced by the measurement process. In a realistic system the variation occurs as part of the device's internal structure and construction. Distinguishing these variations involves accurately predicting the activity of the registers, transistors, or other components. These predictions constitute the leakage model.

A primary model pre-computes all intermediate values of the cryptosystem given a hypothesis for the secret information. The Hamming weight of each value serves as the predictor for the signal variations. This leakage model attempts to identify instances where the full intermediate value is either transferred or enters an empty register (all 0s).

Both of those situations are uncommon in optimized designs, but the model is simple and easy to calculate. It is more effective when used in an EM environment, as the emanations of bus lines are more likely to appear in the observed signals. This is often called a Hamming Weight model [Cob11].

Another common leakage model uses the expected toggling activity of registers as a cryptosystem's internal state changes. The expectation is that transistors only use additional current when switching from one state to another. This activity can be predicted with a hypothesis of the new value entering the register in addition to a hypothesis for the value it is replacing. The two hypotheses are XORed and the Hamming weight of the result is used as the predictor. Since registers are very common in electronic cryptosystems, this model is particularly effective. It may not be feasible in many cases however, as it requires knowledge of the internal system structure and data flow. This type is referred to as a Hamming Distance or Hamming Difference model [Cob11].

There are other leakage models, but it is difficult to create a model that represents all leakages in a device. It is also difficult to create a model that captures leakages of many different devices. The ability to create a proper model for a device is dependent on knowledge of its internal structure and operation. While Kerchoff's principle makes it unlikely an attacker will have no information on a device's underlying structure, there are very few instances where it is reasonable to assume that complete knowledge is available. Just as with the hypotheses for the intermediate values themselves, hypotheses for appropriate models must also be tested to best capture the information leakage.

## 2.3 Modern Attacks and Countermeasures

SPA and DPA are extended by a variety of statistical and probabilistic attacks applicable to many cryptosystems. These attacks are categorized into passive attacks and active (or invasive) attacks. Passive attacks operate on the observations of devices; it is not necessary to alter the performance of the device to execute the attack. Active attacks do affect the execution of the system to cause errors or force certain execution conditions. Such attacks include fault injection, glitching, and physical modification of the logic or circuit elements. Due to the cost and high degree of complexity, active attacks are not discussed in this document. There are many types of passive attacks that exist, but only the attacks used in this research are discussed in-depth.

### 2.3.1 Leakage Mapping

A leakage map [Cob11] is produced by performing a correlation attack on observations of a target cipher via a Pearson product-moment correlation process. This type of correlation matches the expected value of a given operation with the variations at each point in the observation of an encryption. The key metric for this is

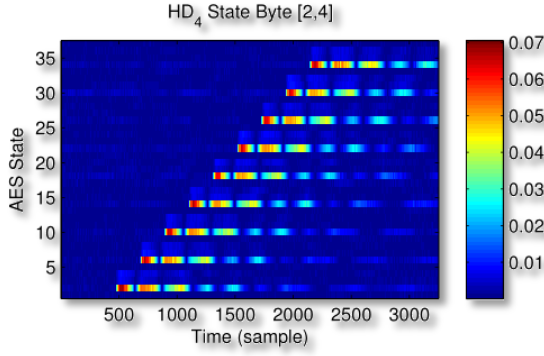
$$r_{xy} = \frac{\sum x_i y_i - n \bar{x} \bar{y}}{(n-1) s_x s_y} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}. \quad (2.1)$$

where each  $\mathbf{x}_i$  term is an observed value for the signal and each  $\mathbf{y}_i$  term is the expected value at time sample  $\mathbf{i}$ . In practice,  $\mathbf{X}$  is a matrix with the observed traces as rows and each observed sample ( $\mathbf{x}$ ) as a column.  $\mathbf{Y}$  consists of the expected values for each stage of the cipher, and is arranged with the cipher stages ( $\mathbf{y}$ ) as the rows and the encryption submission along the columns. The resulting matrix from this calculation is a samples-

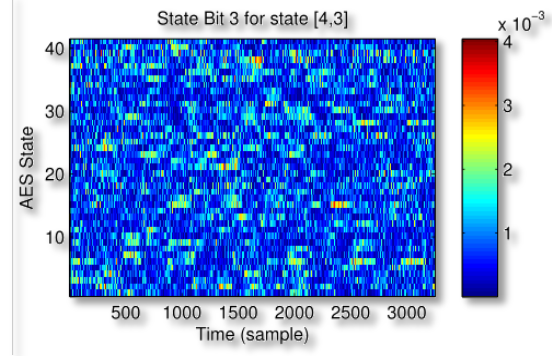


by-stages matrix of correlation coefficients between a sample and the expected value for a given stage.

The matrix returned contains correlation coefficient values in the range  $[-1,1]$ . These coefficients indicate the magnitude of correlation between the two random variables; for leakage mapping, it specifically relates the extent that the hypothesis value for a specific intermediate value is shown in a specific observed sample of the encryption. Note that strong negative correlations still imply dependence between the two variables, but results in a decrease in signal strength as opposed to an increase. Thus the absolute value of the coefficients is used in practice. Knowledge of these correlations is extremely helpful to an attacker, as this information can be used to determine the best samples to attack for a given stage of the cipher. Alternatively, a system designer can use this information to redesign parts of the system to leak less information, or compare different implementations on the basis of information leakage. A sample leakage map of an AES implementation is shown in Figure 2.3. A stepped sequence of high-correlation points is consistent with AES's general structure, as each block of high correlation corresponds to a specific round of AES. In contrast, a leakage map with very little correlation is shown in Figure 2.4, and the coefficients appear effectively random. This is the ideal for a defender and the worst-case for an attacker, but correlation results can be improved by using additional traces. Table 2.1 contains a comparison of several collection scenarios, and describes the maximum allowable correlation coefficient values to assert  $N_t$  trace resistance of a system [Cob11]. Cobb proposes to use an  $\alpha$  value of .1 to allow for brute force attacks to rectify incorrect values, while Mangerd et al suggest an  $\alpha$  value of .0001 to give higher assurance that the full key is correct.



**Figure 2.3: Leakage Map of a Software AES Implementation with High Correlations**



**Figure 2.4: Leakage Map of a Software AES Implementation with Poor Correlations**

**Table 2.1: Comparison of Required Correlation Coefficients to Assert DPA Resistance**

$N_t$	$\alpha = .1$	$\alpha = .01$	$\alpha = .001$	$\alpha = .0001$
1000	.0573	.1038	.1375	.1650
10,000	.0181	.0329	.0437	.0526
100,000	.0057	.0104	.0138	.0166
1M	.0018	.0033	.0044	.0053
1B	.0006	.0010	.0014	.0017
10B	.0002	.0003	.0004	.0005

### 2.3.2 Template Attacks

Chari et al introduced the concept of template attacks in [CRR02], and [ReO04] developed a practical methodology to utilize the attacks. Template attacks assume that a suitable model for a target device can be built, such that only a limited number of observations on an actual target are needed in order to retrieve the secret key. This changes the attack approach to distinct training and attack phases. The training phase builds templates for different key and plaintext values on a device similar to the target device; in this case, a template consists of a vector of means for each observed time

sample and a covariance matrix for each point. This covariance matrix captures the linear dependence between noise vectors for the given operation conditions; specifically, it models the noise probability distribution. A perfect model would evaluate all possible combinations of keys, but this is highly impractical since it is at least as intensive as brute-forcing the secret key. Thus in practice only a small subset of templates are used, but models become more accurate with additional templates. Templates can also be made more efficiently through trial classifications on the same training device [ReO04].

The attack phase takes traces from the actual device under attack and attempts to fit them to the available templates by computing noise vectors for each template and determining the probabilities that the observed trace is statistically the same as the template. In an ideal case, the template would directly relate to the key. However, this classification process is computationally intensive in practice, and the available key spaces make it infeasible to completely search in a reasonable time. Thus an extend and prune strategy to narrow the search space to only those candidates with the highest likelihood of matching the observed key is used [CRR02]. The templates can also be made more efficient by limiting the number of samples examined in the matching process. Determining which samples are most suitable for attack can be done through principal component analysis [ReO04], or through attacks such as leakage mapping that reveal which samples leak the most information [Cob11]. From an information-theoretic perspective, template attacks represent the strongest type of side-channel attack [SMY06]. In practice template attacks are extremely powerful, but depend entirely on the quality of the available templates.

### 2.3.3 Countermeasures

Prevention of SCA attacks is an open problem and a substantial amount of research is devoted to developing countermeasures. A very basic preventative approach is to either increase the amount of random noise in the system or decrease the power of the vulnerable signals. For most attacks this simply increases the number of traces required to successfully retrieve the key, but the exploitable leakages still exist [Cob11].

Asynchronous clocking or randomized clock periods also fall into this category, such that the same operations occur at different times in collected traces [FMP03].

A popular countermeasure uses masking or blinding to separate sensitive information from any physically observable information. Masking randomizes sensitive intermediate values by employing a secret sharing technique [CJR99]. In this way, information about the intermediate values is still leaked, but the intermediate values are independent of the secret key. Incorporating masking into many modern ciphers requires significant modification to the algorithm or hardware to properly handle the new intermediate values. Even then, masking is still susceptible to high-order differential [PRB09] and template attacks [OsM07].

Another type of countermeasure keeps observed power and EM emissions constant such that the variations in the signals can no longer be exploited. This can be done at the transistor [TAV02] or gate [TiV04] levels of hardware design for a cryptosystem, with transistor-level approaches generally having a higher degree of success but are much more difficult to implement. Either strategy attempts to utilize dual-rail logic such that for every intermediate bit within a cryptosystem, an exact complement to that bit also occurs. In this way, every possible data-dependent operation within a

cryptosystem theoretically requires an equivalent power draw. In practice, small differences in gate timings and capacitances still leak valuable information, although the devices are much more difficult to attack [SuS06, MPG05]. The combination of dual-rail logic and some other countermeasures, such as masking, has been shown to be insecure [ScT07].

## **2.4 Advanced Encryption Standard**

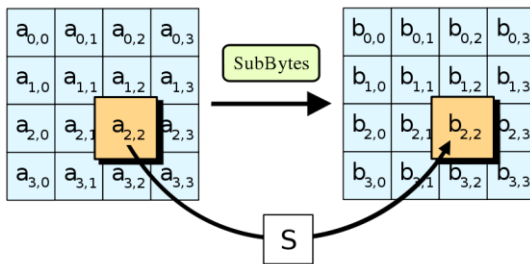
The Advanced Encryption Standard (AES) selection process began in 1997 with the goal of establishing a new cipher to replace the Data Encryption Standard (DES). An important part of the process was that all ciphers submitted as candidates were open to the public, allowing for a wide examination by cryptographers worldwide. The Rijndael cipher was selected as the final candidate in 2000 [Fip01]. AES and Rijndael are thus treated as synonymous throughout this document. AES is widely used in modern systems and is approved for securing classified information by the NSA [Nsa03].

### **2.4.1 General Structure**

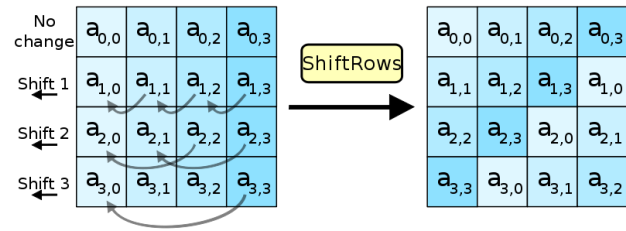
AES is a block cipher capable of handling 128 and 256 bit block sizes with key sizes of 128, 192, or 256 bits. It is not a feistel cipher; a separate decryption process is required to decrypt ciphertexts [Fip01]. A variety of operational modes are available depending on the type of data being encrypted. Electronic Codebook (ECB) is the most common mode used in practice and operates on each block submitted to the system independently. Counter (CTR) and Ciphertext Block Chain (CBC) modes are also used, and each introduce some new element to the input block before being processed by the cipher. These modes introduce dependencies on previous data the cipher has processed so

plaintexts do not directly map to ciphertexts. Use of these modes requires extra measures to preserve proper ordering of the data.

The AES algorithm itself is an arrangement of multiple rounds depending on the block and key sizes. AES-128 uses block and key sizes of 128 bits, and are used as the basis for all further discussion in this document. In this configuration, AES proceeds through 10 rounds of primitives to diffuse and confuse the input plaintext block to a corresponding ciphertext block. A unique round key is generated for each round from the initial full key through the Rijndael key schedule process, but this is not discussed in this document. The plaintext and key bytes are arranged in a 4x4 column-major matrix for the purposes of the cipher; each intermediate state is arranged this way as well [Fip01].



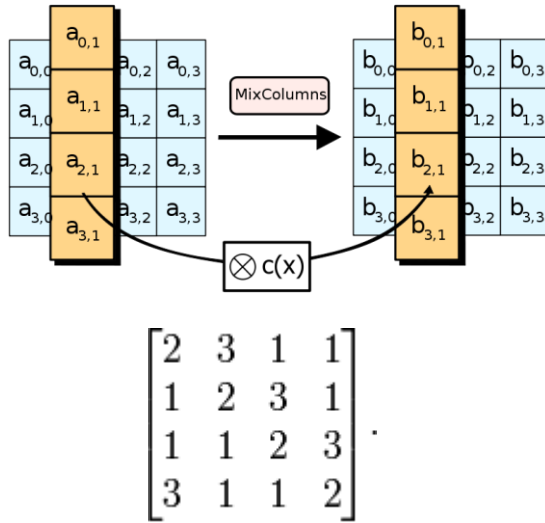
**Figure 2.5: SubBytes Stage of AES**



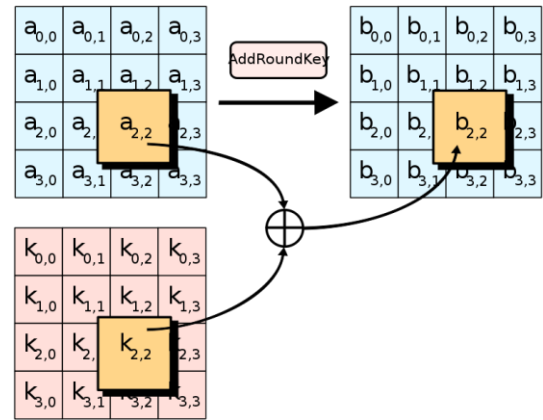
**Figure 2.6: ShiftRows Stage of AES**

The rounds themselves are composed of four unique primitives, with the exception of the final round which only has 3 of the primitives. The first primitive is called the SubBytes (SB) step (Figure 2.5) [Wik12], which performs a non-linear transformation of the input bytes. Next ShiftRows (SR) is applied (Figure 2.6) [Wik12] to the rows of the state matrix to diffuse the data. MixColumns (MC) is the third stage (Figure 2.7) [Wik12], which interprets each element of the columns of the state matrix as part of a polynomial expression. The final step is AddRoundKey (ARK) (Figure 2.8)

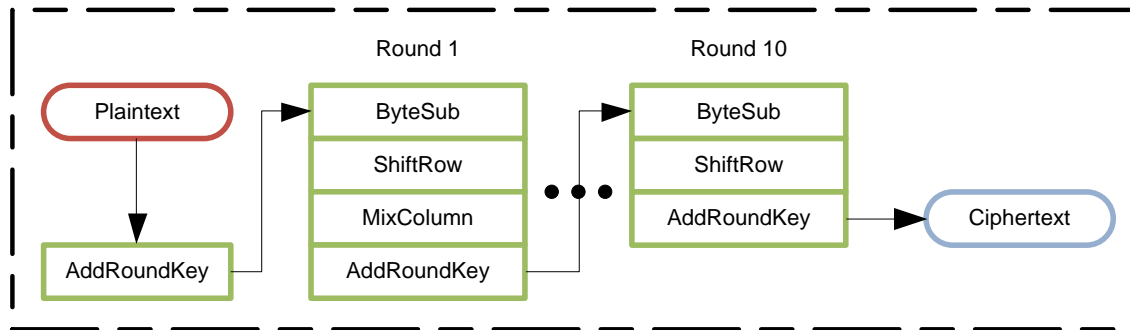
[Wik12], which XORs the current state matrix with the current round key. The full AES process is illustrated in Figure 2.9. The construction of these primitives is discussed further the following section.



**Figure 2.7: MixColumns Stage of AES**



**Figure 2.8: AddRoundKey Stage of AES**



**Figure 2.9: General Structure of AES-128**

### 2.4.2 Primitive Architectures

Each individual primitive can be implemented in a number of different ways depending on the application and platform. For the purposes of this research, only hardware construction styles are discussed. To that extent, both the ShiftRows and AddRoundKey stages are extremely simple in hardware. ShiftRows simplifies to byte routing and AddRoundKey is a block-wide XOR [RSD06]. Both of these operations are very straightforward and additional computation styles introduce unnecessary complexity. Thus the alternate styles are not discussed. Additionally, no decryption architectures are discussed.

SubBytes, at its core, transforms the input byte into its multiplicative inverse under  $GF(2^8)$  and applies an affine cipher to it. Calculation of the multiplicative inverse is done using the Extended Euclidean Algorithm or other methods. Application of the affine cipher simplifies to the linear equation

$$\begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad (2.2)$$

where  $\mathbf{x}$  is the bit vector of the multiplicative inverse. Calculating both the multiplicative inverse and the affine transformation for every byte is a time-consuming process, so often the values are pre-calculated and stored in a look-up table (LUT). LUTs provide constant access time and are very simple to implement, but require additional logic or memory for storing the pre-computed values [RSD06]. There are many other computation methods



but nearly all of them are extensions of the input mapping (LUT) and direct calculation (composite field arithmetic) styles. Some examples include [MoS02] and [SRQ03]. Only the archetypical styles are used for this research.

MixColumns performs a linear multiplication of each column in the AES state matrix, or

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (2.3)$$

This can be simplified to

$$A_0 \oplus A_1 \oplus A_2 \oplus A_3 \oplus 2(A_x \oplus A_{x+1}) \oplus A_x = B_x \quad (2.4)$$

which requires only one multiplication [RSD06, LiF05]. In 2.4, each A represents one of the input bytes, and B represents the corresponding output byte. This is simply multiplication under  $GF(2^8)$ , which is a complex operation different from normal multiplication. Since the multiplication only uses a factor of 2, the multiplication can be performed by a left bit shift with a conditional check for byte overflow. If the byte does overflow, an XOR of 0x1B is applied to the result of the shift. Alternatively, all possible values for the Galois multiply can be pre-computed and stored as a LUT, similar to SubBytes. This again leads to two primary archetypes: input mapping and direct calculation.

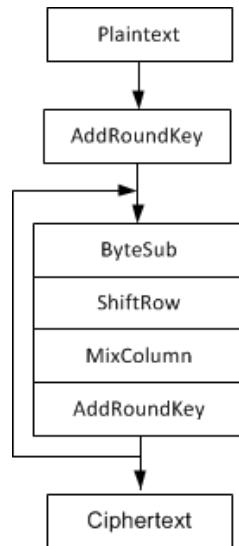
A special primitive combines the SubBytes and MixColumns stages into a single stage, called a TBox [MoS02, KaA10]. Through a re-organization of the stages such that ShiftRows is applied first and the transformations for SubBytes and MixColumns occur immediately thereafter in order. In hardware, this amounts to LUTs to transform the input

bytes (SubBytes) as well as a separate set of LUTs to transform the input bytes directly into the product of their SubBytes transformation and 2 or 3 under  $GF(2^8)$ . All of these values are used in XOR chains to compute the final correct value that would normally emerge from an equivalent MixColumns stage.

### **2.4.3 High-Level Architectures**

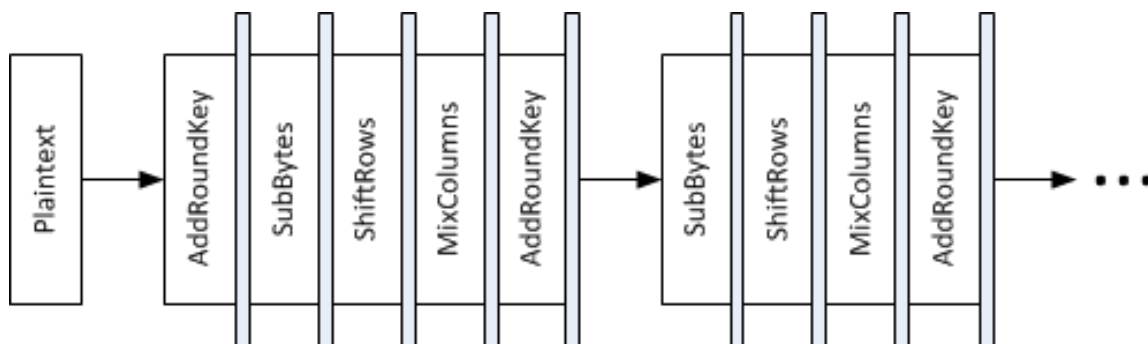
The construction of an AES cryptosystem is flexible so long as it produces ciphertexts consistent with the original algorithm description. The higher-level design of the cryptosystem determines the overall dataflow, as well as its operating characteristics (such as power consumption and throughput). Several common architectures are used in practice depending on the requirements and restrictions of the overall device. Note there are many other architectures for both the high-level flow and the primitives, but the types discussed in this document represent the major archetypes and most other styles can be classified as derivatives.

The first major architecture follows an iterative dataflow. This architecture re-uses resources at the cost of overall throughput. This can be done a number of ways for AES. Figure 2.10 illustrates an iterative architecture with round-based iterations. A finite-state machine can also be used to direct the dataflow to appropriate hardware as required by the AES algorithm. However, since available resources are limited, the throughput is also limited. Most iterative architectures can only operate on a single block of plaintext; that is, it must completely finish processing the block before another can be submitted [RSD06].



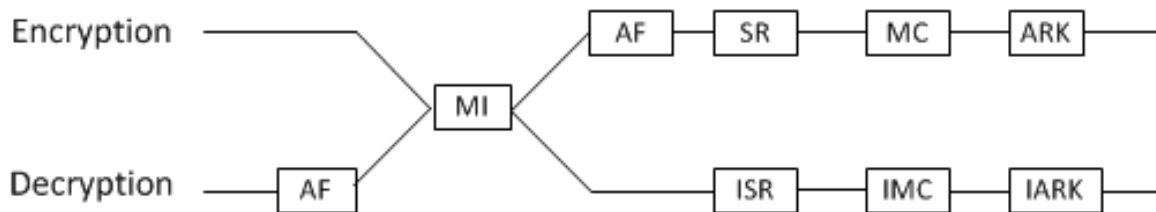
**Figure 2.10: Iterative Architecture for AES.**

A pipelined or fully-unrolled architecture is an alternative to an iterative architecture. A pipelined solution provides distinct hardware for every stage of AES with specific registers between each stage [SRQ03]. This allows the system to simultaneously process as many plaintext blocks as there are stages, and produces one ciphertext every clock cycle in steady state. This is very advantageous for increasing throughput, but the resource costs are considerable. An example of a pipelined AES structure is presented in Figure 2.11.



**Figure 2.11: Pipelined Architecture for AES.**

The final architecture considered combines the encryption and decryption datapaths into a single cryptosystem by having the initial plaintext follow both the encryption and decryption datapaths. A multiplexer is used at the end of each round to determine whether the encryption or decryption datapath inputs are used in the following round. This architecture is preferable in situations where both encryption and decryption capabilities are required in the same system. This can be implemented in either an iterative or pipelined style. The decryption primitives are entirely different structures from their encryption counterparts so very little between them can be re-used. However, if the SubBytes stage is split into separate multiplicative inverse and affine cipher steps, then the multiplicative inverse hardware can be re-used between both datapaths [RSD06]. In this case, the multiplexer is instead placed before the inverse step as shown in Figure 2.12.



**Figure 2.12: Combined EN/DE Core for AES.**

#### 2.4.4 Susceptibility to SCA

Many different AES implementations have been shown to be weak against SCA techniques [SOP04, MaS06, RSV09]. The information that leaks varies by the system, but secret keys can be fairly easily recovered on unprotected systems. Protections are the most common countermeasure added to the system, but research into secure architectures

has been limited. In software, proper secure design practice necessitates building security into the core of the design and not simply adding in security measures after the design is finished [SaS75]. This also holds for hardware security. Much of the research into alternate AES architectures have focused on efficiency and performance metrics [SRQ03], but very little has been examined as to the relative resistance of different architectures to SCA.

Standaert et al. investigate the effects of pipelining AES against SCA in [SOP04]. The theory and results showed that pipelining itself does not serve as an adequate protection, but any increase in the number of simultaneous processing elements increases the overall noise within the system thus making the system more difficult to attack. This is in line with the formal model for cryptosystems discussed in Section 2.2.1. Thus, while pipelining is not a complete countermeasure, it may provide additional protection through noise generation.

## **2.5 Summary**

SCA has emerged as a major threat to many modern cryptosystems by attacking physically observable phenomena during the processing of secret information within a cryptosystem. Formal models of information leakage extend to leakage functions that capture the information within the full signal. This can be done through manual comparison of traces, correlation attacks, and template attacks. Countermeasures such as masking, asynchronous clocking, and dual-rail logic styles have been introduced to combat these leakages, but none have been completely successful in eliminating information leakage.

The structure of an AES-based cryptosystem varies in terms of its primitive and high-level architectures. Each different architecture produces the same output data, but different operations occur on the input data. Limited research has been done to investigate how different architectural styles affect information leakage. Using different architectures is likely to affect the magnitude of information leakage, but it is unclear whether this difference itself can be considered a countermeasure.

### **III. Methodology**

This chapter describes the process used to achieve and evaluate the research goals. The problem is clearly defined, and limits are placed on the scope of evaluations. Parameters and factors are discussed with reference to their effect on the system. An experimental setup is described with specific details relating to equipment and the measurement process. Evaluation metrics are also selected. Finally, the experiment is justified within the bounds of its expected results and the goals of the research.

#### **3.1 Problem Definition**

This section describes the higher-level goals and approach for this research. The hypotheses and general problem are defined to motivate design decisions in the formulation of the experiments. The overall approach of the experiment is presented so that the experimental tasks can be correctly interpreted.

##### **3.1.1 Goals and Hypothesis**

The primary goal of this research is to determine the effectiveness of a FPGA-based AES implementation designed to be resistant to side-channel attacks. The main premise is that countermeasures to side-channel attack aside, there are certain designs that leak less information through physically observable phenomena. That is, it is presumed the underlying architecture of the algorithm can significantly affect its emission characteristics. Analyzing any remaining leakage can then be used to develop specific countermeasures to further enhance a device's security.

The signal-to-noise ratio (SNR) is an effective way to characterize how prone a system is to exploitation. The goal is to make this value as low as possible, which in

general can be accomplished by lowering the signal strength or increasing the noise. This is significant for side-channel attacks in that a doubling of noise squares the number of traces needed to successfully attack the system [Sch00]. Different hardware configurations generate different amounts of full-system noise and target signal strength, so it is advantageous to approach the problem from this viewpoint.

For hardware systems, changing the architecture results in different configurations of the transistors and thus different power and electromagnetic signatures. These are the signals most commonly exploited in side-channel attacks. Thus, a primary goal is to determine the best configurations of AES primitive elements that limit the vulnerability of those signals. The secondary goal is to determine whether or not this reduction in information leakage is sufficient to prevent an attacker from extracting secret information. The overall best configuration is expected to be a simple hardware configuration due to the simplicity of the logical operations resulting in a smaller electrical footprint. This hypothesis is thoroughly tested over the course of the experiments.

### **3.1.2 Approach**

To determine the effectiveness of a SCA-resistant AES configuration, this research performs correlation attacks against each configuration and observes the resulting magnitude of information leakage. The different configurations include different hardware architectures for the Substitute Bytes (SB) and Mix Columns (MC) stages of AES, as described in Section 2.4.2. The remaining stages are not considered because their default configuration is logically simple, consisting of only basic digital logic blocks such



as XORs or wire routing. All designs are compared in terms of the magnitude and variety of information leaked.

Leakage maps [Cob11] are used to assess the correlation of each time sample to each expected intermediate value which are calculated with full knowledge of the secret key to examine the scenario in which the attacker correctly guesses the corresponding bits and bytes; that is, the worst-case scenario for a defender is assessed. Both power draw and electromagnetic emissions are observed and processed with Hamming Weight and Hamming Distance leakage models. The Hamming Distance model is created with full knowledge of the underlying architecture, again assuming the worst-case scenario for the defender.

### **3.1.3 Adversarial Context**

To specify the bounds of how the device can be attacked, a proper attack scenario must be established. It is assumed that the targeted device is in constant operation and cannot be removed or replaced by the attacker without alerting the operators. The attacker has obtained a prior piece of encrypted data encrypted by the target device, and his goal is to obtain the secret key so that the information can be decrypted. The input and output of device can be observed at any given time, and the attacker can observe electromagnetic emissions and power consumption of the device.

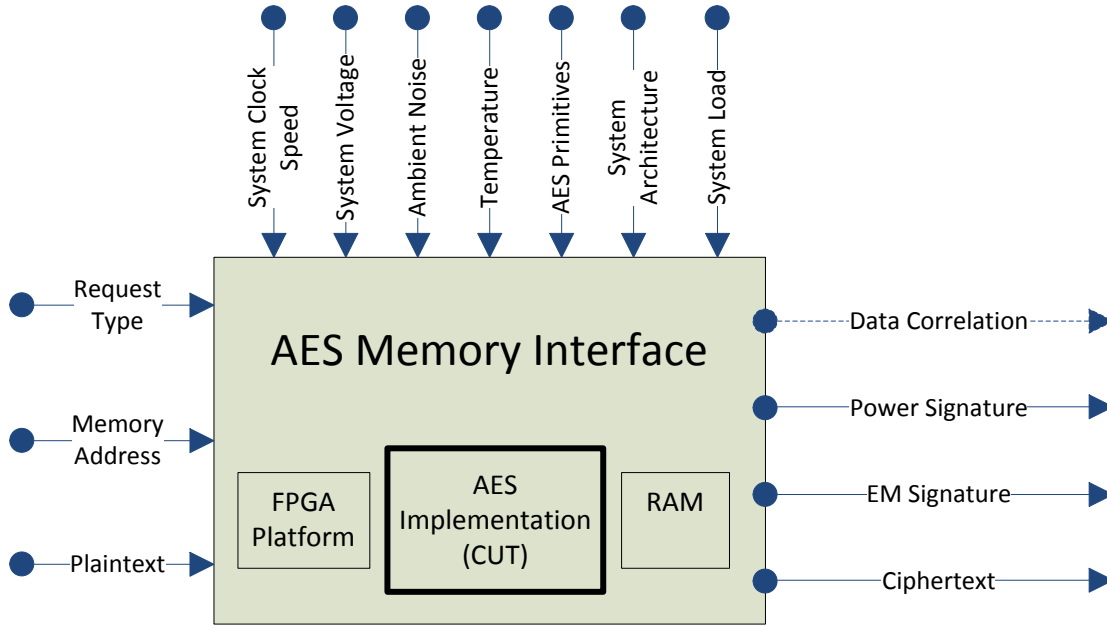
To summarize, the attacker cannot physically alter or disrupt the operation of the device. This means that invasive attacks cannot be performed. He does not have the ability to select plaintexts (cannot perform chosen plaintext attacks), nor can he control the rate at which encryption requests are sent to the system. Finally, the attacker has complete knowledge of the plaintext/ciphertext pairs. From an information-theoretic

standpoint this means that the key can be determined, although it is infeasible to calculate given finite time and resources.

### **3.3 System Boundaries**

The System Under Test (SUT) is the AES Memory Interface. This includes the hardware implementation of AES itself, its interaction with Random Access Memory (RAM), and the FPGA that serves as the execution platform. The focus of this experiment is the AES implementation; that is, it is the primary Component Under Test (CUT). The RAM blocks and FPGA are left as generic devices; that is, no FPGA device-specific features are used that would prevent an equivalent implementation from being realized on a different brand or model FPGA. The AES implementation is limited to a 128-bit Electronic Codebook (ECB) variant. While there are many AES operating modes and key lengths, this variant is the most suitable for parallelization. Additionally, it is the most common operating mode in practice.

Figure 3.1 shows the full SUT with its workload, parameters, metrics, and responses. Although the CUT is the AES implementation, it is not its functionality being tested; that is, its throughput, area, power consumption, and similar characteristics. Instead, the intrinsic factors such as electromagnetic emissions and current draw during operation are tested. Note that in the context of this experiment the RAM is only notional; the actual storage of the ciphertexts is not important in the evaluation of the CUT, as it is considered data at rest and is therefore outside the scope of the attack.



**Figure 3.1: AES Memory Interface**

### 3.4 System Services

This system provides encryption, storage, and decryption of data. Each of these functions is a service of the system. Encryption simply takes plaintext and applies the AES algorithm to it, which produces ciphertext that can only be feasibly recovered with knowledge of the secret key used to transform it. This encrypted data is saved to blocks of RAM and can be recalled for subsequent decryption. This interaction with memory is controlled by the target address space for both encryption and decryption services. Decryption takes ciphertext and converts it to plaintext based on the secret key. Within the system itself, this is performed by taking the data stored in memory at the target address, running it through the AES algorithm, and returning the plaintext to the user or other systems.

The system does not provide authentication, which would validate the encryption/decryption requests. Without authentication, any and all requests made to the system are executed. This enables an attacker with access to the device to decrypt the stored data or overwrite it if the key is pre-stored on the device. Another service not present in the system is data integrity. Bits may be corrupted in rare cases due to excessive noise or attacks on the system causing encryption or decryption services to produce incorrect results and possibly leak more information about the key. No service checks whether bytes are altered when performing the other services. Both authentication and integrity verification are outside the scope of this experiment.

### **3.5 Workload**

The primary workload submitted to the SUT is random plaintext queued for encryption. This plaintext is transformed into ciphertext and stored in the RAM. The stored ciphertext can be recalled and decrypted as well, but the encryption and decryption processes are not run simultaneously. Plaintext is required to be sufficiently random to draw correlations between the bits, bytes, or words as they travel through the system. The secret key is fixed for individual sessions of workload submission, but it is not required to be fixed for distinct workload submissions. Even so, the primary key is not changed throughout these experiments as there is no immediate benefit of varying the key for this research.

Requests to the system are comprised of the plaintext to encrypt and the desired operation. Alternate operations include setting a new secret key and simultaneous submission of a new key and plaintext. Setting a new key completely overwrites the prior

key. Decryption requests and operations are not considered in this research. Random plaintexts are initially generated by a seeded linear feedback shift register (LFSR). Once the system produces a ciphertext, that ciphertext is fed back to the system as plaintext in place of the LFSR plaintexts to produce plaintexts fast enough to keep the system under full load while still allowing each plaintext to be reproducible.

### 3.6 Performance Metrics

Correlation-based DPA attacks relate the overall correlation of expected variations to observed variations. Leakage maps are a collection of the correlations for each state of a cryptosystem against each observed sample of an encryption. These maps are useful in determining locations of information leakage for each stage of a cryptographic algorithm, but can also be used to assess the system's actual susceptibility to correlation attack. Maps for both power draw and electromagnetic emission measurements are analyzed using the Hamming Weight and Hamming Distance leakage models described in Section 2.2.2. Results are deemed to be significant if 0.0 is not within the confidence interval of the correlation coefficient. The confidence interval for  $N$  traces and  $1-\alpha$  confidence is

$$\rho_{max} = \frac{\exp\left(\frac{Z_{(1-\alpha)}}{\sqrt{\frac{N-3}{8}}}\right) - 1}{\exp\left(\frac{Z_{(1-\alpha)}}{\sqrt{\frac{N-3}{8}}}\right) + 1}. \quad (3.1)$$

This value can also be interpreted as the maximum observed correlation coefficient to assert  $N$ -trace resistance to DPA [Cob11].

Leakage maps are assessed in terms of the maximum correlation coefficients observed across the individual primitives with a focus on the rounds and stages most susceptible to attack. Both individual bits and bytes are examined in this way. The minimum, maximum, and average maximum correlations for each stage are reported. Additionally, the magnitude of the average maximum leakages are assessed in terms of the likelihood those samples can be exploited in a correlation attack. Table 3.1 lists the maximum observable correlation coefficients to assert SCA resistance for 1,000,000 traces with an accompanying statement of likelihood based on (3.1). These are meant to provide a general idea of the significance for each confidence level given the sample size of 69,700 correlation coefficients calculated for each leakage map.

**Table 3.1: Correlation Confidence**

Confidence Level	Maximum observed correlation	Assessment
90%	.0018	No exploitable correlations.
99%	.0033	Weak correlations.
99.9%	.0044	Moderate correlations.
99.99%	.0053	High correlations.
99.99%+	>.0053	Exploitable correlations.

### 3.7 System Parameters

A number of parameters affect performance of the SUT from the viewpoint of the goals of this research. The most prominent parameters are described below, as well as assumptions being made about those parameters.

Secret Key – Using a different secret key within the system results in different interactions between the input plaintexts and the intermediate hardware. This effect is not expected to have a large impact on the results, but different keys have the potential for

weakness in different bits and bytes. The secret key is kept constant throughout all experiments so that results are directly comparable.

System Clock – For any system based on transistors and digital circuits, the system clock plays a role in synchronizing the components and data flow. However, the characteristics of transistors can behave differently at different clock speeds. For example, higher clock speeds put more strain on the electronics to the point where they may behave erratically. A varying clock speed would affect some of the EM or power analyses, and thus is kept constant throughout all experiments.

System Voltage – Similar to the system clock, adjusting the input voltage level of a circuit puts additional strain on the transistors, and changes its observable characteristics. In particular, increasing the voltage for a digital circuit results in higher EM emissions and a more powerful signal. Input voltage is kept constant for all experiments.

Temperature – Temperatures, both ambient and circuit-produced, affect how electrons flow within the circuit. Increases in temperature can result in a significant reduction in circuit performance, and further affect its outward characteristics. Ambient temperature is kept roughly constant by keeping consistent laboratory environments during measurements.

Ambient Noise – Noise strongly affects measurements by changing the total SNR. Introduction of excessive ambient noise can result in misleading interpretations of the measurement data. Such noise is kept relatively constant by performing the measurements in a laboratory environment. Additionally, most ambient noise is expected to be eliminated through the correlation calculations.

AES Primitives – For this effort in particular, the selection of AES primitives has a large effect on the resulting signal strength and system-generated noise. Using different hardware configurations for the primitives allows variation of the system’s outward characteristics, as well as affecting its throughput performance.

System Architecture – This parameter is the system’s high-level architecture, specifically its circuit complexity or length of pipeline. Altering this parameter affects the observable circuit characteristics as well as the performance, much in the same way as the AES primitives. The system uses a fully-unrolled pipeline configuration for all experiments, with full byte-level parallelism for each AES stage.

System Load – The system’s relative loading at the time of measurement affects its outward characteristics significantly. For a pipelined system, a system under full load would be one in which data was currently being processed in all stages of the pipeline. Since the attacker has no control over the rate of encryption requests, the system is placed under full load during testing.

### **3.8 Factors**

The previous section described all system parameters, but only a few of the parameters are varied over the course of this experiment. These factors are discussed in greater detail below. The different levels are succinctly described in Table 3.2.

AES Primitives – Primitives form the low-level architecture of the system and describe the transformation of the input to the necessary output. The transformation process for certain primitives can be performed in myriad ways, however, and that forms the main source of variation for this factor. These different hardware configurations are expected



to alter the resulting characteristics of side-channel leakages depending on which are implemented. Specifically, the SB primitive is examined in LUT and composite field arithmetic configurations, and the MC primitive is checked between its LUT and Galois multiplication configurations. Each combination of these primitives is tested for its effectiveness in reducing information leakage.

System Architecture – This factor refers to the higher-level structure of how the primitives interact with one another. Typical configurations are either iterative or pipelined. This factor is varied between a simple fully-unrolled architecture and a more complex combined encryption and decryption style. The difference between these two architectures is the presence of additional hardware components in the combined style; these additional primitives still operate on intermediate data and are expected to generate additional system noise. This factor is varied in a secondary phase of the experiment after the best primitive types have been determined.

**Table 3.2: Factors and Levels**

Phase	Factor	Levels
1	SubstituteBytes Style	{Look-up Table, Composite Field Arithmetic}
1	MixColumns Style	{Look-up Table, Galois Multiplication }
2	High-Level Architecture	{Standard Pipelined, Combined EN/DE Pipelined}

This experiment initially intended to cover additional factor levels, including TBox styles for the SB and MC primitives and an iterative high-level architecture. The code and interfaces for these devices have been created, and simulations show that these other architectures operate as expected. However, implementations of these designs produce invalid results due to hardware synthesis errors (likely due to the synthesis

engine used). These errors could not be fixed as of the writing of this document, and thus those configurations are excluded from the experiment.

### **3.9 Evaluation Technique**

The most suitable evaluation technique for these experiments is the direct measurement of a real system. Simulated or analytical models are not suitable since the underlying electrical properties of the digital circuit would be much more difficult to model vice simply implementing and measuring the circuit.

Measurements are obtained using the Riscure EM probe and an attached oscilloscope. Both EM and power traces are taken simultaneously, with 1 million encryptions observed per experimental configuration. A SASEBO GII FPGA with a Xilinx Virtex 5 core is the execution platform. Configurations are varied according to the levels and listed in Table 3.2. The collection process is triggered via a hardware cue programmed into the system. These initial measurements are sampled at 5 gigasamples per second to account for internal processes occurring at up to 2500 MHz, however the FPGA is only clocked at 100 MHz. The initial seed and request is sent to the system over through a UART (serial) connection. New measurements are taken for each new configuration. Several pilot measurements are also done on configurations to properly set up analysis tools, but are not used towards the full trace sets of those configurations.

Measurements are validated through simple power/EM analysis techniques including averaging, bit correlation, and noise filtration. The goal for this type of validation is to ensure that the data is consistent over the course of trace collection. The correctness of an AES core is verified by comparing the generated plaintexts/ciphertexts against known

good plaintext/ciphertext pairs. These correct pairs are obtained from the implementation in the AESObject MATLAB library, which is distributed as part of the OpenSCA toolbox.

### **3.10 Experimental Design**

The experiment is a full factorial design in two phases. Testing in this manner leads to the best overall design in terms of SCA-resistance, as well as allows some insight into why certain designs leak less than others. This results in only 5 total experiments; 4 in phase 1 to select the best configuration of primitives, and one additional experiment in phase 2 to test the effects of other high-level architectures. Each of the experiments is measured in two distinct ways, and each type of measurement is analyzed with two distinct leakage models. This leads to four distinct analyses for each experiment, resulting in 20 total analyses.

Variance in the resulting maximum correlation coefficients is handled according to the assessment proposed in Table 3.1. Analyses that show moderate to high correlations are potentially exploitable, and maximum correlation coefficients above that range are considered to be weak to SCA attacks. Replications of the experiments are not performed because it is assumed that errors due to environmental noise are negligible after the correlation process. A confidence interval of 99% is used in determining statistical significance of results from  $t$ -tests.

### **3.11 Methodology Summary**

The primary approach in this effort is a full evaluation of suitable AES primitives followed by an examination of differences due to high-level structure. This provides

direct insight into the effects of each primitive as well as their interactions across common measurement types and leakage models. Varying the high-level architecture reveals whether different design styles at that level have a meaningful impact on information leakage. The end result indicates which primitive implementations and architectural styles have the best security characteristics, as well as describe the relative security of the full AES system.

While many other parameters affect the electrical characteristics of the system, factors are limited to the architectural decisions since those are the factors that can be enforced on all such devices based on them. The remaining parameters are left to the attacker's discretion and cannot be controlled once the device is released.

The analyses of these results provide further insight into best practice of architectural style for low- and high-level views of a system when the goal is resistance against side-channel attack. These results are not limited to the AES algorithm and its implementations but also apply to a broad range of cryptographic and communication devices.

## **IV. Results and Analysis**

This section performs a systematic analysis of the experimental results. The results are analyzed at several different levels, starting with bit leakages and continuing to byte and full leakage analyses. Each level is analyzed based on the maximum points of information leakage between the different primitives, individual rounds, and leakage models. Each stage of analysis explores both the power and electromagnetic leakages. Note that configurations are abbreviated according to the styles used for each component, with the SB variant listed first and the MC variant listed second. Styles implemented using look-up tables are abbreviated LUT, and composite field arithmetic styles are abbreviated GAL. Thus, LUT-GAL indicates a configuration using look-up tables for the SB primitive and Galois multiplication in the MC primitive. All references to the confidence in a given value are based on Table 3.1. Higher confidence levels give a higher likelihood the target sample is not a false-positive. Thus, attacking a device with a lower confidence level is more likely to introduce errors and increase the amount of brute-forcing an attacker must use to recover the secret information.

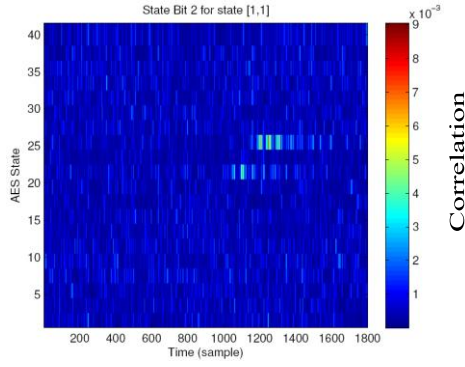
### **4.1 Effects of Primitive Architecture Variation**

This section analyzes how different configurations of the primitives affect the total leakage characteristics of the implementation. This analysis is performed separately at the bit and byte levels. The best configuration of primitives is selected based on the results of these analyses, and is used in the secondary phase of the experiment to determine the effects of high-level architecture variation.

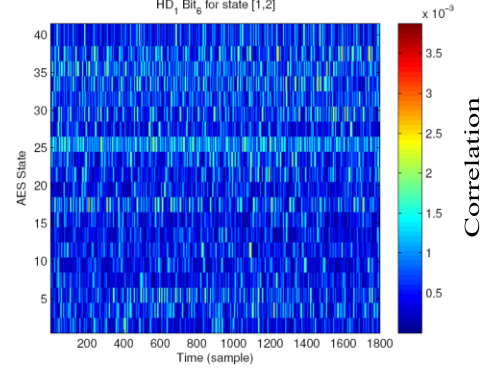
#### 4.1.1 Bit Analysis

An attacker desires a high correlation between the expected intermediate values and the observed variations in the signal. This can be done at several different levels, with bit correlations as the lowest and most direct level. Individual bits are attacked by positing a hypothesis for the bit values and analyzing how closely the observed data correlates to it. This works best in attacking registered data after the initial add round key stage, such that at most 256 hypotheses must be tested (2 for each bit in the key). More strenuous analysis can be used to attack later rounds of the algorithm, but the best case for an attacker is that the first ARK stage leaks sufficient information to attack. Alternatively, the final ARK stage can be attacked similarly with the ciphertext instead of the plaintext, but this attack can only determine the 11<sup>th</sup> round key and not the full original key.

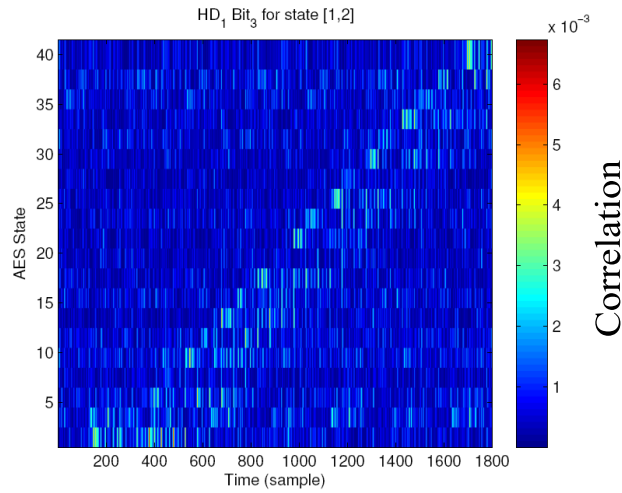
Leakage maps for individual bits on each of the architectures reveals several trends in the data. For the EM measurements, neither of the leakage models was able to capture the general structure of the algorithm for each round. Samples are shown in Figures 4.1 and 4.2. Only the EM measurements processed under the HW leakage model showed subtle indications of the later rounds. This initial inspection shows it is likely unreasonable to exploit individual bits throughout the algorithm, regardless of the architecture. The power measurements analyzed with the HD model in Figure 4.3 revealed a clearer view of the system structure, but was still limited for many of the bits.



**Figure 4.1: EM HW Leakage Map**



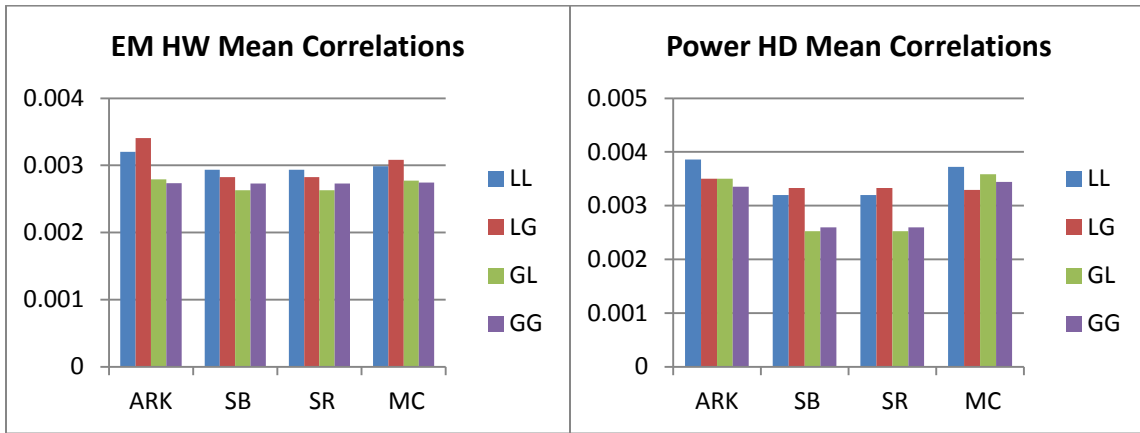
**Figure 4.2: EM HD Leakage Map**



**Figure 4.3: Power HD Leakage Map**

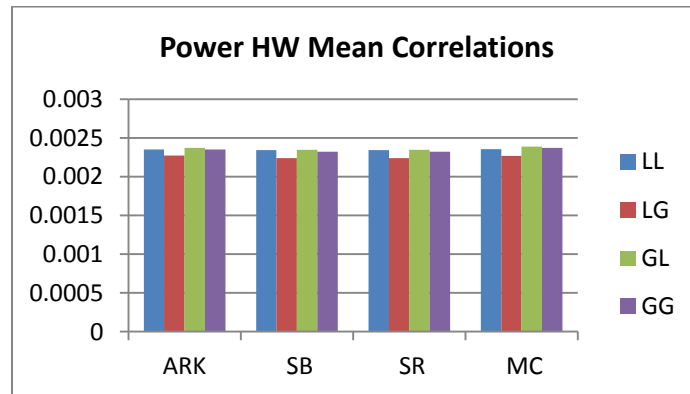
Using each individual state bit as a separate sample, the maximum correlation values for individual primitives are aggregated and compared in terms of their minimum, maximum, and mean values. The full tables are listed in Appendix A. Across all architectures, leakage models, and measurement type, the most significant variations due to architecture were seen in the EM measurements under the HW leakage model (Figure 4.4), and the power measurements analyzed with the HD model (Figure 4.5). The remaining types have only very minor variations from architectures. This is most likely due to the effects of additional system noise due to the architectures. A sample of this is

shown in Figure 4.6; other combinations feature similar variations. Note that all reported correlation values are the maximum observed coefficients for each primitive. The points of maximum correlation are the only points an attacker needs when attacking a system. Thus, references to the average case refer to the mean of the maximum values observed across all bits, and references to best case refer only to the absolute maximum values observed for each stage.



**Figure 4.4: EM HW Average Maximum Correlations**

**Figure 4.5: Power HD Average Maximum Correlations**



**Figure 4.6: Power HW Average Maximum Correlations**

Variation of the architecture affects all stages similarly under the EM HW leakage map, while the power HD correlations show significant variations only in the stages with



architecture changes. Table 4.1 lists the impact of each factor on the individual stages of the power HD and EM HW analyses. These values were obtained through an allocation of variation process. For all models and measurement types, SB and SR stages share equivalent values because the expected values are not actually changed between rounds and are merely routed to the necessary locations. Thus, SR is not listed in the results. The allocation of variance highlights that changing the style of one type of primitive has the most significant effect on that primitive's leakage in the power HD model. It also shows that variation in the ARK primitive is almost equally affected by changing either of the primitives. This is not true for the EM HW model, as variation of the SB component controls the observed variation in each primitive.

**Table 4.1: Bit Allocation of Variation**

<b>Power HD</b>	<b>Factor Impact</b>				<b>EM HW</b>	<b>Factor Impact</b>		
<b>Stage</b>	<b>SB</b>	<b>MC</b>	<b>SB/MC</b>		<b>Stage</b>	<b>SB</b>	<b>MC</b>	<b>SB/MC</b>
<b>ARK</b>	46.57%	45.36%	8.07%		<b>ARK</b>	92.72%	1.72%	5.57%
<b>SB</b>	97.77%	2.02%	0.21%		<b>SB</b>	78.53%	0.07%	21.40%
<b>MC</b>	0.03%	80.15%	19.83%		<b>MC</b>	93.72%	1.59%	4.69%

The initial and final ARK stages are the most likely to be targeted by bit-wise correlation attacks. Examining these stages alone reveals that both configurations with the GAL SB primitive reduce the leakage of the power HD and EM HW analyses for both ARK stages (Figures 4.7 and 4.8). Leakage is reduced by 25.35% in the power HD analysis and 39.75% in the EM HW analysis for the initial ARK; leakage was also reduced in the final ARK by 39.05% and 34.74%, respectively. Varying the MC architecture affected these rounds by less than 5%. This difference reduces the accuracy

of the attack, meaning that the measurements were only weakly correlated to the hypotheses. Leakage was not affected by architectures for the other types of analyses.

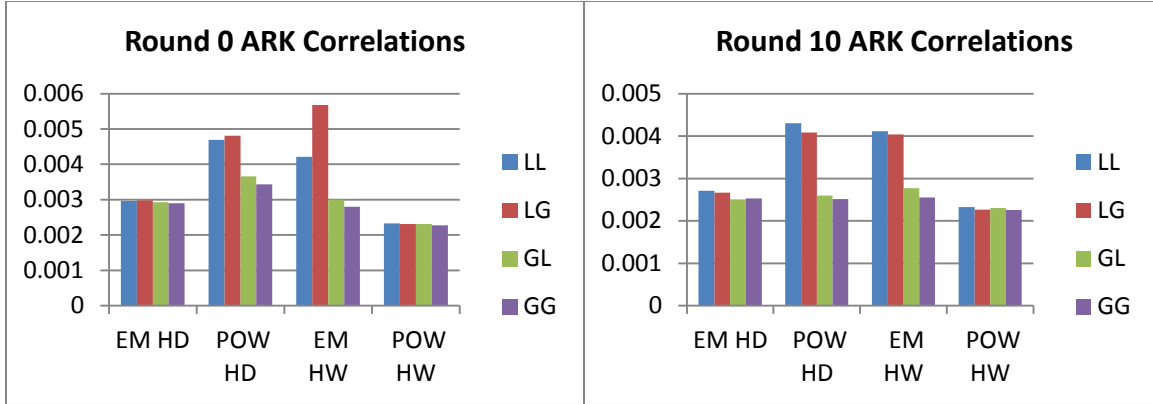
**Table 4.2: Average Bit Correlation Differences by Stage**

<b>ARK</b>	<b>Power HD</b>			
	<b>LL</b>	<b>LG</b>	<b>GL</b>	<b>GG</b>
<b>LL</b>	x	4.047E-04*	5.675E-04	7.706E-04
<b>LG</b>	-4.047E-04*	x	1.628E-04*	3.659E-04
<b>GL</b>	-5.675E-04	-1.628E-04*	x	2.031E-04*
<b>GG</b>	-7.706E-04	-3.659E-04	-2.031E-04*	x
<b>SB</b>	<b>Power HD</b>			
	<b>LL</b>	<b>LG</b>	<b>GL</b>	<b>GG</b>
<b>LL</b>	x	4.602E-05*	1.336E-03	1.230E-03
<b>LG</b>	-4.602E-05*	x	1.290E-03	1.184E-03
<b>GL</b>	-1.336E-03	-1.290E-03	x	-1.064E-04*
<b>GG</b>	-1.230E-03	-1.184E-03	1.064E-04*	x
<b>MC</b>	<b>Power HD</b>			
	<b>LL</b>	<b>LG</b>	<b>GL</b>	<b>GG</b>
<b>LL</b>	x	6.391E-04	1.346E-04*	3.509E-04*
<b>LG</b>	-6.391E-04	x	-5.045E-04	-2.882E-04*
<b>GL</b>	-1.346E-04*	5.045E-04	x	2.163E-04
<b>GG</b>	-3.509E-04*	2.882E-04*	-2.163E-04	x

\*- Not statistically significant within 99% confidence interval

A student *t*-test is used to determine whether the estimated means between different configurations are statistically different. Table 4.2 lists the differences in the averages for the compared architectures under a power HD model where marked values are not statistically significant within a 99% confidence interval. The calculated p-values are detailed in Appendix D. These results indicate that the observed means are different for most configurations across the ARK and SB primitives. Notable exceptions include the LUT-GAL and GAL-LUT comparison for the ARK stage and the LUT-LUT and LUT-GAL comparison for the SB stage. Both of these cases have a limited likelihood of

being different. This is supported by the allocation of variation analysis in Table 4.1; varying either architecture has a similar effect on ARK, while varying the SB primitive has a much stronger impact on the SB means. For the MC analysis this is also true, however the p-values for those cases are much higher compared to equivalent comparisons in SB.

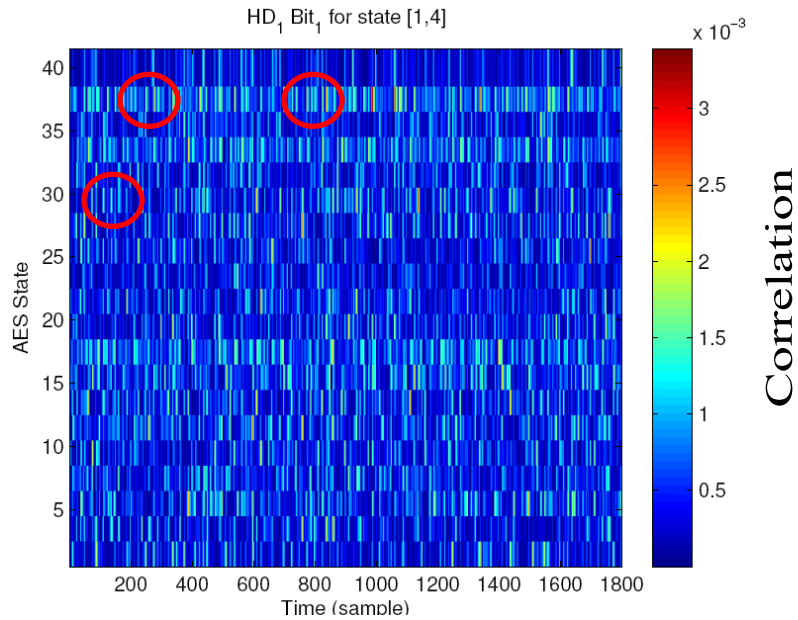


**Figure 4.7: Initial ARK Average Maximum Correlations**

**Figure 4.8: Final ARK Average Maximum Correlations**

Referring to Table 3.1, the maximum observed correlation general case is weak in each stage within a 90% confidence interval, but not within 99% for GAL SB configurations. These confidence intervals relate directly to the accuracy of an attack where lower confidence intervals have a higher likelihood of being false-positives. This means that the implementations have potential weakness to correlation attacks, however it is likely that an attack will contain some errors and the attacker will not be able to obtain the complete key through the correlation attacks alone. This is supported by manual inspection of the leakage maps, which show spikes of high correlation at time samples that cannot contain information about that particular AES state (Figure 4.9). This applies for both the EM HW and power HD analyses, however the EM HW type exhibits

correlations nearly twice as strong as the power HD in the best case. Best case maximum values for EM HW analysis are well beyond 99.99% confidence, meaning that those bits are highly exploitable. The stages most likely to be exploited in a bit-level correlation attack are moderately weak to attack with 99% confidence. Composite field arithmetic SB primitives showed the best effects in lowering information leakage, while GAL MC primitives show mixed results depending on the type of analysis. Thus GAL SB and MC LUT primitives are selected as the best tested primitives at a bit analysis level.



**Figure 4.9: Leakage Map with Inaccurate Correlation Spikes**

Table 4.3 lists the number of bits that can be compromised for a given confidence level. These values assume that only a high correlation value is necessary to exploit any given bit, each stage and round is equally exploitable, and that every possible point of a leakage map is valid. Results are only shown for the GAL-LUT configuration, as that was selected as the best overall configuration at the bit level. With only 90% confidence, all bits are exploitable under the listed assumptions. This drops off with higher confidence

levels, but some of the bits are still exploitable beyond 99.99% confidence, meaning that the target samples in those bits are highly unlikely to be false-positives. If the correct values of 33 bits can be learned by an attacker, the search space for a brute force attack on the remaining bits is reduced from  $2^{128}$  to  $2^{95}$ . This indicates that this implementation is still weak to SCA at the bit level, but it is unlikely that all bits can be cleanly exploited in an attack. Also note that the EM HD and power HW models are ineffective at higher confidence levels compared to EM HW and power HD.

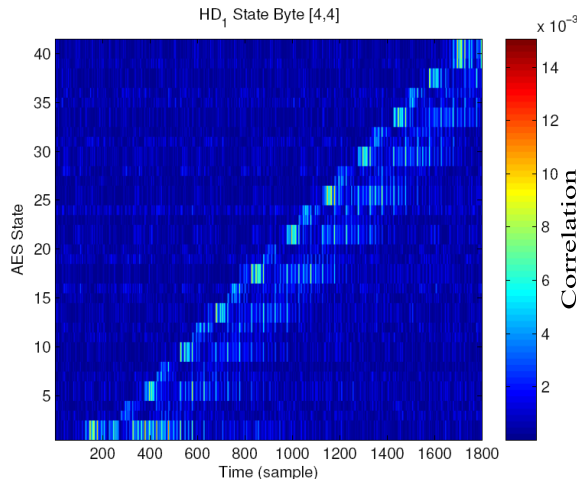
**Table 4.3: Number of Exploitable Bits per Confidence Level**

	Confidence Level			
Model	0.9	0.99	0.999	0.9999
EM HW	128	127	74	30
EM HD	128	120	6	0
Power HW	128	90	0	0
Power HD	128	128	101	33

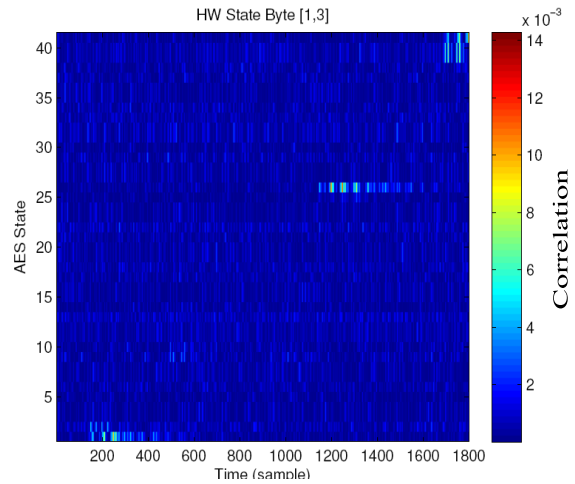
#### 4.1.2 Byte Analysis

Similar to bit analysis, individual bytes of the secret key can be attacked by forming hypotheses about their values and testing these hypotheses through correlation attacks. Byte correlation attacks can be more effective than bit correlation attacks since most AES stages operate at a byte level. Using only simple analysis, the first round can be exploited in each stage up to MC; more advanced analytical processes are required for exploiting later rounds and stages. An attacker needs to perform at most 4096 distinct analyses if the first round can be exploited for each byte (one for each possible byte value for each of the 16 bytes).

The leakage maps for the state bytes show a much clearer indication of the internal progression than the individual bit analysis. This likely indicates that the bytes are exploitable at many stages and rounds. Again, the EM signals show clearer results in the HW leakage model, whereas the power traces give higher correlations under the HD model. Unlike the bit analysis, EM measurements analyzed with the HD model also show some of the AES structure at the byte level. Power traces under the HW model still do not show any clear structure, however. Samples of the leakage maps are displayed in Figures 4.10 and 4.11.



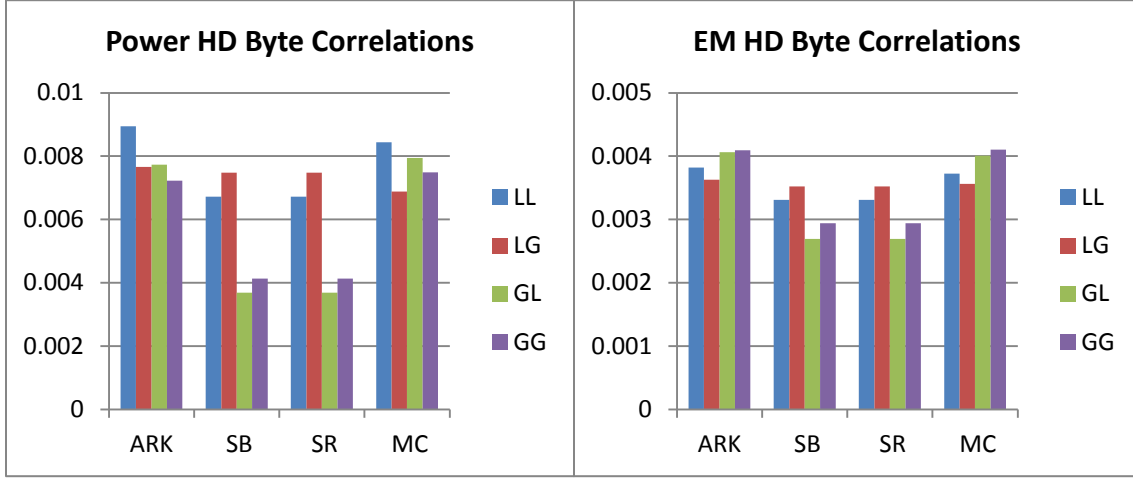
**Figure 4.10: Power HD Byte Leakage Map**



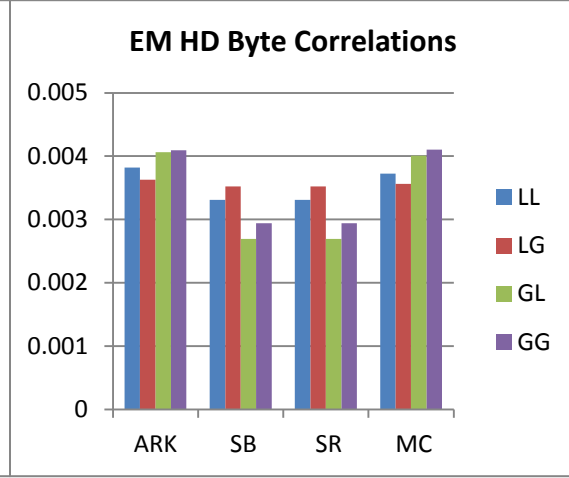
**Figure 4.11: EM HW Byte Leakage Map**

The maximum values across each stage are aggregated for all 16 bytes in each analysis are in Appendix B. These results are visualized in Figures 4.12 and 4.13. Just as with the bit analysis, composite field calculation for the SB stage has a significant impact on the leakage from the SB and SR stages. Varying the architecture of the MC stage also had an effect on the average case, but the results are not as promising. Using Galois multiplication in the MC stage results in lower average leakages for the MC and ARK stages, but actually caused more leakage in the SB and SR stages. The reasoning for this

is likely that the Galois multiplication actually creates less system-wide noise than the LUT variant, while the simplified operations involve less data-dependent operations.



**Figure 4.12: Power HD Byte Average Maximum Correlations**

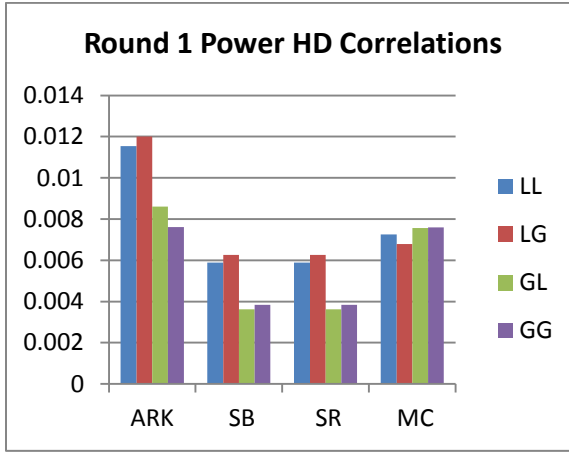


**Figure 4.13: EM HD Byte Average Maximum Correlations**

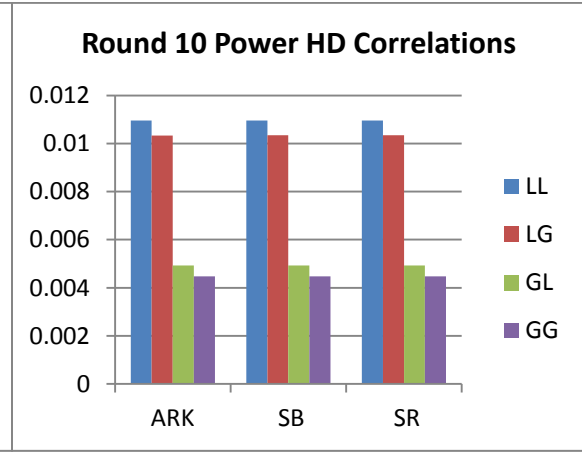
Table 4.4 lists the average maximum leakages for the first and last round primitives; the remaining values are in Appendix C. These rounds are the easiest to attack since the attacker can directly test hypotheses, and thus they are given special attention. Figures 4.14 and 4.15 visualize a couple of the types with fluctuations due to architecture choice. The GAL SB variant again has the most significant impact, reducing leakages in most primitives by 25.02% in Round 1 and 55.88% in round 10 under the power HD analysis. The EM HW analysis also showed benefits, but only 16.80% reduction in round 1 and 37.97% in round 10. Using the GAL MC variant again has mixed results, with slight advantages in round 10 but higher leakage in round 1; observed effects were less than 10% for each primitive and analysis type.

**Table 4.4: First and Last Round Average Maximum Correlations**

Byte	POW	HD	Arch					
	LL		LG		GL		GG	
	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10
ARK	0.01153	0.01095	0.01201	0.01033	0.00860	0.00492	0.00760	0.00447
SB	0.00588	0.01096	0.00626	0.01033	0.00362	0.00493	0.00383	0.00447
SR	0.00588	0.01096	0.00626	0.01033	0.00362	0.00493	0.00383	0.00447
MC	0.00725		0.00679		0.00756		0.00760	



**Figure 4.14: First Round Power HD Average Maximum Correlations**



**Figure 4.15: Last Round Power HD Average Maximum Correlations**

Allocation of variation analysis and  $t$ -tests for the byte level show similar results as in the bit-level analysis, with a few exceptions. The most significant difference is that none of the tested configurations had a high probability of being different within the MC stage; the p-values are shown in Table 4.5. This indicates that none of the configurations tested provided significantly different values across the MC stage at a byte level within a 99% confidence interval. This is contrasted with the bit level analyses, where a significant difference was observed when varying the MC architecture.



**Table 4.5: Power HD MC  $t$ -Test p-Values**

<b>MC</b>	<b>Power HD</b>			
	<b>LL</b>	<b>LG</b>	<b>GL</b>	<b>GG</b>
<b>LL</b>	x	0.184365	0.331222	0.231752
<b>LG</b>	0.184365	x	0.380402	0.581512
<b>GL</b>	0.331222	0.380402	x	0.295397
<b>GG</b>	0.231752	0.581512	0.295397	x

All analysis methods and measurement types exhibited at least weak correlation in each of the primitives, regardless of architecture. The average case for power measurements analyzed with the HD leakage model showed the highest overall correlations in each stage, but the composite field arithmetic SB primitive configurations resulted in reductions in the total leakage. This is also true for the maximum observed correlations in each stage. The EM HW maximum correlation is still extremely high with values above .008, which indicates that at least some of the bytes are exploitable. Composite field arithmetic SB configurations exhibit correlations above .0053 (above 99.99% confidence threshold) in the power measurements under the HD model, meaning that at least some of the bytes are exploitable under this model. Rounds 1 and 10 in particular are within only 99% confidence, meaning that moderate correlation exists for the rounds that are most likely to be exploited and an attacker is likely to use false-positives when attacking these rounds.

Again using the same assumptions listed in the bit level summary, the number of exploitable bytes for given confidence levels in the GAL-LUT configuration are listed in Table 4.6. All 16 bytes are exploitable across all confidence levels under the power HD analysis. This means that an attacker can compromise all bytes of the secret key, even

with the best tested configuration. 15 of the bytes can also be compromised for EM measurements, meaning that this implementation is not able to adequately protect the secret key under either measurement type. This indicates that simply varying the architecture of primitives is not sufficient to protect an implementation from SCA.

**Table 4.6: Number of Exploitable Bytes per Confidence Level**

	Confidence Level			
Model	0.9	0.99	0.999	0.9999
EMHW	16	16	16	15
EMHD	16	16	16	13
POWHW	16	13	1	0
POWHD	16	16	16	16

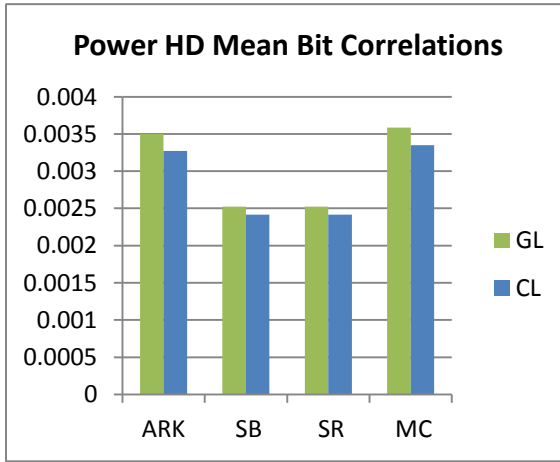
## 4.2 Effects of High-Level Architecture Variation

This section compares the GAL-LUT standard configuration with a more complex GAL-LUT combined decryption/encryption implementation. Bit- and byte-level analyses are performed to determine if there are significant differences between the two implementations. All leakage models and measurement types are used in these analyses.

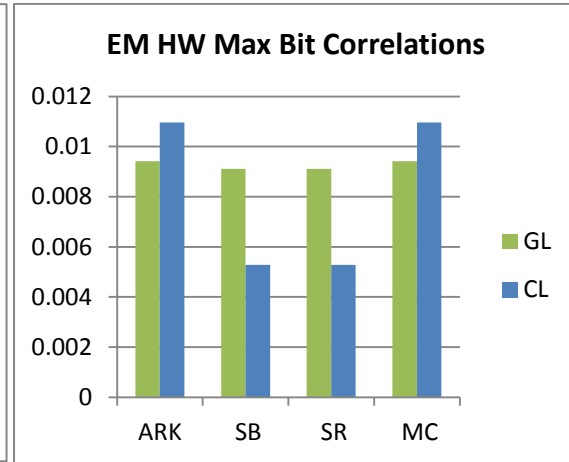
Table 4.7 lists the power HD analysis minimum, maximum, and mean maximum correlation coefficients across each configuration under the bit analysis. The other tables are listed in Appendix A. Figure 4.16 is the average case. Under this type of analysis, the combined configuration provides only slightly better correlation values across each primitive. All other types of analyses appear to only provide negligible differences in each stage. The best case for the EM HW model is greatly reduced in the SB stage and SR stages with the combined type, but increases the maximum leakage in both the ARK and MC stages (Figure 4.17).

**Table 4.7: Maximum Correlation Statistics by Architecture**

Bits	Power	HD	Architecture			
		C-LUT			GAL-LUT	
Stage	Min	Max	Mean	Min	Max	Mean
ARK	0.001445	0.005842	0.003273	0.001552	0.006953	0.003501
SB	0.00118	0.004302	0.002415	0.001321	0.004812	0.002525
SR	0.00118	0.004302	0.002415	0.001321	0.004812	0.002525
MC	0.001879	0.005606	0.00335	0.001934	0.006954	0.003584

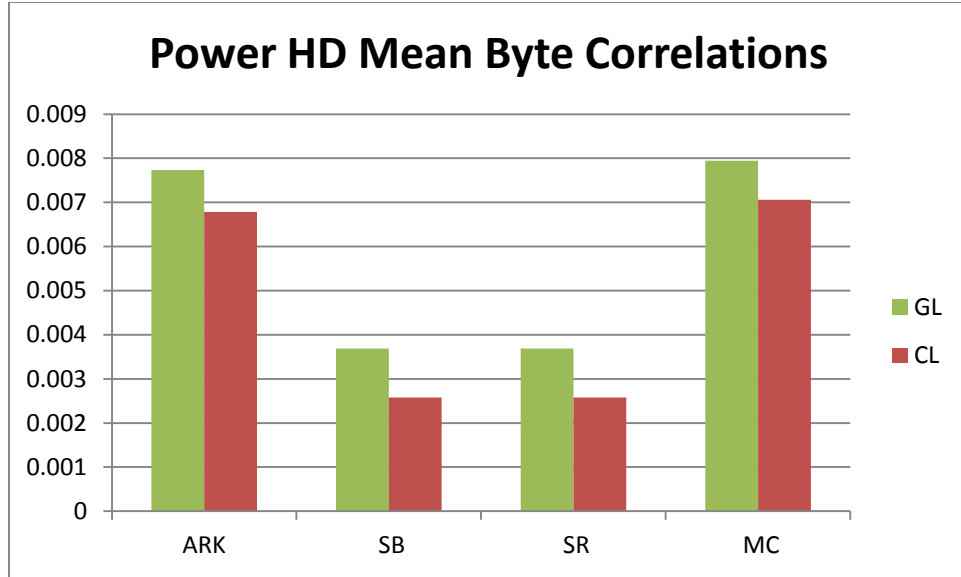


**Figure 4.16: Mean Bit Correlations**



**Figure 4.17: Maximum Bit Correlations**

Similar trends appear in the byte-level analysis. Figure 4.18 shows the byte-level power HD analysis for the average case. The combined style reduces correlations by 10-25% depending on the stage. Other models and measurement types do not show significant differences. This means that the additional hardware components likely add noise only to the power measurements; EM is largely unaffected due to its localized nature.



**Figure 4.18: Mean Byte Correlations**

Table 4.8 details the differences between the two high-level architectures for each of the primitives between the EM HW and power HD models at both the bit and byte levels. At both levels under the power HD model there is a high likelihood that the means for all three primitives are different between the GAL-LUT and combined architectures. The EM HW analysis does not follow the same trend, however, and none of the stages have statistically significant differences within 99% confidence.

**Table 4.8: High-Level Average Correlation Differences**

Bit	EM HW	Power HD	Byte	EM HW	Power HD
ARK	-1.110E-04*	4.354E-04	ARK	-5.235E-04*	1.167E-03
SB	1.970E-04*	1.201E-04*	SB	9.150E-04*	1.205E-03
MC	-2.271E-04*	4.678E-04	MC	-6.192E-04*	1.214E-03

\*- Not statistically significant within 99% confidence interval

Using a combined encryption/decryption architecture has been shown to be beneficial for limiting leakages under the power HD model, but has little effect on other analysis and measurement types. Table 4.9 shows the number of bits and bytes that can

be compromised at a given confidence level, again using the assumption that a high correlation coefficient can alone be used to exploit individual bits and bytes. The impact of the combined implementation can be seen in the bits for the power HD analysis, reducing the number of exploitable bits from 33 to 7 under the highest confidence level. Unfortunately it had no effect on the EM HW model, so 32 bits can still be compromised. The combined style did not affect the byte-level correlations enough to reduce the number of exploitable bytes, so the implementation is still quite weak to SCA at the byte level.

**Table 4.9: Combined EN/DE Exploitable Bits and Bytes**

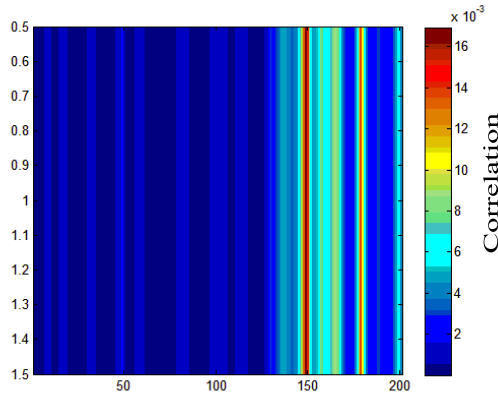
Bit	Confidence Level				Byte	Confidence Level			
Model	0.9	0.99	0.999	0.9999	Model	0.9	0.99	0.999	0.9999
EM HW	128	127	77	32	EM HW	16	16	16	15
EM HD	128	116	9	0	EM HD	16	16	16	15
Power HW	128	99	1	0	Power HW	16	16	0	0
Power HD	128	128	70	7	Power HD	16	16	16	16

### 4.3 Conclusions & Reasoning

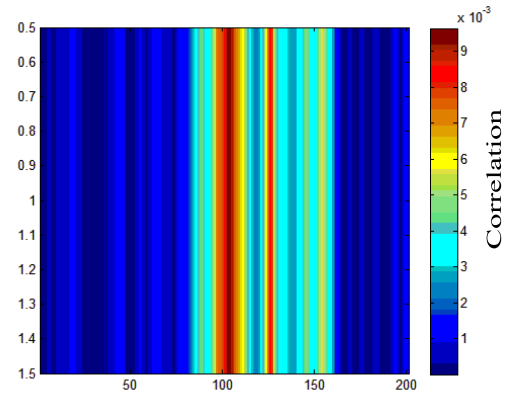
Galois field calculation showed reduced leakage in many of the analyses. This is opposite of the initial hypothesis that the LUT variants would leak less information due to less data-dependent operations. This may occur because each of the LUT variants execute in constant time regardless of the input data, meaning that the target register is overwritten during the same time sample each time. This would lead to higher correlations at that one sample, or any other immediately surrounding samples. The composite field arithmetic process uses an asynchronous calculation method that may overwrite the target register in varying time samples. This means that the information

leakage is spread across a range of samples, rather than highly focused in a single or much smaller range of samples. This is supported by the general research into asynchronous or random clocking [FMP03].

This can be investigated with the current leakage maps by examining the samples immediately surrounding the highest points of information leakage. In a LUT variant (Figure 4.19), a single point of high correlation is visible with limited additional correlation within 20 samples on either side of the main spike. In an equivalent GAL variant (Figure 4.20), the relatively high correlations can be seen within 30 samples of the central spike. In addition, the spike's highest correlations are spread across 10 samples, whereas the spike spans only 3 samples in the LUT variant. A direct comparison between the two is shown in Figure 4.21. This loosely supports the theory of asynchronous clocking distributing the register writing times, however other experiments should be conducted to investigate this further.



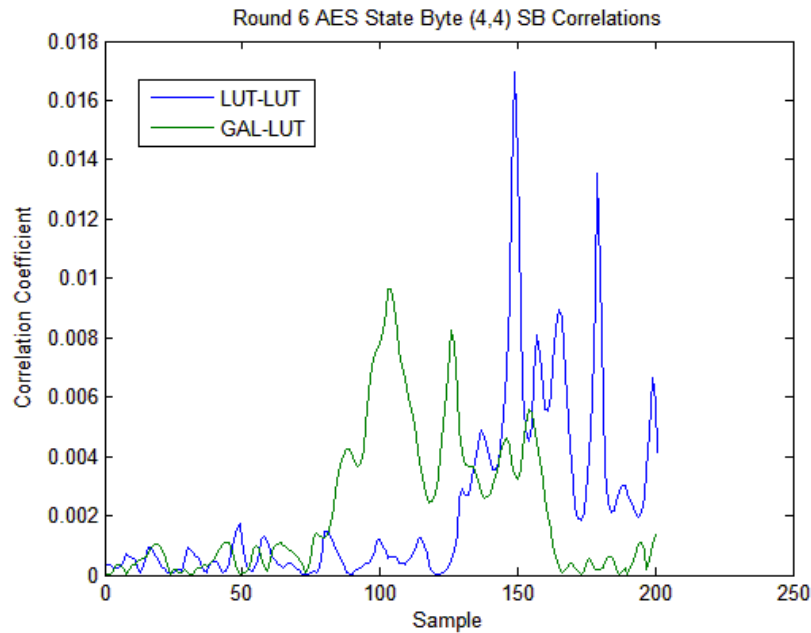
**Figure 4.19: LUT-LUT Configuration  
Round 6 SB**



**Figure 4.20: GAL-LUT Configuration  
Round 6 SB**

Varying the architecture of the MC primitive showed mixed results in many of the different analysis types. Using Galois multiplication resulted in higher correlations for most stages in the average case, however it lowered the maximum correlation in the best

case. This variation was less significant than the variation seen from changing the SB primitive, often resulting in a difference of less than 10% between the LUT alternative. This could be due to the fact that a reduced form of the MC operation was used which required only a single multiplication. It is possible that other functionally equivalent but more complex MC primitives would show more consistent effects on the information leakage.



**Figure 4.21: Round 6 SB Correlation Comparison**

Varying the high-level architecture had a significant effect on the observed correlations under the power HD analysis. The combined encryption/decryption implementation was expected to add additional noise to the system by processing additional information simultaneously along the decryption datapath, however this additional noise had a negligible impact on the magnitude of correlations for most the measurement types and leakage models. The more complex architecture was successful

in limiting the number of exploited bits that showed high correlations in the power HD model, but did not affect the EM HW model. This means that the implementation as a whole is still weak, but provides some evidence that varying the high-level architecture can affect information leakage. Additional types of high-level architectures must be tested to gain a better understanding of their impact. Both types of high-level architectures tested used a fully-unrolled pipelined approach; an iterative or partially-unrolled pipeline may have a different impact on the results.

#### **4.4 Summary**

Across both the bit and byte levels, composite field arithmetic calculation used in the SB primitive resulted in reduced leakage in each analysis method. Varying the MC architecture had a limited effect on the information leakage, but other architecture types can potentially lead to more effective leakage reduction. Each architecture shows at least weak correlations, however the average cases were improved in GAL SB configurations. Some bits and bytes are still highly exploitable based off of the level of correlation at different stages, however the most direct points of attack do not have exceptionally high leakage. Varying the high-level architecture showed limited impact on only one of the analysis types, and did not have a significant effect on the correlations of other analyses.



## V. Conclusions and Future Work

The analysis in Chapter 4 shows that different primitive architectures do in fact affect the information leakage of a cryptosystem. In particular, composite field arithmetic calculation of the SB stage is effective in reducing the overall leakage across each measurement type and leakage model. Varying the MC primitive between LUT and Galois multiplication variants also affected the information leakage. However, assessment of whether or not this variation reduced or added to leakages was inconclusive. The best practice lesson from this analysis is that LUT-based SB stages will probably leak more information than a direct calculation alternative. Other MC types must be tested and explored to find a style that is more effective at reducing leakage across each of the analysis types.

The exact reasoning for the differences in information leakage is unclear, but leakage map inspection supports the theory that wider temporal distribution of the register write times is related to the lower leakage levels. Due to inconsistent circuit timings and internal asynchronous signaling, the final result for a given stage is written at varying times within a normal distribution. These experiments were not designed to test this hypothesis however, and other experiments must be conducted to explore this problem in greater depth.

Across every type of analysis and level, the ARK and MC stages exhibited the highest correlations of the primitives. The maximum correlation coefficients for the SB and SR stages were often a third less than the observed maximum correlations for ARK or MC. The experiments provided evidence that changing primitive structures can affect the amount of information leaked by that primitive, so it is possible that alternate

implementations of ARK and untested MC styles can lead to lower information leakage in those primitives.

All tested configurations exhibited at least weak correlation, meaning that the device was potentially exploitable through correlation attacks. Significant variation in correlation coefficients was observed by changing some of the primitives, which suggests that other primitive architectures can also affect the observed leakage. The experimental methodology presented in this document can be extended to other types of primitives, such as a T-Box architecture or other unique SB and MC styles.

Attacks on the key schedule were not explored in this research, but these are a potential threat to many types of systems. As with the primitives, a number of different processes exist to calculate the individual round keys. These other options need to be explored to form a broader scope of knowledge with regard to best practice in constructing secure AES implementations.

Beyond the primitive architectures, varying the high-level system structure also showed some effect on information leakage. While not consistent across all leakage models and measurement types, the additional noise from the complex combined encryption/decryption architecture reduced the correlation coefficients of the power HD model. This is not sufficient to claim an increase in general SCA resistance, but it does mean that it is possible to fine-tune the development of some cryptosystems to resist specific types of SCA. Additional high-level architectures, such as iterative and partially-unrolled pipelines, must be tested to determine how they affect a wide number of leakage models and measurement types.

Other leakage models may more accurately capture intermediate operations than those used in this research. The effectiveness of these and other configurations at limiting information leakage is entirely dependent on the leakage model's ability to capture leakage. The two leakage models tested in this research showed different results for all of the architectures and measurement types; some were far more effective than others at correlating the observed information to the hypotheses. Other leakage models will have to be researched and applied to this methodology to be able to truly assert any level of resistance.

Among the configurations tested, none were able to assert complete resistance to SCA attacks, and some bits and bytes were always susceptible to attack. However, the results show that it is possible for other untested configurations to achieve full resistance at this level of testing. This means that proper secure hardware design practices could potentially act as a countermeasure by itself. This would reduce the necessary resources and development times to create secure cryptosystems with enough breadth and understanding. Further research with this methodology could be very beneficial in this manner, but it also has the possibility of failing to reduce correlations low enough to act as a countermeasure.

## Appendix A: Bit Maximum Leakage Tables

Bits	EM	HD	Arch									
		LUT-LUT			LUT-GAL			GAL-LUT			GAL-GAL	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001553	0.00617	0.002822	0.001506	0.005051	0.002737	0.001458	0.004843	0.002658	0.001384	0.004999	0.002681
SB	0.001432	0.004605	0.002738	0.001699	0.004692	0.002733	0.001428	0.004609	0.002525	0.001392	0.004469	0.002571
SR	0.001432	0.004605	0.002738	0.001699	0.004692	0.002733	0.001428	0.004609	0.002525	0.001392	0.004469	0.002571
MC	0.001555	0.00617	0.002819	0.001506	0.005051	0.002718	0.00146	0.004843	0.002646	0.001384	0.004999	0.002673

Bits	POW	HD	Arch									
		LUT-LUT			LUT-GAL			GAL-LUT			GAL-GAL	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001449	0.008862	0.003861	0.001248	0.007425	0.003504	0.001552	0.006953	0.003501	0.001614	0.007496	0.003356
SB	0.001484	0.008199	0.003195	0.001493	0.008347	0.003329	0.001321	0.004812	0.002525	0.00132	0.004646	0.002594
SR	0.001484	0.008199	0.003195	0.001493	0.008347	0.003329	0.001321	0.004812	0.002525	0.00132	0.004646	0.002594
MC	0.001449	0.008855	0.00372	0.001248	0.007425	0.003294	0.001934	0.006954	0.003584	0.001618	0.007493	0.003441

Bits	EM	HW	Arch									
		LUT-LUT			LUT-GAL			GAL-LUT			GAL-GAL	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001654	0.017201	0.003201	0.001465	0.016425	0.003407	0.001269	0.009419	0.002793	0.00133	0.008508	0.002734
SB	0.001551	0.017201	0.002934	0.00158	0.01324	0.002824	0.00144	0.009113	0.00263	0.001403	0.009259	0.002729
SR	0.001551	0.017201	0.002934	0.00158	0.01324	0.002824	0.00144	0.009113	0.00263	0.001403	0.009259	0.002729
MC	0.001654	0.015321	0.002987	0.001465	0.015342	0.003085	0.001269	0.009419	0.002772	0.00133	0.008508	0.002747

Bits	POW	HW	Arch									
		LUT-LUT			LUT-GAL			GAL-LUT			GAL-GAL	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001074	0.004358	0.002349	0.000993	0.004467	0.002272	0.001186	0.004365	0.002373	0.001158	0.004422	0.002351
SB	0.001074	0.00441	0.002341	0.001079	0.004509	0.00224	0.000993	0.004189	0.002345	0.001161	0.004053	0.00232
SR	0.001074	0.00441	0.002341	0.001079	0.004509	0.00224	0.000993	0.004189	0.002345	0.001161	0.004053	0.00232
MC	0.001265	0.004358	0.002354	0.000993	0.004393	0.002268	0.001186	0.004365	0.002387	0.001158	0.004422	0.00237

Bits	EM	HD	Arch			
		C-LUT			GAL-LUT	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001554	0.005122	0.002682	0.001458	0.004843	0.002658
SB	0.001416	0.004554	0.002485	0.001428	0.004609	0.002525
SR	0.001416	0.004554	0.002485	0.001428	0.004609	0.002525
MC	0.00155	0.005122	0.002679	0.00146	0.004843	0.002646

Bits	POW	HD	Arch			
		C-LUT			GAL-LUT	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001445	0.005842	0.003273	0.001552	0.006953	0.003501
SB	0.00118	0.004302	0.002415	0.001321	0.004812	0.002525
SR	0.00118	0.004302	0.002415	0.001321	0.004812	0.002525
MC	0.001879	0.005606	0.00335	0.001934	0.006954	0.003584

Bits	EM	HW	Arch			
		C-LUT			GAL-LUT	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001442	0.010957	0.002783	0.001269	0.009419	0.002793
SB	0.0013	0.005282	0.002561	0.00144	0.009113	0.00263
SR	0.0013	0.005282	0.002561	0.00144	0.009113	0.00263
MC	0.001503	0.010957	0.002852	0.001269	0.009419	0.002772

Bits	POW	HW	Arch			
		C-LUT			GAL-LUT	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001143	0.004488	0.002403	0.001186	0.004365	0.002373
SB	0.00118	0.004182	0.002385	0.000993	0.004189	0.002345
SR	0.00118	0.004182	0.002385	0.000993	0.004189	0.002345
MC	0.001143	0.004488	0.002418	0.001186	0.004365	0.002387

## Appendix B: Byte Maximum Leakage Tables

Bytes	EM	HD	Arch									
		LL			LG			GL			GG	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001979	0.00847	0.003819	0.001961	0.007278	0.003626	0.001763	0.007829	0.004063	0.002095	0.007032	0.004095
SB	0.001804	0.005485	0.003308	0.002071	0.006244	0.003524	0.001505	0.004559	0.002694	0.001586	0.005586	0.002939
SR	0.001804	0.005485	0.003308	0.002071	0.006244	0.003524	0.001505	0.004559	0.002694	0.001586	0.005586	0.002939
MC	0.001979	0.00847	0.003723	0.001961	0.006747	0.003563	0.00244	0.006134	0.004005	0.002303	0.005956	0.004104

Bytes	POW	HD	Arch									
		LL			LG			GL			GG	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.002167	0.020894	0.008945	0.002312	0.016098	0.007664	0.003834	0.012391	0.007732	0.003332	0.011065	0.007228
SB	0.003683	0.015104	0.006722	0.004138	0.012012	0.007474	0.001957	0.006375	0.003689	0.00246	0.006507	0.004132
SR	0.003683	0.015104	0.006722	0.004138	0.012012	0.007474	0.001957	0.006375	0.003689	0.00246	0.006507	0.004132
MC	0.002167	0.020902	0.008434	0.002301	0.015241	0.006884	0.005698	0.012391	0.007946	0.005007	0.011065	0.007492



Bytes	EM	HW	Arch									
		LL			LG			GL			GG	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.001987	0.018897	0.003753	0.001715	0.016486	0.004056	0.001546	0.013521	0.003143	0.001385	0.008796	0.002999
SB	0.001762	0.01053	0.003168	0.00162	0.010996	0.002922	0.001503	0.009685	0.002971	0.001637	0.009438	0.003254
SR	0.001762	0.01053	0.003168	0.00162	0.010996	0.002922	0.001503	0.009685	0.002971	0.001637	0.009438	0.003254
MC	0.001564	0.01111	0.00322	0.00175	0.012533	0.00358	0.001625	0.010211	0.002835	0.001627	0.006362	0.002735

Bytes	POW	HW	Arch									
		LL			LG			GL			GG	
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
ARK	0.00141	0.00394	0.002326	0.001422	0.00395	0.00227	0.001219	0.004597	0.002322	0.001374	0.004182	0.002293
SB	0.001522	0.003478	0.002354	0.001332	0.003642	0.002279	0.001266	0.003927	0.002397	0.001353	0.003889	0.002378
SR	0.001522	0.003478	0.002354	0.001332	0.003642	0.002279	0.001266	0.003927	0.002397	0.001353	0.003889	0.002378
MC	0.001249	0.003789	0.002349	0.001182	0.003851	0.002298	0.001431	0.003802	0.002403	0.001296	0.003421	0.002358

Bytes	EM	HD	Arch			
		CL			GL	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001746	0.007691	0.004329	0.001763	0.007829	0.004063
SB	0.001375	0.004103	0.002498	0.001505	0.004559	0.002694
SR	0.001375	0.004103	0.002498	0.001505	0.004559	0.002694
MC	0.002247	0.007295	0.004277	0.00244	0.006134	0.004005

Bytes	POW	HD	Arch			
		CL			GL	
	Min	Max	Mean	Min	Max	Mean
ARK	0.002943	0.009989	0.006785	0.003834	0.012391	0.007732
SB	0.001225	0.00475	0.00258	0.001957	0.006375	0.003689
SR	0.001225	0.00475	0.00258	0.001957	0.006375	0.003689
MC	0.004678	0.009992	0.007064	0.005698	0.012391	0.007946

Bytes	EM	HW	Arch			
		CL			GL	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001571	0.013847	0.003392	0.001546	0.013521	0.003143
SB	0.001344	0.007885	0.002804	0.001503	0.009685	0.002971
SR	0.001344	0.007885	0.002804	0.001503	0.009685	0.002971
MC	0.001611	0.010275	0.003018	0.001625	0.010211	0.002835

Bytes	POW	HW	Arch			
		CL			GL	
	Min	Max	Mean	Min	Max	Mean
ARK	0.001394	0.004369	0.002372	0.001219	0.004597	0.002322
SB	0.001403	0.004184	0.00247	0.001266	0.003927	0.002397
SR	0.001403	0.004184	0.00247	0.001266	0.003927	0.002397
MC	0.001454	0.003413	0.002385	0.001431	0.003802	0.002403

### Appendix C: Byte First and Last Round Tables

Byte	EM	HD		Arch				
	LL		LG		GL		GG	
	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10
ARK	0.004741	0.003759	0.005033	0.002787	0.005753	0.002887	0.005254	0.002864
SB	0.002898	0.00376	0.003156	0.002787	0.002791	0.002887	0.003061	0.002864
SR	0.002898	0.00376	0.003156	0.002787	0.002791	0.002887	0.003061	0.002864
MC	0.00319		0.003324		0.003839		0.004164	

Byte	POW	HD		Arch				
	LL		LG		GL		GG	
	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10
ARK	0.011534	0.010959	0.012014	0.010337	0.008605	0.004929	0.007608	0.004473
SB	0.005886	0.010961	0.006266	0.010338	0.003628	0.00493	0.003836	0.004473
SR	0.005886	0.010961	0.006266	0.010338	0.003628	0.00493	0.003836	0.004473
MC	0.007251		0.006796		0.007563		0.007601	

Byte	EM	HW		Arch				
	LL		LG		GL		GG	
	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10
ARK	0.004744	0.00576	0.00553	0.007427	0.003133	0.002688	0.002542	0.002515
SB	0.002649	0.004437	0.002926	0.003687	0.00266	0.003276	0.002859	0.002654
SR	0.002649	0.004437	0.002926	0.003687	0.00266	0.003276	0.002859	0.002654
MC	0.002763		0.004055		0.002644		0.002499	

Byte	POW	HW		Arch				
	LL		LG		GL		GG	
	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10	Round 1	Round 10
ARK	0.002365	0.002171	0.002164	0.002123	0.002097	0.002294	0.002117	0.002172
SB	0.002388	0.002343	0.002428	0.002221	0.002406	0.00232	0.002403	0.002253
SR	0.002388	0.002343	0.002428	0.002221	0.002406	0.00232	0.002403	0.002253
MC	0.002128		0.002144		0.00224		0.002092	

### Appendix D: *t*-Test p-Value Tables

Bits	ARK	EMHW		
	LL	LG	GL	GG
LL	x	0.016889039	2.46E-08	6.90723E-12
LG	0.016889039	x	1.54E-17	3.43142E-22
GL	2.46044E-08	1.53832E-17	x	0.005852759
GG	6.90723E-12	3.43142E-22	0.005853	x

Bits	SB	EMHW		
	LL	LG	GL	GG
LL	x	0.167275232	1.88E-06	0.000480166
LG	0.167275232	x	1.83E-05	0.011074805
GL	1.88089E-06	1.83206E-05	x	0.007642919
GG	0.000480166	0.011074805	0.007643	x

Bits	MC	EMHW		
	LL	LG	GL	GG
LL	x	0.551662128	4.47E-05	1.01142E-08
LG	0.551662128	x	2.78E-06	3.49069E-10
GL	4.47494E-05	2.78443E-06	x	0.01382771
GG	1.01142E-08	3.49069E-10	0.013828	x

Bits	ARK	POWHD		
	LL	LG	GL	GG
LL	x	0.017791	0.000290413	1.02477E-06
LG	0.017791	x	0.143911576	0.000918902
GL	0.00029	0.143912	x	0.013386391
GG	1.02E-06	0.000919	0.013386391	x

Bits	SB	POWHD		
	LL	LG	GL	GG
LL	x	0.683709	4.74946E-36	1.40721E-32
LG	0.683709	x	1.55371E-30	2.25172E-27
GL	4.75E-36	1.55E-30	x	0.048924795
GG	1.41E-32	2.25E-27	0.048924795	x

Bits	MC	POWHD		
	LL	LG	GL	GG
LL	x	0.001494	0.443079823	0.04527418
LG	0.001494	x	9.87972E-05	0.022443616
GL	0.44308	9.88E-05	x	0.009414969
GG	0.045274	0.022444	0.009414969	x

Bytes	ARK	EMHW		
	LL	LG	GL	GG
LL	x	0.351728935	0.039057	0.004812
LG	0.351729	x	0.000701	2.72E-05
GL	0.039057	0.000701009	x	0.220598
GG	0.004812	2.72333E-05	0.220598	x
Bytes	SB	EMHW		
	LL	LG	GL	GG
LL	x	0.55304918	0.68645	0.889858
LG	0.553049	x	0.79032	0.425819
GL	0.68645	0.790319885	x	0.52731
GG	0.889858	0.425818619	0.52731	x
Bytes	MC	EMHW		
	LL	LG	GL	GG
LL	x	0.233574823	0.113307	0.029084
LG	0.233575	x	0.00514	0.000613
GL	0.113307	0.005140277	x	0.611725
GG	0.029084	0.000612503	0.611725	x

Bytes	ARK	POWHD		
	LL	LG	GL	GG
LL	x	0.309985	0.004081	0.001705
LG	0.309985	x	0.00123	0.000215
GL	0.004081	0.00123	x	0.223594
GG	0.001705	0.000215	0.223594	x
Bytes	SB	POWHD		
	LL	LG	GL	GG
LL	x	0.456595	1.87E-10	4.39E-10
LG	0.456595	x	1.86E-16	7.59E-16
GL	1.87E-10	1.86E-16	x	0.190812
GG	4.39E-10	7.59E-16	0.190812	x
Bytes	MC	POWHD		
	LL	LG	GL	GG
LL	x	0.184365	0.331222	0.231752
LG	0.184365	x	0.380402	0.581512
GL	0.331222	0.380402	x	0.295397
GG	0.231752	0.581512	0.295397	x

<b>Combined-LUT vs GAL-LUT</b>						
<b>Bit</b>	<b>EM</b>	<b>POW</b>		<b>Byte</b>	<b>EM</b>	<b>POW</b>
<b>ARK</b>	0.4874125	1.29E-08		<b>ARK</b>	0.560026	0.001042
<b>SB</b>	0.0238987	0.023903		<b>SB</b>	0.108607	4.11E-07
<b>MC</b>	0.1583897	1.37E-09		<b>MC</b>	0.412668	0.001032



## Bibliography

- [ARR03] Agrawl, D; Rao, J; Rohatgi, P. "Multi-channel Attacks." Cryptographic Hardware and Embedded Systems – CHES 2003. 8-10 September, 2003.
- [CJR99] Chari, S; Jutla, C; Rao, J; Rohatgi, P. "Towards Sound Approaches to Counteract Power-Analysis Attacks." CRYPTO, volume 1666 of Lecture Notes in Computer Science, pp 398-412, Springer. 1999.
- [CRR02] Chari, S; Rao, J; Rohatgi, P. "Template Attacks." Cryptographic Hardware and Embedded Systems – CHES 2002. 13-15 August, 2002.
- [Cob11] Cobb, W. "Exploitation of the Unintentional Information Leakage of Integrated Circuits." Dissertation, Air Force Institute of Technology. September, 2011.
- [Fip01] Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES)." National Institute of Standards and Technology. 26 November, 2001.
- [FMP03] Fouque, P-A; Martinet, G; Poupard, G. "Attacking Unbalanced RSA-CRT Using SPA." Cryptographic Hardware and Embedded Systems – CHES 2003. 8-10 September, 2003.
- [GMO11] Gandolfi, K; Mourtel, C; Olivier, F. "Electromagnetic Analysis: Concrete Results." Cryptographic Hardware and Embedded Systems - CHES 2011. 2011.
- [KaA10] Kakarlapudi, B; Alabur, N. "FPGA Implementations of S-box vs T-box Iterative Architectures of AES." George Mason University, Secure Telecommunications Systems term project.
- [KJJ98] Kocher, P; Jaffe, J; Jun, B. "Introduction to Differential Power Analysis and Related Attacks." White paper, Cryptography Research Inc. 1998.
- [KJJ99] Kocher, P; Jaffe, J; Jun, B. "Differential Power Analysis." CRYPTO Conference. 1999.
- [KJJ11] Kocher, P; Jaffe, J; Jun, B; Rohatgi, P. "Introduction to Differential Power Analysis." Journal of Cryptographic Engineering vol 1 issue 1, pp 5-27. 3 March, 2011.
- [LiF05] Li, H; Friggstad, Z. "An Efficient Architecture for the AES Mix Columns Operation." IEEE International Symposium on Circuits and Systems 2005. 23-26 May, 2005.

- [MPG05] Mangard, S; Popp, T; Gammel, B. “Side-Channel Leakage of Masked CMOS Gates.” RSA Conference 2005. 14-18 February, 2005.
- [MaS06] Mangard, S; Schramm, K. “Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations.” Cryptographic Hardware and Embedded Systems – CHES 2006. 10-13 October, 2006.
- [MOP07] Mangard, S; Oswald, E; Popp, T. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer Science+Business Media LLC. 2007.
- [MiR04] Micali, S; Reyzin, L. “Physically Observable Cryptography.” TCC 2004, Lecture Notes in Computer Science, vol 2951, pp 278-296. February, 2004.
- [MoS02] Morioka, S; Satoh, A. “A 10-Gbps Full-AES Crypto Design with a Twisted BDD S-Box Architecture.” IBM Research, Tokyo Research Laboratory. 24 June, 2002.
- [Nsa03] “National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information.” CNSS Policy No. 15, Fact Sheet No. 1. National Security Agency. June, 2003.
- [OsM07] Oswald, E; Mangard, S. “Template Attacks on Masking – Resistance is Futile.” RSA Conference 2007. 5-9 February, 2007.
- [PRB09] Prouff, E; Rivain, M; Bevan, R. “Statistical Analysis of Second Order Differential Power Analysis.” IEEE Transactions on Computers, pp 799-811. June, 2009.
- [ReO04] Rechberger, C; Oswald, E. “Practical Template Attacks.” 5<sup>th</sup> International Workshop on Information Security Applications – WISA 2004. 23-25 August, 2004.
- [RSV09] Renaud, M; Standaert, F-X; Veyrat-Charvillon, N. “Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA.” Cryptographic Hardware and Embedded Systems – CHES 2009. 6-9 September, 2009.
- [Ris10] “Session EMA-1: Physics of EM Emanations.” SCARE Crash Course, Riscure Inc. 2010.
- [RSD06] Rodriguez-Henriquez, F; Saqib, N; Diaz-Perez, A; Koc, C. Cryptographic Algorithms on Reconfigurable Hardware. Springer Series on Signals and Communication Technology. Springer Science+Business Media, LLC. 2006.
- [SaS75] Saltzer, J; Schroeder, M. “The Protection of Information in Computer Systems.” Proceedings of the IEEE, vol 63 Issue 9, pp 1278-1308. September, 1975.

- [ScT07] Schaumont, P; Tiri, K. “Masking and Dual-Rail Logic Don’t Add Up.” Cryptographic Hardware and Embedded Systems – CHES 2007. 10-13 September, 2007.
- [Sch00] Schneier, B. “A Self-Study Course in Block-Cipher Cryptanalysis.” Cryptologia, Volume 24, Issue 1, pp 18-33. 2000.
- [StL07] Stamp, M; Low, R. Applied Cryptanalysis. John Wiley & Sons Inc., Hoboken, New Jersey, 2007.
- [SRQ03] Standaert, F-X; Rouvroy, G; Quisqatar, J-J; Legat, J-D. “Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs.” Cryptographic Hardware and Embedded Systems - CHES 2003. 8-10 September, 2003.
- [SOP04] Standaert, F-X; Ors, S; Preneel, B. “Power Analysis of an FPGA Implementation of Rijndael: Is Pipelining a DPA Countermeasure?” Cryptographic Hardware and Embedded Systems – CHES 2004. 11-13 August, 2004.
- [SMY06] Standaert, F-X; Malkin, T; Yung, M. “A Formal Practice-Oriented Model for Analysis of Side-Channel Attacks.” Columbia University. 2006.
- [SuS06] Suzuki, D; Saeki, M. “Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style.” Cryptographic Hardware and Embedded Systems – CHES 2006. 10-13 October, 2006.
- [TAV02] Tiri, K; Akmal, M; Verbauwhede, I. “A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards.” Solid-State Circuits Conference – ESSCIRC 2002. September 2002.
- [TiV04] Tiri, K; Verbauwhede, I. “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation.” Design, Automation and Test in Europe Conference and Exhibition – DATE 2004. 2004.
- [Wik12] “Advanced Encryption Standard.” Wikipedia. Accessed June 16, 2012. url: [http://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 13-09-2012		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) Sept 2010 – Sept 2012	
TITLE AND SUBTITLE  Effects of Architecture on Information Leakage of a Hardware Advanced Encryption Standard Implementation				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Koziel, Eric A., Civilian				5d. PROJECT NUMBER 12G403	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENG) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GCO/ENG/12-25	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate 2241 Avionics Circle, Bldg 620, Wright-Patterson AFB OH 45433 James Alverson, (937) 986-5774				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RYWA	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT Side-channel analysis (SCA) is a threat to many modern cryptosystems. Many countermeasures exist, but are costly to implement and still do not provide complete protection against SCA. A plausible alternative is to design the cryptosystem using architectures that are known to leak little information about the cryptosystem's operations. This research uses several common primitive architectures for the Advanced Encryption Standard (AES) and assesses the susceptibility of the full AES system to side-channel attack for various primitive configurations. A combined encryption/decryption core is also evaluated to determine if variation of high-level architectures affects leakage characteristics. These different configurations are evaluated under multiple measurement types and leakage models. The results show that different hardware configurations do impact the amount of information leaked by a device, but none of the tested configurations are able to prevent exploitation.					
15. SUBJECT TERMS Side Channel Analysis, Hardware Security, Anti-tamper Design					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES  79	19a. NAME OF RESPONSIBLE PERSON Dr. Rusty Baldwin ADVISOR
a. REPORT  U	b. ABSTRACT  U	c. THIS PAGE  U			19b. TELEPHONE NUMBER (Include area code) (937) 255-6565, (rusty.baldwin@afit.edu)

Standard Form 298 (Rev. 8-98)  
Prescribed by ANSI Std. Z39-18