12-2018

A Quantitative Study of Advanced Encryption Standard Performance as it Relates to Cryptographic Attack Feasibility

Daniel Hawthorne
United States Military Academy, daniel.hawthorne@westpoint.edu

A QUANTITATIVE STUDY OF ADVANCED ENCRYPTION STANDARD PERFORMANCE AS IT RELATES TO CRYPTOGRAPHIC ATTACK FEASIBILITY

A Dissertation Presented in Partial Fulfillment of the Requirements for the Degree of Doctor of Computer Science

By

Daniel Stephen Hawthorne

Colorado Technical University

December, 2018

Committee

Dr. Richard Livingood, Ph.D., Chair

Dr. Kelly Hughes, DCS, Committee Member

Dr. James O. Webb, Ph.D., Committee Member

December 17, 2018

Abstract

The advanced encryption standard (AES) is the premier symmetric key cryptosystem in use today. Given its prevalence, the security provided by AES is of utmost importance. Technology is advancing at an incredible rate, in both capability and popularity, much faster than its rate of advancement in the late 1990s when AES was selected as the replacement standard for DES. Although the literature surrounding AES is robust, most studies fall into either theoretical or practical yet infeasible. This research takes the unique approach drawn from the performance field and dual nature of AES performance. It uses benchmarks to assess the performance potential of computer systems for both general purpose and AES. Since general performance information is readily available, the ratio may be used as a predictor for AES performance and consequently attack potential. The design involved distributing USB drives to facilitators containing a bootable Linux operating system and the benchmark instruments. Upon boot, these devices conducted the benchmarks, gathered system specifications, and submitted them to a server for regression analysis. Although it is likely to be many years in the future, the results of this study may help better predict when attacks against AES key lengths will become feasible.

Acknowledgements

I need to begin by thanking my research supervisor and dissertation chair Dr. Rick. His guidance and patience made this dissertation possible. I also need to thank Colonel Chewar for telling me that I am crazy for trying to do this on active duty, but encouraging me throughout the process and even volunteering to be an external committee member.

To my friends and family: thank you for your patience and understanding as I completed this journey, especially my wife Tammy who stood by me through the process, and my father, Jon for introducing me to technology, computers, and programming.

This research would not have been possible without the support of research facilitators who volunteered their system hardware to conduct the benchmarks. I would like to thank the individuals and organizations that participated by name in order of samples: The Army Cyber Institute, Jon Hawthorne, Michael Devens, Alexander Mentis, Craig Maybee, Brian Lebiednik, Stephen Hamilton, Alexander Kedrowitsch, Ryan Depping, Aaron Williams, Samuel Lee, Joseph Sagisi, and Corey Crosser.

Table of Contents

List of Tables

List of Figures

CHAPTER ONE

Since the National Institute of Standards and Technology (NIST) announced the advanced encryption standard (AES) in 2001, it has become the standard symmetric-key, block-cipher for use in both commercial and government sectors (NIST, 2001). Today, AES is in widespread, global use and one of the most popular cryptographic algorithms of the twenty-first century (Bogdanov, Khovratovich, & Rechberger, 2011; Soleimany, Sharifi, & Aref, 2010). Because of its popularity, ensuring the security of AES is vital (Jayasinghe, Ragel, Ambrose, Ignjatovic, & Parameswaran, 2014). Consequently, the cryptographic community placed the evaluation of the security provided AES as one of its top priorities (Alghazzawi, Hasan, & Trigui, 2010). The scholarly cryptographic community ensures the security of algorithms by evaluating the key lengths, available processing power, and ensuring no shortcuts exist by emulating adversaries attempting to subvert or break the algorithms (Güneysu, Kasper, Novotný, & Paar, 2008). Since the key lengths intuitively have an adequate margin of security, the literature focuses on emulating attacking adversaries (Burr, 2003).

Topic Overview/Background

Throughout history, the art of communicating secrets, cryptography, was a prominent tool for rulers, militaries, and uprisings. Since the dawn of transformative writing in ancient Egypt, components of cryptography have independently evolved and assembled leading to the state of cryptography today (Kahn, 1996). Modern cryptography, including AES, is secret only in the key; the details of the algorithm are public. This principle dates back to the late 1800s as Kerckhoffs realized the need for reformed cryptography during the rise of the telegraph, and published *Cryptographie Militaire* (Singh, 1999). This era of cryptography began in the 1970s as the National Bureau of Standards (NBS), the predecessor organization to NIST, began the

process of defining and eventually selecting the first public encryption standard: the data encryption standard (DES) (NBS, 1972).

DES remained secure until the late 1990s when its short key length rapidly caused the algorithm to become irrelevant. Shortly thereafter, NIST selected AES as the next block cipher standard. It quickly became the focal point of the cryptographic community (Biryukov & Großschädl, 2012; Bogdanov, Khovratovich, & Rechberger, 2011). As the second decade of the twenty-first century was coming to a close, the interest in the longevity and continued use of AES has grown. DES lasted around 20 years (Curtin & Dolske, 1998; NBS, 1977). AES was designed to last at least 20 years (Baudron et at., 1999; Burr, 2003). NIST monitors AES and ensures it is secure for continued use every five years for the next five years, but both the state of computing and the importance of technology today are very different from the late 1990s when NIST began the process of selecting AES (Barker & Roginsky, 2015).

The connected world is much larger today than it was when DES was the leading cryptographic solution. Up from less than one percent in 1995, the increase in the last two decades is massive. Based on data from the International Telecommunication Union, the World Bank, and the United Nations, nearly half of the world's population uses the Internet (Internet Live Stats, 2018). While users are one component of the connected devices, many more devices are online than distinct users. These devices range from the personal devices used to access the Internet – such as smartphones, tablets, and personal computers – to the commercial and corporate devices that host the Internet and its many services – such as routers, switches, and servers. Given the massive increase in users and the reliance on technology and connectivity, ensuring the security of those communications is even more critical.

The state of computing is also significantly advanced from its state in the late 1990s. One example of this change is the availability of processing power. In 1997, the top supercomputer in the world, Intel's ASCI Red, was capable of 1,453.0 giga-floating point operations per second (GFLOPS) (Top500, 2017). Today, a single consumer device, the Nvidia GeForce GTX 1080 TI, is capable of 11,340 GFLOPS (Tech Power Up, 2018). In just 20 years, many consumer homes contain the equivalent processing power of the world's foremost supercomputers. Other innovations, including tri-gate transistors, multi-core architectures, improvements to lithography, and expanded instruction sets, including AES hardware instructions, have had a profound influence on the availability of processing power and the level of global computational potential. These changes, combined with the massive increase in the number of interconnected devices, causes concern about the events that brought an end to DES occurring with AES.

Problem Opportunity Statement

AES is among the most popular cryptographic algorithms in use today (Bogdanov, Khovratovich, & Rechberger, 2011; Soleimany, Sharifi, & Aref, 2010). It is the premier symmetric-key, block-cipher in both the government and commercial sectors. Given its importance, the literature concerning its security is robust. The literature also includes implementations, components of the algorithm, and various cryptographic attacks. Despite the importance of its security, the study of attack feasibility has a limited stake in the literature. A component of the limited share in the literature for attack feasibility may be the perceived security of the key lengths, as even the smallest key length was predicted to remain secure for close to six decades (Burr, 2003).

Another component of the limited attack feasibility coverage in the literate may relate to the current state of practical attacks. Practical attacks seek to subvert the algorithm rather than directly defeat it. All known, feasible attacks fall into this category. The lack of the

cryptographic complexity component of an attack does not encourage the community to consider sheer feasibility. Instead, rough estimations rely on approximations, such as Moore's Law, to project when the availability of computing may intersect with the strength of each key length.

Examining the relationships between traditional benchmark results and AES performance benchmarks for different, general-purpose hardware configurations may identify correlated or causal relationships in addition to potential grouping around hardware types. Those relationships could provide a foundational component for practical modeling of cryptographic attack potential by establishing a ratio for general-purpose systems. The practical modeling may enhance the accuracy of attack feasibility assessments and help the security community by better projecting key-length vulnerabilities allowing industry more time to transition to stronger AES keys.

Purpose Statement

The purpose of this research was to contribute to the security community at large through this addition to the AES body of knowledge. This dissertation accomplished the contribution in the categories of attack feasibility and performance. Additionally, the study may serve as a foundation for future attack feasibility research by expanding the scope of the experiment to additional categories of hardware platforms. Finally, the literature review provides a recent summary of the breadth of the literature surrounding AES from the background of the previous standard to the most recent developments.

As with the attack feasibility literature on the preceding algorithm, DES, AES attack feasibility literature could be grouped into three categories. The categories of strictly theoretical, special hardware, and general hardware encompass the spectrum of attack feasibility literature and considerations. With DES, the literature, shifted from theoretical, through special purpose, to general purpose as processing power approached the point of feasible attacks against the key length. This study focused on the general hardware subset of attack feasibility. Although the

results regarding attack feasibility were expected to be far from actually feasible, the study used the practical approach of benchmarks.

The goal of the attack feasibility component of this research was not to discover that attacks are feasible. Rather this study sought to assess the pure feasibility of an attack, by testing the ratio between traditional benchmarks and AES benchmarks on a variety of hardware platforms. If the results indicated that general performance correlates or is a valid predictor for AES performance, the ratios could then predict the attack potential of future systems. Additionally, the ratios were expected to differ from system to system. Groups around hardware components, including the amount of memory, the type of processor, or the presence of AES special instructions, may provide additional utility or applicability to future hardware platforms.

Research Question(s)

The purpose of the study was to address two concise research questions. These questions were founded in the problem statement and provided the basis for the hypotheses.

R1: How correlated are the results of traditional benchmarks and AES benchmarks conducted on systems in the sample population?

R2: How do the hardware configurations of systems in the sample population affect the level of correlation between traditional benchmarks and AES benchmarks?

Hypotheses

This study tested four hypotheses regarding the relationship between traditional and AES benchmarks to address the research questions. Collecting both benchmarks and hardware configuration information, including the processor, memory, and presence of AES hardware instructions, comprised the testing of the hypotheses. The first alternate hypothesis, $H0_A$, states that statistically significant correlations will exist between the results of traditional benchmarks in floating-point operations per second (FLOPS) and AES benchmarks. The rest of the

hypotheses relate to the specifics of the hardware. The first hardware component considers the presence or absence of AES instructions, which H1 addresses. However, this hypothesis is a unique case since most modern processors have AES instructions and may not make it possible to determine its effect. The processor type and memory may also have an impact on the results, which H2 and H3 address to determine if any groups in the results exist. If groups do exist, they may allow a component, or components, to be used as a predictor. The null and alternate expressions of each hypothesis are expressed as follows:

$H0_0$: No statistically significant correlations exist between traditional benchmarks and AES benchmarks conducted on systems in the sample population.

$H0_A$: Statistically significant correlations exist between traditional benchmarks and AES benchmarks conducted on systems in the sample population.

$H1_0$: The AES hardware instructions component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H1_A$: The AES hardware instructions component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H2_0$: The processor type component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H2_A$: The processor type component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H3_0$: The memory component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H3_A$: The memory component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

Theoretical Perspectives/Conceptual Framework

Cryptographic attacks against AES provide components such as memory, data, and compute as requirements for their execution (Nechvatal et al., 2000). Memory and data are generally either feasible or infeasible without concern for the rapidly changing state of computers today. Conversely, compute is based on the margin between the necessary compute to break a key size and the available compute. Focusing on compute, the literature contains both theoretical and practical approaches to attack feasibility. Attack feasibility was considered as early as the AES conferences to determine which algorithm would be selected and continues to be considered by NIST today.

Early cryptographic attack feasibility research for AES focused on the raw key length and on partial implementations (Ferguson et al., 2000). The key length was a much-needed improvement on 56-bit key length of DES, which became so feasible that it could be broken in a day (McNett, 1999). It was suggested that the 128-, 192-, and 256-bit keys would endure for an estimated six decades to several millennia (Burr, 2003). These estimations were beyond adequate at the time but as compute continued to improve, the nature of feasibility estimations evolved. The simple approximations addressed how processing power progressed to complex performance considerations for modern, AES accelerated, and special-purpose systems.

Throughout the body of knowledge, performance benchmarks typically focus on practical usability (Schneier & Whiting, 2000; Yuan, He, Gong & Qiu, 2014). Some consider the component of attack feasibility with performance (Biryukov & Großschädl, 2012; Manavski, 2007). To attack an algorithm, including brute-force key search, an adversary must conduct components of the algorithm repeatedly. The components usually include the key expansion and a trial decryption. This similarity causes both the usability-focused and cryptography-focused AES performance research to have relevance to this study.

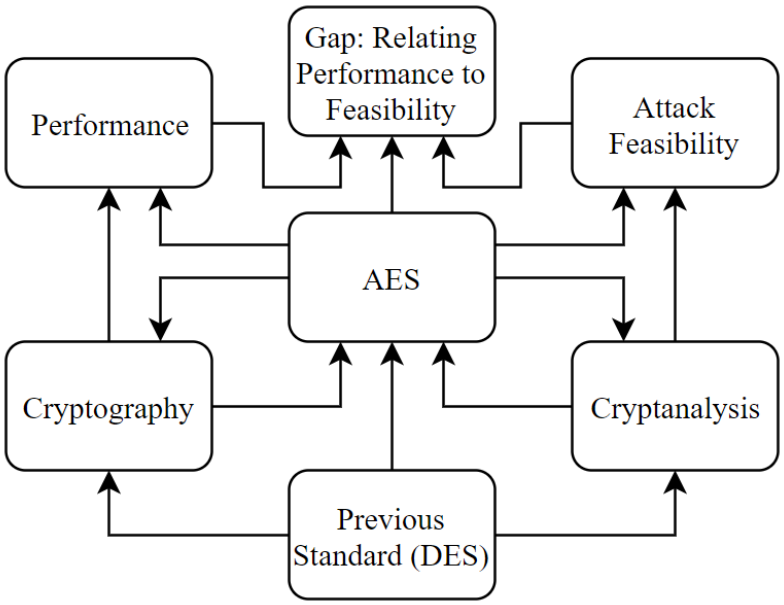

Figure 1. Related Topics by Daniel S. Hawthorne. Copyright 2018

This research draws its theoretical framework from several sources in the body of knowledge. Figure 1 illustrates those sources as they relate to each other with the foundation of the previous standard through the gap in the literature that this research sought to address. It includes the background of DES for context and the breadth of the cryptographic literature on AES. It follows the feasibility assessment component of the approach used by Biryukov and Großschädl (2012), which considered a best-case attack on a large-scale theoretical, special-

purpose system. This current study considered the hypothetical, best-case scenarios while placing emphasis on the more practical approach of benchmarks. It draws the benchmark and performance ratio components from the system performance planning and AES performance sections of the literature.

Assumptions/Biases

The several personal biases, which lead to assumptions about the topic area, were identified as being acquired during the literature review process; they are consequently based on the body of knowledge. They are described here to ensure they do not manifest in the literature review without proper references. Although these assumptions and biases are based on the body of knowledge and on the facts, they are not proven in the scope of this study or they cannot be proven at this time. It is because of the lack of proof that these conjectures fall into this category.

The first assumption is that AES in all key lengths is adequately secure today. The second is that AES will fall into the feasible attack range sooner than initially projected. The third is that implementation weaknesses, side-channel-attacks, and other means of reducing the complexity of cryptographic attacks will continue to present the greatest threats to the security of the algorithm. The fourth is that the impact of a single AES key-length falling into the feasible range without significant time for the technical industry to move away from that key length would be significantly more damaging today than DES was in the late 1990s. These presuppositions led to the topic decision for this dissertation.

The first assumption, that all AES key lengths are adequately secure today, is based on assessments throughout the body of knowledge ranging from early assessments of the selection process in 2003 to the latest NIST assessment in 2015. In the summary document, which depicts the selection criteria process and attack estimations, Burr (2003) describes the state of attack feasibility estimation referencing Moore's Law as the "best estimator we have" (p.45). This early

estimation predicted the 128-bit key length would be as secure in the year 2066 as an 80-bit algorithm was in 2003. At the time, an 80-bit key was well beyond the margin security for even the most sensitive usage.

NIST reviews approved security functions, including encryption algorithms, hash functions, key exchange, and secure random number generators, every five years. The review process involves providing a projected timeframe for each algorithm and key length in which the key length for the algorithm is expected to provide adequate security. NIST does not provide a projected timeframe for continued use if the algorithm has no near-term security concerns. In November 2015, NIST again reaffirmed all key lengths of AES as being adequately secure without providing a projected timeframe (Barker & Roginsky, 2015). The NIST affirmation is an authoritative assessment of the near-term security provided by AES, but its focus remains on the five-year usability.

The second assumption, that AES will fall into the feasible attack range sooner than initially projected, relates to the designed longevity of AES during the selection process, which was at least 20 years (Baudron et al., 1999). The initial feasibility assessments, however, expected the key lengths to endure for around six decades (Burr, 2003). Many revolutionary technology changes have influenced the departure from the rate of computing change that was present in the 1990s. These technologies include multi-core CPUs, hardware AES acceleration, and eventually the prevalence of quantum computing. The NIST assessment for continued use is robust and detailed, however the initial longevity assessments were lacking (Barker & Roginsky, 2015; Burr, 2003).

The third assumption, that other means of subverting encryption or reducing cryptographic attack complexity will present the greater threat, has held true from the AES

selection process until now. The only attack that has lower compute than brute force requires an infeasible amount of cipher text in exchange for a minor reduction to compute (Bogdanov, Khovratovich, & Rechberger, 2011). Although this attack is by definition a break, attacks which require special, impractical circumstances such as partial implementations, related-keys, side-channels, or access to plaintext will pose the more computationally feasible cryptographic threats. The greatest threats are not purely cryptographic or necessarily technical. These threats comprise external factors such as the ever-present human element, operating system vulnerabilities, and other software on the system.

The fourth assumption relates to the impact of cryptographic attacks against AES becoming feasible. The example case is DES. Attacks became feasible against DES prior to a replacement (McNett, 1999). Additionally, DES lacked a structured transition, leaving a gap as it was still in use seven years later (Kelley, 2006). The gap was partially filled by the triple data encryption algorithm (TDEA), which simply increased the size of the key by conducting DES two to three times (NIST, 1999b). However, due to DES becoming vulnerable prior to projections, NIST enacted TDEA the same year that DES became vulnerable, leaving a gap in security while industry adapted to the new standard. Given the increase in the size and reliance on the Internet, a gap in the security provided by AES would cause substantially greater impact than the gap in security of DES.

The combination of these intuitive factors led to further investigation into the possibility that AES may fall into the feasible attack range sooner than intended. Although AES is secure today and will remain secure for the immediate future, barring a revolutionary jump in the rate of technological progress, the ability to further out and more accurately project key length vulnerabilities may allow industry a larger transition time to avoid the issues that arose when

DES transitioned. This endeavor is even more critical since a transition without a major gap in the security provided by AES is increasingly important relative to the usage and dependence on connectivity and security.

Significance of the Study

The research topic described in this dissertation sought to add to the body of knowledge and the security community in two ways. The first relates to the thorough nature of the literature review. The study comprises an exhaustive attempt to include every piece of available, relevant research relating to AES, including attacks, attack feasibility, performance, and security, in addition to related background information including DES and the changes to state of computing. The inclusive combination of these elements is not found elsewhere in the body of knowledge. The comparison of related publications and the logical progression of the literature review makes it especially applicable to this study and future attack feasibility research.

The second, potentially significant contribution relates to the possible findings of the study. If the relationship between traditional benchmarks and AES performance shows that AES performance is strongly correlated to traditional benchmarks or that traditional benchmarks may be used as a predictor for AES performance, the results may be able to help project future attack feasibility. Where traditional performance-based attack feasibility estimations rely on gross approximations, such as Moore's law, the correlation of AES performance and FLOPS may provide a more accurate means of estimating attack feasibility potential based on readily available information (Burr, 2003). The culmination of the predictive potential has at least three use cases: better projecting the need for key length migrations, determining the attack potential of new technologies, and determining the attack feasibility of new attacks.

The gap in security provided by DES, as brute-force attacks became feasible long before industry adopted an alternate standard, demonstrates the need for longer timeframes for industry

to adapt to new standards before the previous standards become vulnerable (Curtin & Dolske, 1998; IETF, 2006). The multiple key lengths of AES are a mitigating factor by design, however, the results of this research could allow additional warning beyond the five years of expected usability that NIST provides (Barker & Dang, 2015). This predictive model could be applied to many different subsets, from global compute to individual systems, to determine attack potential against different key lengths.

The second use case involves the attack potential of new technologies. A breakthrough or revolutionary new technology may result in uncharacteristically significant improvements in processing power. The recently announced Nvidia Titan V (2017) is approaching the scale of the most powerful supercomputer in 2005 (Top500.org). Twelve years is not a huge gap between cutting-edge, global computing power and computing power available to consumers. A revolutionary new technology might further reduce that gap. If the gap were abruptly shortened, the processing power available could quickly make otherwise infeasible attacks feasible. This research is especially applicable to improvements in large scale computing because it uses the same benchmark instrument that Top500 uses to measure supercomputers: high performance Linpack (HPL).

Determining the attack feasibility of new attacks is another potential application of this research. Although most attacks seek to subvert encryption, as they are generally more feasible, even some subversions require AES operations. Furthermore, a new break of AES with significant reduction in processing power required to conduct the attack may require a given number, likely still a massive number, of AES operations. Many attacks include a paragraph or even less about feasibility. This study could be used as a foundation for more accurate

assessments of processing power requirements with little or no more effort required than the rough estimation usually included in cryptographic attack publications.

Delimitations

The delimitations are in place on the experiment to constrain the scope and variables to a manageable number. The delimitation reduces the ability to extend the results to a larger set of processing platforms, but it is necessary to keep the experiment uniform. The delimitation is the inclusion of only the x86_64 architecture sometimes referred to as x64. Although several other system architectures exist, including ARM and GPUs, the x86_64 is the widespread architecture found in the majority of personal computer and server CPUs. ARM and GPUs do represent significant portions of the global compute, but the design of a collection instrument to measure performance on the vastly different platforms would essentially require three entirely different collection mechanisms, which would not align well with a single study.

Limitations

Limitations relate to the human element. Since the only component of this research involving humans is in facilitation, the currently identified limitation related to potential mistakes during the facilitation process. The limitation is based on temperature. Overheated systems throttle performance to reduce heat. The mitigation is two-fold. The instructions that accompany the collection instrument include a wait period for systems to cool down if they were in heavy use before booting to the collection instrument. Additionally, the instructions request users to not run the tests on systems when the ambient temperature is outside their normal operating temperature range. The second component of the mitigation is found in the design of the collection instrument and the nature of computer systems. Unlike humans, computers complete tasks with near identical performance each time. Identical configurations and repeated tests throughout the sample will help identify systems that have throttled due to overheating.

Definition of Terms

This section contains the definition of the terms used throughout the study to provide clarity, context, and as a central location for reference. The definitions align with those most frequently found throughout the body of knowledge, providing clarity where variations exist. Some additional terms are defined specifically for the context of this study, such as benchmark and system, which may have a variety of definitions in different contexts.

Advanced Encryption Standard (AES) is the symmetric-key, block-cipher standardized by the National Institute of Standards and Technology (NIST) in 2001 (NIST, 2001). Today, AES is in widespread, global use and one of the most popular cryptographic algorithms of the twenty-first century (Bogdanov, Khovratovich, & Rechberger, 2011; Soleimany, Sharifi, & Aref, 2010). Internally, AES is a subset of the cipher, Rijndael, which was the winning algorithm of the AES candidate competition. NIST specified the block size, rounds, and key lengths for AES based on the available configurations for Rijndael (NIST, 2001). The specifications included 128-bit block, 128-, 192-, and 256-bit keys and 10-, 12-, and 14-rounds per key size respectively. The algorithm begins with key expansion to create a unique key for each round. The cipher itself involves byte substitution, shifting rows, mixing columns, and the addition of the round key for each round (Daemen & Rijmen, 1999).

Benchmark refers to the process of measuring system performance. Traditional benchmarks measure general purpose compute or operations that easily approximate general-purpose performance, such as FLOPS. Many benchmark solutions exist in the literature and industry, but this study employs only a couple. Linpack is a standard FLOPS benchmark utility used by leading performance evaluation organizations, including Top500. It is used by this study to measure FLOPS. Cryptsetup, an encryption utility included in many Linux distributions,

contains an encryption and decryption benchmark, which includes AES. It is used to measure AES throughput, which is translated to AESOPS for this study.

Cryptanalysis is the processes of defeating of cryptosystems (SANS Institute, 2001). It involves gaining access to the plaintext or recovering the key by means other than the intended use of the cryptosystem. Kerckhoffs introduced the necessity of cryptanalysis as means of determining the security provided by a cryptosystem in the late 1800s (Kahn, 1996). Since then, the role of cryptanalysis has grown from simply trying to break an adversary's cryptosystem to gain a military or strategic advantage to trying to break every trusted and used cryptosystem to determine its security (Singh, 2000).

Cryptography is the art of communicating secrets through techniques including the transformation of writing, substitution of characters, and diffusion of the plaintext throughout the ciphertext.

Cryptology refers to the combined study of cryptanalysis and cryptography (SANS Institute, 2001).

Data Encryption Standard (DES) preceded AES. It was announced by National Bureau of Standards (NBS), the predecessor of NIST, in 1977 as the first open encryption standard (NBS, 1977). International Business Machines (IBM) designed the internal algorithm in 1974, although it was slightly revised before standardization in 1977. Attacks against DES became both feasible and practice between 1997 and 1999 (Curtin & Dolske, 1998; McNett, 1999). Even though DES was replaced by the triple data encryption algorithm (TDEA) in 1999 and AES in 2001, it was still in use in 2006 (Kelly, 2006; NIST, 1999b; NIST, 2001). As the preceding standard to AES, the process of selecting DES, the lifespan of the algorithm, and the issues with transition away from DES provide insight into those components of the current standard.

National Institute of Standards and Technology (NIST) is a robust United States government organization with a variety of roles. This study considers only the computer security division of NIST, specifically the role involving the selection and maintenance of cryptographic standards. NIST and its predecessor organization, the National Bureau of Standards (NBS) led the selection process for standards examined in this study including DES and AES in addition to many others. NIST also monitors the margin of security provided by each cryptographic standard and provides recommendations for continued use (Barker & Dang, 2015).

Processing Power refers to the measurable performance of a system. In this study, processing power is synonymous with compute. The measurable performance is usually in terms of an operation type per second. Floating point operations per second (FLOPS) is a well-known performance standard that is published and measured by various organizations in addition to vendors. The performance in terms of AES operations per second (AESOPS) is a less common unit, but it is used throughout this study as key component of AES performance estimation.

System refers to any device with a microprocessor. These devices range from the largest supercomputers to the smallest Internet of Things (IoT) devices. They include devices that access services such as personal computers, smart phones, and wearables. They also include devices that provide services like routers, servers, and other network devices. Multiple systems may comprise a system internally or in a distributed fashion. Systems have two overarching categories: general-purpose and application-specific. Although application specific may solely comprise of application specific integrated circuits (ASICs), they are still considered systems for the purpose of this study.

General Overview of the Research Design

The current study examined the relationships between traditional benchmarks using FLOPS and specific benchmarks using AESOPS to answer the research questions and fulfil the

purpose statement. The collection phase used a Linux-based, bootable USB collection device. The collection devices were distributed to facilitators who volunteered their systems for the study. Volunteers facilitated the research by booting systems to the USB collection devices. Each device ran benchmark scripts for AES and the traditional performance value of FLOPS in addition to collecting system information including the CPU and memory. The scripts reported the results to an EC2 server that validated input and stored the results. The collection phase was repeated until an adequate sample size and variety was reached.

The sampling will be purposive, specifically aimed for a variety of hardware configurations. Repeat configurations served as additional validation of the consistency of the benchmark results. Once adequate sample and variety were reached, the results were transferred to a spreadsheet using a Python script. Before continuing, further validation of the data and handling of errors and outliers were considered. Once the data was validated, statistical analysis commenced to measure the relationships between the benchmark results and impacts of the system components. The results of those tests were used to either reject the null or fail to reject the null hypothesis.

Summary of Chapter One

This chapter introduced the proposed research topic. It included the identification and articulation of the problem which this research sought to address. It demonstrated the opportunity to contribute by conducting the proposed research. The chapter also introduced the overarching purpose of the research and the larger problem that purpose falls into. It mentioned the specific research questions and the hypotheses that were addressed. It also included the biases, assumptions, limitations, delimitations, and the overview of the research design, detailed in Chapter 3.

Organization of Dissertation

Chapter 2 of the dissertation is the literature review. It contains a topological review of the body of knowledge, which is organized chronologically within each topic. The body of knowledge includes AES, cryptography, attack feasibility, the changing computing environment, and performance planning. Those fields form the background for the study. The gap exists within the union of those fields, where attack feasibility and performance planning meet.

Chapter 3 of the proposal begins with the background and traditions behind security, encryption, and cryptographic research. It continues to present the details of the problem addressed by the research and the opportunity to contribute to the community. The research questions, hypothesis, and design precede the details of the design, collection procedures, and quantitative methods, which will be used to analyze the data in Chapter 4. Interpretations of the findings and future recommendations are presented in Chapter 5.

CHAPTER TWO

This chapter contains a topological, chronological review of the literature surrounding the topic. It begins with the context of the events leading to the advanced encryption standard (AES). This context includes the previous standard, the data encryption standard (DES), the lessons learned which influenced the AES criteria, and the advances in cryptography which DES brought about that apply to AES. The chapter continues with the announcement and criteria for the new standard, AES, the selection process, and the reasons why NIST selected the algorithm, Rijndael. The state of computing throughout the lifespans of both DES and AES is also relevant. Computing from the time that DES was selected to the time that it became vulnerable parallels the potential for the changing compute environment to effect AES. Attack feasibility throughout both DES and AES are also included and closely relate to one another and to the changing state of computing.

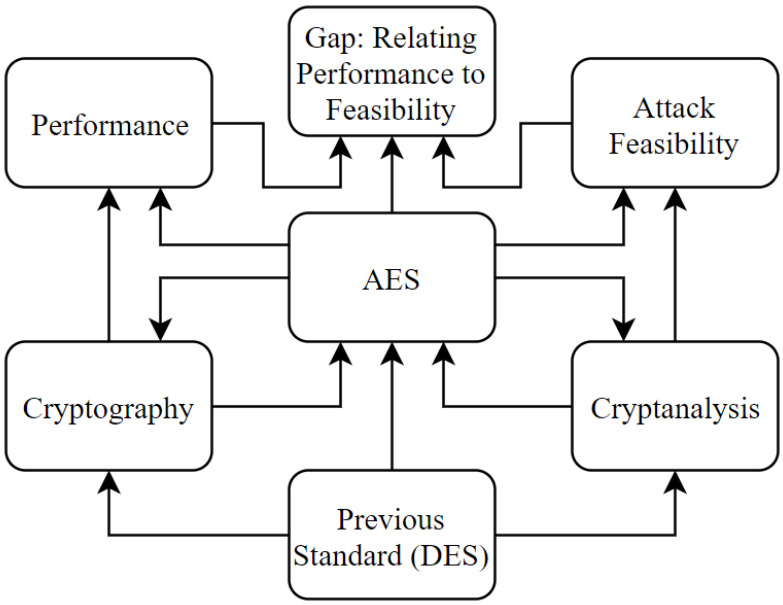

Figure 2. Related Topics by Daniel S. Hawthorne. Copyright 2018

Once NIST announced AES, the cryptographic community shifted focus to the new standard. Early attack feasibility estimations, which initially validated algorithmic endurance and ensured the minimal secure variant was adequately secure, relied on very simple approximations and often referred to Moore's Law to anticipate the rate of change. These early estimations relied on the simplest form of performance planning, but they did establish the relationship between those fields. Performance planning includes considerations for different system components and various methods of predicting system performance. Attack feasibility combines the performance considerations and cryptographic attacks to determine the security provided by AES. Figure 2 depicts the relation of the topics covered in this chapter which lead to the gap addressed by this study.

Foundation of the Modern Cryptographic Era

Cryptography, the communication of secret messages, originates from ancient Egypt (Kahn, 1996). The first practice considered a cryptographic technique was the transformation of hieroglyphics to dignify writing. Cryptography evolved from that simple beginning through history as it found use in military, political, insurgent, and religious contexts. Classic cryptography had various forms at it progressed ranging from phonetic and alphabetic substitutions to simple transpositions. The fundamental cryptographic concepts of substitution, replacing a symbol or letter with another, and transposition, the shifting of characters, endure today as components of modern cryptography. Cryptography evolved independently in different regions for different reasons, but these foundational concepts were found in many different independent branches.

Although substitution and transposition remain as foundational to cryptography today, the first historic event that had a profound impact on modern cryptography did not occur until the emergence of the telegraph. The invention of the telegraph in the nineteenth century marked a

major change in technology and resultantly in cryptography. The telegraph would replace hand-carried messages as the transmission medium of secret messages. As this transition began, Kerckhoffs, a leading cryptographer at that time, published *Cryptographie Militaire* (Kahn, 1996). His work, published in 1893, included a comprehensive history of cryptography encompassing every known method. He additionally presented two overarching principles and six specific requirements for cryptography. His first principles involved the difference in cipher requirements for short term exchanges of written messages and the requirements for military use for an extended period of time.

The second overarching concept was that cryptanalysis, the process of defeating a cipher without knowledge of the key, was the only way to measure the cipher's strength. This principle endures today; it is the same approach used for assessing the security of DES, all of the AES candidates, and every other form of cryptography today. Of the six specific requirements, the second became a foundational concept of open cryptographic standards; the presence of all six would be indicative of the ideal cipher. World War I had a significant impact on cryptography with William Frederick Friedman coining the term cryptanalysis and beginning the governmental inclusion of cryptanalysts. Cryptography also had a role in World War II with the Enigma machine. These events were important precursors to the modern cryptographic era, but they were arguably not in themselves as crucial as the work done by Kerckhoffs in the late 1800s (Singh, 2000).

In 1972, the National Bureau of Standards (NBS), the predecessor organization to NIST, began the process of defining and eventually selecting the first public encryption standard (NBS, 1972). This event began the cultural departure from disparate, secretive, and closed-source cryptographic methods to standardized, open-source solutions. This cultural shift marked a large

step toward the cipher that Kerckhoffs described nearly 80 years earlier. Open standards intrinsically met the second specific requirement, that compromise of the system not inconvenience the correspondents, which is often restated as *a cryptosystem should be secret only in the key*. Six years later, the data encryption standard (DES) emerged as the product of this transition. DES endured for about 20 years before attacks entered the feasible range (Electronic Frontier Foundation, 1998a).

Leading to the Advanced Encryption Standard

The events leading to the call for a new encryption standard are an important contextual component of the background of AES. The components of the previous standard, the data encryption standard (DES), and the events leading to its abrogation, influenced initial requirements for AES. The attacks against DES entered the realm of financial and practical feasibility. Although an interim solution was in place, it had its own set of problems. The risk associated with continued use of DES were too great. In response to these concerns, the National Institute of Standards and Technology (NIST) announced the call for AES candidates in 1997 (NIST, 1997).

In 1972, the National Bureau of Standards (NBS), the predecessor of NIST, initiated the effort to develop computer security standards (NBS, 1978). At the time, practical encryption was just emerging as a security concept. The call for candidates had the requirements that the algorithm would be publicly available, implementable on various platforms, and unambiguous, having a single, symmetric key. In 1973, the first call went unfulfilled. The only notable suggestion was a digital implementation of a one-time-pad (Davis, 1978). The failure of the first solicitation to find a suitable candidate produced positive interest in the community. By the second solicitation, several submissions were ready. One of those submissions became DES.

Developed in 1974 by IBM, the algorithm officially became DES in 1977. The initial projection for a feasible attack was 17 years to build a 70 million USD system that would be able to recover one key per day on the megawatt power scale (NBS, 1977). This margin of security was acceptable at the time. Over the lifespan of DES, the algorithm endured various forms of cryptographic evaluations including feasibility assessments and criticism of the short key length of 56-bits. New cryptographic techniques and improvements to known techniques emerged as the security community continuously evaluated the first encryption standard.

In 1993, 16 years into the lifespan of the algorithm, the cryptanalysis of DES was approaching maturity; three breakthroughs occurred which predicated the downfall of DES. The first breakthrough was a cryptographic technique. Biham and Shamir (1993) discovered and published the new technique, which become known as differential cryptanalysis. Differential cryptanalysis was effective at reducing the attack complexity of substitution permutation cryptosystems. A month before publication, Coppersmith, one of the members of the design team announced that they were aware of differential cryptanalysis in 1974 and took design precautions to partially mitigate the method but kept it a secret for the sake of national security (Biham & Shamir, 1993).

The second breakthrough was an improvement on linear cryptanalysis. It was highly capable against DES implementations that used half, 8-rounds, of the specified rounds, even without the normally required known-plaintexts. The improved technique only required 2^{29} cipher texts for English ASCII or 2^{37} cipher texts for other data given the 8-round DES implementation (Matsui, 1993). Although a cipher-text only attack against partial implementations, the attack required significant known-plaintexts for the full 16-round

implementation of DES. The margin of security provided by even the full implementation of DES was closing on multiple fronts.

In 1994, Coppersmith responded to Biham and Shamir by publishing a feasibility assessment of their attack and a reaffirmation that differential cryptanalysis was a design consideration from the beginning. Coppersmith concluded that the attack presented by Biham and Shamir demonstrated the success of the algorithm because it required an infeasible number of chosen plaintexts for success, on the order of 10^{15} bytes. Although the open publication of differential cryptanalysis was a huge breakthrough, it did not directly lead to the downfall of DES. It did influence the AES design requirements as any new algorithm required resistance to this cryptographic technique.

The final breakthrough in 1993 related to the key length of 56-bits. The short key length did eventually lead to the downfall of DES. Wiener (1993) designed a particularly effective, scalable DES key search chip. Given one million USD of the search chips, the resulting machine would be capable of an exhaustive search of the key space every 3.5 hours. Although this machine is not the first theoretical assessment of its type, it was by far the most capable and economical. This brute-force method was quickly approaching the feasible point.

As computers continued to increase in availability and capability, the 56-bit key length of DES became the focus of the cryptographic community. In January 1997, RSA Laboratories began a series of secret-key challenges, which would continue for over a decade. The challenges offered cash rewards for DES and RC5 key-recovery (RSA Laboratories, n.d.). RSA Laboratories designed these challenges to assess the relative security provided by DES and RC5. The first set of DES challenges, released on 28 January 1997, included what later became known

as DES I (Curtin & Dolske, 1998). Two additional DES challenges followed as organizations completed the challenges with increasing efficiency.

In June of 1997, a distributed computing project, named "DESCHALL" for DES challenge, was awarded $10,000 for completing the first DES challenge. The coordination of approximately 10,000 mid-1990s clients completed the exhaustion of the key space in under three months (Curtin & Dolske, 1998). The leaders of the project published their method and some observations from the orchestration of this historic success. The project marked the first known use of distributed computing for a cryptographic purpose but also proved that distributed computing could overcome the inefficiencies of software implementations.

The second DES challenge, DES II, consisted of two objectives. The first was to improve on the rate that a distributed group could discover the key; the second was to solve the key by another means besides exhaustion through a distributed approach (Electronic Frontier Foundation, 1998a). Using a larger distributed pool and just 39 days in 1998, the first objective revealed "The secret message is: Many hands make light work" (McNett, 1998, p. 1). The Electronic Frontier Foundation (EEF) solved the second in 56 hours using a 200,000 USD application specific machine, "Deep Crack", which utilized application specific integrated circuits (ASICs). This effort concretely demonstrated that financial investment quickly leads to broken keys as they recovered, "The secret message is: It's time for those 128-, 192-, and 256-bit keys" (EFF, 1998a, p. 3).

The final DES challenge, DES III, presented by RSA Laboratories, required a further improvement upon the speed (RSA Laboratories, 1999). The third challenge was solved by a combination of the two parts of DES II. Together, Distributed.net and the EEF "Deep Crack" machine recovered a key in less than a day that revealed the message "See you in Rome (second

AES Conference, March 22-23, 1999)". Distributed computing technologies together with EEF published an immediate press release of their results (McNett, 1999).

DES was indisputably insecure. Through a series of DES challenges, from January 1997 to January of 1999, the cryptographic community proved that brute-force search of the 56-bit DES key space was not only feasible, it was possible in less than a day (McNett, 1999). In response to the growing threat of key-length vulnerability, NIST approved the triple data encryption algorithm (TDEA), also known as triple DES (3DES), as an interim solution. TDEA increased the key size to 112- or 168- bits while conducting three rounds of the DES algorithm (NIST, 1999b). Although NIST provided TDEA as an interim solution as DES became increasingly vulnerable, and despite the ease of transition compared to employing a new standard, the lack of implementation time for the industry to adapt left a gap in security.

Despite DES entering the realm of attack feasibility in 1997 and becoming increasingly practical as time continued, DES was still in use almost 10 years later (Kelly, 2006). Figure 3 depicts the transition timeline.

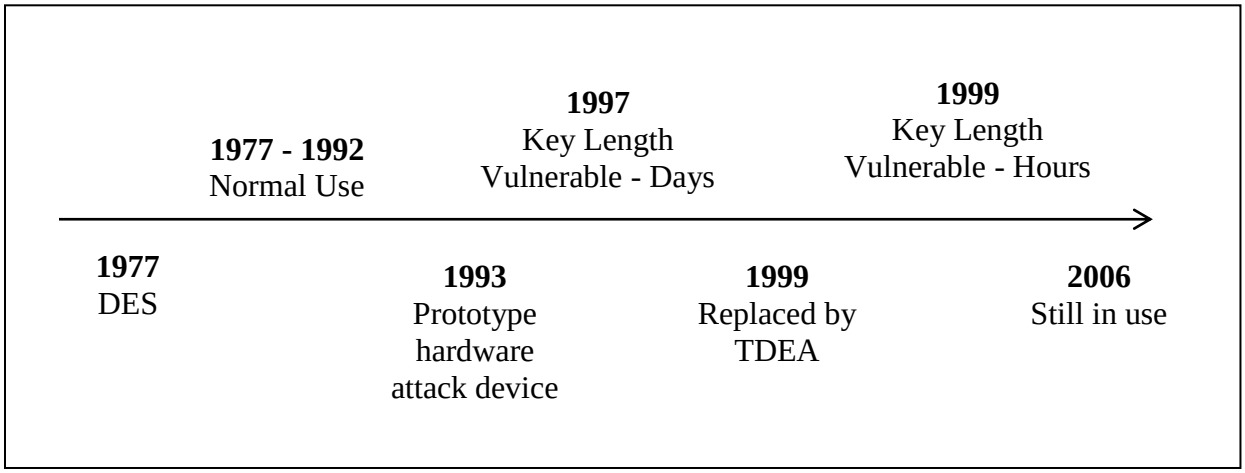

Figure 3. DES timeline by Daniel S. Hawthorne. Copyright 2018

In 1977, the NBS made DES the first symmetric-key encryption standard. It endured 15 years of normal use until 1993 when the key-length was approaching the point of feasible attacks

and the cryptographic attacks targeting the algorithm were approaching maturity. By 1997, a small group of systems could break the encryption in days. Although DES was formally superseded by TDEA in 1999, when it was vulnerable with just hours of dedication and a moderate investment, it was still in use in 2006. In 2005, NIST published an official DES transition plan to warn of the dangers of continued use and a final chance to transition to TDEA or AES (NIST, 2005).

Despite the numerous demonstrated feasible attacks, many organizations continued to use DES. The Internet Engineering Task Force (IETF) is the well-known source organization of standards documents, known as Request for Comments (RFCs), which define the specifications for every component of connectivity today. As an additional confirmation, the IETF published RFC 4772 in 2006 as an informational RFC which conducted an in-depth review of the security implications of using DES. The RFC served as a means of formal warning to entities still using DES of the high-level of risk associated with continued use. Citing Moore's law, RFC 4772 listed the expected cost of hardware and time to complete an attack at $15,625 and 0.5625 days respectively (Kelly, 2006).

After the announcement of AES and the subsequent deprecation of DES, AES cryptographic research efforts largely superseded DES efforts. Additionally, given the feasible range had unequivocally arrived, research that did include DES focused on less specific block ciphers with application to both AES and DES. In 2008, a unique attack platform emerged called COPACOBANA. This platform used field programmable gate arrays (FPGAs) to improve efficiency and parallelism close to that of ASICs, with the added ability to reprogram the FPGAs to conduct attacks against various algorithms. The designers found that COPACOBANA DES

key recovery took only 6 days at the affordable price of 10,000 Euros for complete reusability (Güneysu, Kasper, Novotný, & Paar, 2008).

The last specifically DES-focused contribution explored a graphics processing unit (GPU) -based attack in 2010. The attack utilized a processing improvement specific to DES, known as “bit splicing”, which was made possible by the many-core architecture of GPUs. The authors found that a small cluster of general-purpose systems with Nvidia compute unified device architecture (CUDA) GPUs was a formidable attack platform. Their clusters had between 7,168 and 10,560 CUDA cores, each GPU boasting between 128 and 240 cores; placing their price mark near that of COPACOBANA. The research found that these attacks would take 18 to 25 days to complete (Agosta, Barenghi, Santis, & Pelosi, 2010).

The comparison of the top and bottom, 55nm-based GPUs used by Agosta et al. in 2010 with the high- and low-end, 16nm-based GPUs available in 2016, serves as an anecdotal example of the impact of increasing availability of processing power on cryptography. Table 1 groups the aforementioned GPUs with comparable models today to demonstrate the significant reduction in price per core.

Table 1

GPU comparison price per core

Group	GPU	Year	CUDA cores	Launch price	Price per core
Low-end	GTS 250	2009	128	$199	$1.55
GPUs	GTX 1050	2016	640	$109	$0.17
High-end	GTX 295	2009	480	$500	$1.04
GPUs	GTX 1080	2016	2,560	$599	$0.23

Note. Table derived from data in the Tech Power Up GPU database (2018).

Additionally, GPU clock and memory bandwidth have improved dramatically (Tech Power Up, 2018) furthering the improvement as an attack platform. Since Nvidia GPUs continue

to use the same CUDA language and architecture, a pair of modern systems, each boasting a pair of Nvidia GTX 1080s, would have the same number of CUDA cores that Agosta et al. had with 11 to 28 systems. It is reasonable to assume this pair of systems would complete the attack at least as fast as Agosta et al. in 2010, if not faster thanks to the other GPU architecture improvements.

From an estimated starting point of 70 million USD in 1977 and many years to manufacture, attacks against DES became more feasible and attack platforms became more economical and much faster than initially anticipated (NBS, 1977). By 1997, 250 thousand USD completed an exhaustive attack in a matter of hours. A new standard was needed and called for in 1997; however, DES use and research continued. By 2006, around 15 thousand USD invested in ASICs and a few hours could produce a key (Kelly, 2006). In 2008, COPACOBANA took just 6 days (Güneysu et al., 2008). By 2010, general purpose hardware was as fast as 18 days (Agosta et al., 2010). Today, just a pair of high-end, personal computers could easily outpace the previous general-purpose estimation.

Selecting the Advanced Encryption Standard

In 1997, NIST requested candidate nominations for the new encryption standard. NIST aptly incorporated lessons learned from DES throughout the process. The initial document provided guidance for candidate algorithm submissions including non-technical requirements, technical requirements, and evaluation criteria. It provided nine months of lead time for submissions, potentially a lesson learned from the lack of viable solutions to the first call for DES candidates. Similar to DES, the AES candidates had to be open source and royalty free; any patented candidates had to agree up front to release their algorithm to the public if it was selected as AES (NIST, 1997). The aforementioned non-technical requirements, together with the

technical requirements, formed the prerequisites for consideration. The evaluation criteria would determine the winner.

The technical requirements for the algorithm included three components. The first was the type of cipher. Like DES, AES required candidates to be a symmetric key cipher, meaning encryption and decryption use the same key. Second, the algorithm had to support at least three key lengths: 128-, 192-, and 256-bit. NIST required the lengthened key lengths to mitigate the computing factor that overcame DES, "The secret message is: It's time for those 128-, 192-, and 256-bit keys" (EFF, 1998a). The multiple key lengths were a logical step from the single key length of DES. As exhaustive attacks against the first key length become feasible, another two key lengths would remain within the acceptable margin of security. The final technical requirement pertained to the cipher type. Like DES, AES was required to be a block cipher; unlike DES, it used a 128-bit rather than the 64-bit block used by DES.

The initially stated evaluation criteria included three components, however, many subcomponents became sizable influencing factors as the process of selecting AES continued. The first was security, as in, how much security did each candidate algorithm provide. The second was cost. Cost in this case was in terms of performance and efficiency, although the prerequisite of royalty free was mentioned again, it would have already been met for candidates reaching the evaluation stage. The third was implementation characteristics. Implementation characteristics focused on flexibility including additional block sizes, key sizes, implementations on a variety of platforms, and algorithm modes (NIST, 1997).

The announcement for candidates also included the plan for the first AES conference and two rounds of evaluations covering the above criteria. The author(s) of every viable candidate algorithm were invited to present at the first AES conference. The first round of evaluations had

the unique component of correctness, which ensures that the algorithms performed correctly when compiled, which NIST conducted prior to acceptance (NIST, 1997). Both rounds of testing would reconfirm the key sizes and block size, but would place the majority of the focus on efficiency testing. The testing platforms were disclosed as ANSI C and Java.

The first round of evaluations began prior to the first AES candidate conference (AES1) as NIST reviewed the 21 submitted candidate algorithms. Of the 21 candidate algorithms received by NIST, 15 met the minimum requirements for acceptance and were included in AES1 (Roback & Dworkin, 1999). The first AES candidate conference, AES1, took place in August of 1998. The conference was primarily informational to prepare the community to respond to the candidates and conduct independent evaluations. The author(s) of each algorithm presented their candidate and answered initial questions. A discrepancy on the stated goal for longevity of the algorithm exists in the literature. The AES candidate report states “first twenty years of the twenty first century” (Baudron et at., 1999, p. 1). The conference report states “at least thirty years” (Roback & Dworkin, 1999, p. 98). Regardless, the design was intended to exceed the lifespan of DES.

After the first conference, the community evaluated the candidate algorithms and evaluation platforms in preparation for the second conference. Although the initial choice of platforms, ANSI C and JAVA, required some algorithms to be optimized for different endianness, NIST, along with several independent groups, began the performance comparisons while allowing authors to submit optimized implementations. Other considerations between conferences included initial cryptanalysis, security provided, key schedule, simplicity, trustworthiness, and hardware implementations (Baudron et at., 1999).

The second AES conference (AES2) took place in March 1999. It focused on the public comments pertaining to the candidate algorithms and the initial NIST evaluations. By the end of AES2, the five algorithms that would become the finalists, Rijndael, RC6, Twofish, MARS, and Serpent, stood above the rest of the competition to the attendees (NIST, 1999a). A month after AES2, the first round of evaluations ended and the second round began. NIST announced the finalists in the summer of 1999 allowing the community to focus their efforts onto the remaining five candidate algorithms for the final AES conference the following year.

The third and final AES conference (AES3) took place in April 2000. It included FPGA, platform-specific, and ASIC implementations in addition to more performance comparisons, preliminary cryptanalysis, and an initial consideration of the future resiliency of AES. Schneier and Whiting (2000) contributed a series of extensive benchmarks on a variety of platforms for each AES candidate for each key length and each component – key setup, encrypt, and decrypt. They concluded that the performance on different platforms varied greatly by algorithm and that there was no clear best performing algorithm on all platforms tested, so NIST would have to prioritize the platforms to create an order for the performance category. In 2000, Johnson evaluated the future resiliency of AES for extended long-term use. Johnson concluded that selecting a single algorithm for AES could present issues transitioning if it became vulnerable. A month after AES3, the round two evaluation comment window closed giving NIST about four months to review the research and select the winning candidate algorithm.

Announcing the Advanced Encryption Standard

In October of 2000, NIST announced the selection of the Rijndael algorithm as AES for its simplicity, performance, and efficiency. By November 2001, the federal government completed validation and adopted AES as the federal information processing standard (FIPS) publication 197 (NIST, 2001). FIPS 197 included the detailed specifications, implementation

instructions, examples, and pseudo code to allow the industry and government entities to begin use of the new standard. The Rijndael algorithm, and consequently AES, consists of three intrinsic components: key expansion, encryption, and decryption. Key expansion increases the size of the key so that each round of the algorithm uses a different subkey. The key lengths 128-, 192-, and 256-bit use 10-, 12-, and 14-rounds of internal operations respectively (NIST, 2001). AES uses two-dimensional arrays of bytes, known as "states" for the internal operations of the algorithm.

Each round consists of a non-linear transformation, two linear transformations, and key-based transformation. The round begins by using a substitution box or s-box to conduct a non-linear transformation of the state. The substitution is followed by two linear transformations involving shifting of rows and mixing of columns. The round completes with a bitwise exclusive-OR of the round key and the state. The algorithm repeats the round as necessary based on the key length. Decryption is intuitively the inverse of encryption. Key expansion involves a rotation, a substitution, and a series of exclusive-ORs. Each of the components add strength against cryptographic attacks to the algorithm.

With the new standard published, several organizations provided reviews of the selection process. Richards (2001), with the SANS institute, noted the dramatic shift from the secretive and government influenced process of selecting DES to the process of selecting AES. Unlike the like DES process, NIST held an open, international competition to select AES. The origins of winning Rijndael algorithm demonstrated the paradigm shift. Rijndael was developed by two well-respected cryptographers, Joan Daemen and Vincent Rijmen, of Belgium. The departure from a standard of government origin or government involvement helped improve industry

confidence and willingness to adopt the standard in addition to reducing the concerns of a trapdoor or preexisting cryptographic shortcuts.

Another summary of the development included robust reduced-round attack information for all of the finalists. The evaluation of attacks against reduced-round implementations provided a fair comparison to assess the margin of security provided by each candidate. The margin of security, or safety margin, is the distance between the number of rounds of operations that could be feasibly broken and the number of rounds proposed for AES. While some candidates favored performance, others favored security at the expense of performance. This imbalance in the candidates made the reduced-round attack feasibility an especially important measure of the margin of security provided by each candidate. The summary concluded that Rijndael had an adequate margin of security with the best overall performance and flexibility (Nechvatal et al., 2001).

Burr (2003) authored the Institute of Electrical and Electronics Engineers (IEEE) review of the AES selection process. He provided the first look at attack feasibility in addition to a comprehensive review of the process and algorithm, including the modes of operation. Citing Moore's law (Moore, 1965), Burr estimated that the 128-bit AES would be as secure as an 80-bit key was in 2003, extending until 2066 (Burr, 2003). The modes of AES, which are common to block ciphers, including DES, were electronic code book (ECB), cipher block chain (CBC), cipher feedback (CFB), and output feedback (OFB). NIST added the counter mode (CTR) for AES which improves upon the security of parallelizable modes. The modes define how the algorithm is used and have their own body of knowledge. The modes are important as applied to all block ciphers, but are a separate topic outside of the primary field of AES cryptography.

The transition from DES to AES was a complex process. Attacks against DES were far into the realm of feasibility before AES was announced and use of DES continued for several years. The continued use of the previous standard serves as evidence of the transition time. The NIST issued DES transition plan in 2005 set a final transition period of two years for all federal agencies to transition to TDEA or AES as the standard allowing DES to be formally withdrawn. Additionally, the IETF security warning in 2006 focused on continued use of DES by the commercial sector (Kelley, 2006). Common reasons for continued use included backward compatibility, performance, and ignorance. The public warning sought to address the ignorance component, while additionally making a strong argument against continued use for the other two reasons. Although, this gap in security during the transition from DES to AES was concerning, NIST addressed the concern better for AES.

NIST continues to maintain awareness and responsibility for AES. Approximately every five years, NIST releases a special publication that contains recommendations and requirements for algorithm and key-length use. NIST conducted its most recent review in 2015. The review found all key lengths of AES acceptable for use until the next assessment (Barker & Roginsky, 2015). Based on the verbiage for phasing out other algorithms, when attacks against a key length are approaching feasibility, NIST will include a recommendation to phase out during the next five-year period. When key lengths or algorithms no longer provide security, NIST marks them as legacy use, indicating nothing new can be encrypted using the key length or algorithm; the proposed retirement candidate may only be used to decrypt legacy data.

Cryptanalysis of the Advanced Encryption Standard

AES has been a central topic of the cryptographic community since its announcement. Cryptanalysis assesses the strength of the algorithm, confirming the margin of security, through "attacking". These cryptographic attacks emulate an adversary with the goal of learning the

secret key or otherwise gaining access to the plaintext contained in the ciphertext. Attacking the cipher is generally the most complicated means of access. Attacking the cipher could be described as gaining access to a locker by guessing the right combination rather than unscrewing a vulnerable external hinge. Vectors outside of the cipher usually present significantly easier means of access. Despite the easier modes of access, researching cryptographic attacks ensures that the component, which is most often assumed secure, remains true to the assumption.

Cryptographic attacks have a few universal components. Requirements for data, compute, and memory are common. Data refers to the size of the ciphertext in either bits or in blocks. Attacks range from requiring a block worth of data to nearly the size of the brute force key space. Data may include specific data or pairs. Some attacks require known- or chosen-plaintext pairs. These pairs involve a block of the ciphertext that corresponds to a known or chosen plaintext. Every attack, including brute force, requires some way to verify the output is as expected as any number of candidate keys may appear to produce an expected result. This effect is more prominent for the smaller the ciphertext.

Compute, also known as workload, is the number of operations required to complete the attack. For cryptographic attacks against AES, the internal rounds vary be key length: 10 rounds for 128-bit, 12 for 192-bit, and 14 for 256-bit keys. These rounds are a component of complexity. Additionally, key setup operations vary by key length. Reduced round implementations are also analyzed as a means to measure the margin of security. Because the rounds may not match the specified key length for these assessments, the key length and the number of rounds are independent factors of the compute component of attacks against AES. Compute is usually measured in the number of full AES operations based on the key size and rounds.

Memory refers to the amount of storage, not necessarily random-access memory (RAM), which an attack requires. Some attacks require very little memory to conduct, just enough to store candidate keys while they are tested. Others use precompute to produce large structures that require significant memory to reduce the final compute requirements. Aside from the common components, data, compute, and memory, AES has the additional components of key length and rounds. However, some categories of attacks against AES have additional, special requirements.

Cryptographic attacks fall into several categories. At the highest level, these attacks either rely on special conditions about the situation or they do not. Attacks that do not rely on special conditions are general purpose and apply to the cipher operating in any condition. Attacks in this category that improve upon the compute of brute force are considered breaks. Attacks that do rely on special conditions are not breaks of the algorithm because they rely on conditions outside of the algorithm. The conditions include some form of physical access, access to the input and output of the cipher, related keys, or implementation nuances. Most attacks require some understanding of the plain text. Without the ability to verify the plaintext, determining the validity of a candidate key is not possible. For example, if a fully random block was encrypted, decrypting with the incorrect keys and the correct key would produce similarly random outputs. However, if the plaintext is known to contain ASCII text, the key validation may simply consist of checking for outputs that contain ASCII characters. These checks are usually parallelizable and are not always included in the compute component.

General Attacks

All traditional, non-quantum encryption is subject to the simpliest of attacks: brute force. While simple conceptually, it is generally the worst case in terms of compute as it forms a starting point for future cryptographic efforts to improve upon. Although far from feasible, Rijndael was subject to brute force from its inception. As seen with DES, the key length is the

mitigating factor for this attack. The key lengths of 128-, 192-, and 256-bit for AES were determined before any algorithms were submitted. Almost 20 years later, even the 128-bit keylength successfully places brute force out of the range of feasibility. Brute force is data agnostic. It does not require large samples or pairs of ciphertext and plaintext; as long as enough is known about the plaintext to test if a decryption attempt was successful with a given key, nothing else is needed besides an immense amount of compute, time, or both.

Table 2

General attacks

Attack	Key(s)	Rounds	Complexity			Year
			Data	Compute	Memory	
Brute Force	128	10	Trivial	2^{128}	Trivial	1998
	192	12		2^{192}		
	256	14		2^{256}		
Biclique	128	10	2^{88}	$2^{126.1}$	2^{8}	2011
	192	12	2^{80}	$2^{189.7}$		
	256	14	2^{40}	$2^{254.4}$		

Note. Derived from Bogdanov, Khovratovich, & Rechberger (2011) and Burr (2003).

Aside from brute force, the only known general attack against AES is biclique cryptanalysis. Even in its worst-case performance it improves on brute force by a factor of three to five times depending on key length (Bogdanov, Khovratovich, & Rechberger, 2011). An algorithm is considered broken if any attack, which does not require prerequisite assumptions, is faster in terms of compute than brute force (Burr, 2003). The biclique cryptanalysis attack is the only known break of AES (Bogdanov et al., 2011). While it is a ground breaking attack, it has generally infeasible data requirements and is not a large enough improvement on compute to be feasible. Future improvements upon the data and compute components of this attack method may

represent a formidable threat to the continued use of AES; however, no improvements have been published to date. Table 2 depicts the general attacks against AES, their key lengths, rounds, complexity, and the year they were introduced to the body of knowledge.

Partial-Implementation Attacks

Attacks against partial implementations help validate the margin of security provided by the algorithm. Internally, AES repeats the substitution, transformations, and mixing steps for a given number of rounds based on the key length. For AES, the rounds are 10, 12, and 14 for the given keys lengths of 128-, 192-, and 256-bit respectively. The security margin is the gap between the highest-round, feasible attack and the rounds required by the algorithm. If the gap is too narrow, a single innovative shortcut could result in a feasible attack, which would render the algorithm obselete. In practice, partial implementations are rare because they lack security and consequently lack approval for use in most situations, but dispite the drawbacks, are sometimes used to implement the internals of stream ciphers (Bouillaguet et al., 2012).

Even before NIST selected the Rijndael algorithm as a finalist for AES, its authors, Joan Daemen and Vincent Rijmen, considered known cryptographic methods and Rijndael's strength against those techniques. The authors asserted that the full implementations for each key length were secure from known cryptographic methods, but partial implementations were expectedly vulnerable. They found feasible attacks for all key lengths when only four or five rounds were used and a generally infeasible attack, requiring 2^{72} operations, with six rounds (Daemen & Rijmen, 1999). The required round lengths were purposefully selected for their balance of performance and acceptable margins of security.

Partial implementation attacks often rely on other assumptions including related keys or the ability to manipulate or knowedge of the plaintext. Table 3 contains notable partial implementation attacks regardless of their other dependencies. The following sections omit these

attacks to prevent overlap. Although the attack relies on related keys, which makes it inpractical, the related-key boomerang attack presented by Biryukov and Khovratovich in 2010 is not only in the feasible range but is also only one round short of a full implementation. A single improvement in the attack technique could pose a feasible, albeit still costly threat to AES use in related key environments.

Table 3

Partial-implementation attacks

Attack	Key(s)	Rounds	Complexity			Year
			Data	Compute	Memory	
Truncated Differential	All	4	2^{9}	2^{9}	Trivial	1999
		5	2^{11}	2^{40}	Trivial	1999
		6	2^{32}	2^{72}	2^{32}	1999
		6	2^{32}	2^{44}	2^{32}	2000
		7	$2^{128} - 2^{119}$	2^{120}		2000
Partial Sums (Chosen-Plaintext)	128	7	$2^{128} - 2^{119}$	2^{120}	2^{64}	2000
	192	8	$2^{128} - 2^{119}$	2^{188}	2^{64}	2000
	256	8	$2^{128} - 2^{119}$	2^{204}	2^{64}	2000
	256	9	2^{85}	2^{226}	2^{32}	2000
Related-Key Chosen-Plaintext	256	9	2^{85}	2^{224}	2^{32}	2000
Impossible Differential	128	7	$2^{115.5}$	2^{119}	2^{109}	2008
	192	7	2^{92}	2^{186}	2^{157}	2004
	256	7	$2^{92.5}$	$2^{250.5}$	2^{157}	2004
Related-Key Differential	256	10	2^{44}	2^{45}	2^{33}	2010
		11	2^{70}	2^{70}	2^{33}	2010
Related-Key Boomerang	256	9	2^{59}	2^{119}	$2^{42.5}$	2010
	256	9	2^{67}	$2^{135.3}$	$2^{42.5}$	2010
	256	13	2^{76}	2^{76}	2^{76}	2010

Note. Derived from Nechvatal, et al. (2000), Phan (2004), Ferguson, et al. (2000), Biryukov & Khovratovich (2010), and Soleimany, Sharifi & Aref (2010).

Key Attacks

Key attacks generally rely relationships between keys but may also refer to attacks that exploit the key schedule. Frequently it is the properties of the key schedule that enable the observation of related keys to result in key recovery. For every criticism of AES, a solution is

presented. Several different modifications have been proposed to strengthen the key schedule, as one of the most criticized components of AES and the foundation for many related key attacks. Early proposals did not always uphold the requirement for hardware implementations, for example, by increasing the key expansion by adding rounds of AES, they made the key schedule stronger and importantly irreversible, but not hardware implementation compatible (Choy, Zhang, Khoo, Henricksen, & Poschmann, 2011). Later proposals improved security at little cost to efficiency, while remaining hardware implementation compatible.

Although many of the partial implementation attacks in Table 3 rely on related keys, these attacks also extend to full implementations found in Table 4. Related-key attacks rely on a number of related keys, an attack requiring the observation of a few keys, like the boomerang attack in Table 4, is much less practical than the attack requiring 2^{35} related keys (RKs). The tradeoff in terms of practicality and the other components of complexity are common throughout the attack types, but especially important here where an attack with the complexity of the related-key differential would be devastating without the requirement for the impractical number of related keys.

Table 4

Key attacks

Attack	Key(s)	Rounds	Complexity				Year
			RKs	Data	Compute	Memory	
Related-Key Boomerang	256	14	4	2^{119}	2^{119}	2^{77}	2009
Related-Key Differential	256	14	2^{35}	2^{96}	2^{96}	2^{65}	2009

Note. Derived from Biryukov & Khovratovich (2009); Biryukov, Khovratovich & Nikolić (2009)

Plaintext Attacks

Plaintext attacks rely on access to known or chosen plaintext in addition to the cipher text. Many other attacks, including brute force, require at least some limited information about the plaintext, encoding for example, to allow trial decryptions to be evaluated. With known plaintext attacks, key recovery relies on many known pairs of plaintext and ciphertext. Chosen plaintext attacks extend this approach by requiring the attacker to be able to encrypt arbitrary plaintext and observe the resulting ciphertext. Although several of the partial implementation attacks in Table 3 require chosen plaintexts, most AES modes are, by design, resistant to this type of attack. Since these attacks were known even prior to attacks becoming feasible against DES, they were appropriately considered during the AES selection process (Baudron et al., 1999; Biham & Shamir, 1993).

Side-Channel Attacks

Side-channel attacks employ unique approachs to defeat cryptosystems. They utilize various forms of leaked information or implementation weaknesses instead of directly attacking the cryptographic algorith (Alghazzawi, Hasan, & Trigui, 2014). The exploited components include physical properties such as timing, power consumption, noise, or radiation. Generally, these attacks are simultaneously the most feasible and the least practical (Zhou & Feng, 2005). The feasibility stems from the avoidance of the cryptosystem and focus on the other available channels. The impracticality is based on the high degree of access to the system that measuring the precise timing, power draw, or electromagnetic radiation would require (Khan & Mahanta, 2014). Although these attacks avoid the cryptosystem, the usage mode of the cryptosystem has an impact on the feasibility of power analysis (Jayasinghe, Ragel, Ambrose, Ignjatovic, & Parameswaran 2014). Table 5 contains several notable side-channel attacks. Unlike the previous attack categories, every type of side-channel attack requires the exposure of drastically different

components of the algorithm. In 2011, Floissac and L'Hyver adapted a differential fault analysis attack to all key sizes which required generating faults in the key expansion process. Although the internal states of the key expansion process would be very difficult to access, this attack only requires a few bytes worth of faults to recover the key. Another approach, collision timing, requires access to the input, output, and clock of AES ASICs to recover the key. Improvements on this method made it capable of defeating fault and power analysis protected hardware implementations (Moradi, Mischke, & Paar, 2013).

Table 5

Side-channel attacks

Attack	Key(s)	Rounds	Access	Complexity			Year
				Data	Compute	Memory	
Differential Power Analysis	128	10	Micro-Second Power Draw	Trivial	Trivial	Trivial	2008
Differential Fault Analysis	All	All	Key Expansion Internal States	Trivial	Trivial	Trivial	2011
Collision Timing	All	All	ASIC I/O, Clock	2^{24}	Trivial	Trivial	2013
Trace Driven Cache	All	All	Cache	Trivial	Trivial	Trivial	2013

Note. Derived from Yu, Xue-cheng, Zheng-lin, & Yi-cheng (2008); Floissac & L'Hyver (2011); Moradi, Mischke & Paar (2013); Zhao, et al. (2013).

Performance

Regardless of attack type, AES performance forms some component of the feasibility of the attack. The simplest attack, brute force, is entirely performance-based. The faster AES operations can be conducted, the faster the attack can be completed. Other attacks require various AES operations, but performance is not only a consideration that relates to attacks. It was a

priority during selection to ensure that AES was usable by the computers of that time. This section includes the performance considerations during the selection process, AES performance since selection, and general performance since selection. AES performance since selection involves the changes to implementations, both hardware and software, that improve upon the initial performance characteristics of the algorithm. Lastly, general performance pertains to the state of computing, including innovations in lithography, parallel computing, and architectures.

Another component of performance is the study and application of performance predictions. This practice involves the identification of system and software factors effecting performance through the use of benchmark and modeling. Even in its early forms, the design of representative workloads involved modeling and tuning, which made external validity difficult (Berry, 1992). The rise of standard benchmark solutions like Linpack and HPL helped address those difficulties. From those standard platforms, results became consistent enough to use the benchmark ratios as performance predictors (Gustafson & Todi, 1998).

Predicting performance relies on the modeling of the process. Queuing networks are a method for modeling steps in a process to predict performance and better design representative workloads. They use nodes to represent components of a process, which makes them especially applicable to the understanding of AES performance (Balsamo, Di Marco, Inverardi, & Simeoni, 2004; Menascé, Almeida, & Dowdy, 2004). Using queuing networks, a brute force attack on AES could be portrayed as a linear network where each node is parallelizable: key expansion, trial decryption, and result verification. Other attacks require more complex models; however, nearly all the attacks have some component of AES operations.

AES Performance Considerations during Selection

The performance of the candidate algorithms was a crucial component of the selection process. Even the announcement requesting candidates mentioned efficiency as a component of

the evaluation criteria (NIST, 1997). At that time, DES was waning and the rising TDEA alternative, which simply increased the key by conducting DES multiple times, was very inefficient (EFF, 1998a; NIST, 1999b). The concern that the next standard be more efficient than DES was evident during the process by the various performance publications which addressed this component of the requirements. Furthermore, as technology continued to become more mobile and miniaturized, NIST appropriately included a variety of microprocessors and other miniature devices (Burr, 2003).

After the first candidate conference, the performance of the leading 15 candidate algorithms was measured on a variety of platforms. The study measured not only encryption but also key setup performance. It provided detailed summaries and comments for each algorithm. Rijndael, the candidate that eventually became AES, had only positive comments boasting great performance across all platforms without any incompatibilities or negatively affected platforms (Schneier et al., 1999). Although performance was not the only design consideration, the algorithms that became the finalists all performed very well in this initial assessment.

When the five finalists emerged, Schneier and Whiting (2000) conducted an additional study focused specifically on the performance characteristics of the remaining candidate algorithms. The study included additional fields, such as comparisons of the minimal secure and maximally insecure variants of each algorithm, to help delineate the differences in performance for each algorithm's round count. The study also included concern that the primary purpose of an encryption standard, security, might not be as high of a priority as performance. Despite the concern, all of the finalists had excellent margins of security and the emphasis on performance would allow industry to better adapt to and implement the new standard. In the end, Rijndael not

only had very low memory requirements but was also twice as fast as the other finalists (Nechvatal, et al., 2000).

AES Performance since Selection

AES performance since its selection refers to performance improvements specifically targeting AES. These performance improvements and optimizations have equal impact on AES for normal cryptographic use as they do cryptanalysis. Despite having impacts relating to both normal use and cryptographic attacks, in the literature, these improvements usually focus on just one component of the performance improvement. This current research draws upon both categories since improvements in one mean improvements in the other. The drivers for the improvements primarily focus on hardware, since the algorithm specifications have not changed.

GPUs are mentioned throughout the literature as powerful platforms for cryptography. In 2007, Manavski found that GPU architecture very capable as an accelerator for AES. Both encryption and decryption saw improvements, which peaked at nearly 20 times faster than the CPU. This study focused on the CUDA architecture found in Nvidia GPUs. Another more recent CUDA study saw an increase in performance over CPU by 87 times (Khan et al., 2014). Further research on GPU-based efficiency and price to performance comparisons followed. Not every study found the same degree of performance improvements, but helped to validate the capability of the platform for AES operations (Shao, Chang, & Zhang, 2010). A notable study in 2014 found that the platform agnostic GPU library OpenCL was also capable of significant improvements (Yuan, He, Gong, & Qiu, 2014). Although all types of GPUs have formidable AES potential, the implementations are still in software, albeit highly parallelized software.

Perhaps the largest performance improvement was the advent of AES hardware instruction sets on the CPU. In 2010, the Intel® AES new instructions (AES-NI) brought hardware implementation to an unprecedented level, increasing performance by more than an

order of magnitude (Gueron, 2010). AMD, and eventually ARM, have since adopted similar AES hardware instructions. Because every system has a CPU, this performance improvement effected a much larger population of platforms. Although this improvement alone made routine use of AES seamless, it does not pose a direct threat independently. However, this improvement combined with a large group of systems or a supercomputer might pose a significant as technology continues to improve. A combination of the massive parallelism of GPUs in the form of application specific integrated circuits (ASICs) and hardware instructions would produce a dangerously powerful platform. In the offensive sense, Biryukov and Großschadl theorized that a GPU-like ASIC would have the throughput of 77.6 Gbit/s (2012).

General Performance since Selection

The general state of computing performance since NIST selected Rijndael as AES in 2001 involves both improvements in the prevalence of technology and improvements in the capability of technology. As mentioned in the introduction, the sheer size of the connected world is the first major factor. Based on stats from the International Telecommunication Union, the World Bank, and the United Nations, nearly of half of the world's population uses the Internet (Internet Live Stats, 2018). Up from less than one percent in 1995, the increase in the last two decades is enormous. While the introduction and assumptions and biases sections focused on the increased need for protecting what has become a critical component of society, the increase also has a performance implication: the amount of available processing power has risen at the same alarming rate as the world has adopted connected technology. This increase is compounded by the actual evolutionary and revolutionary improvements in technology that follow.

The improvements in the capability of technology have two components, evolutionary and revolutionary changes. When AES was selected, Moore's Law was the best approach for estimating the future performance of computing systems. It related the cost per component and

the number of components per integrated circuit with time (Moore, 1965). For decades, performance has closely aligned with this law. However, revolutionary changes in technology have separated performance from sheer number of components, or in most cases, transistors. Production scale, lithography, is an example of evolutionary changes. From 350-nanometer manufacturing scale in the mid-1990s to the 14-nanometers that is mainstream today, this change accounts for significant computing performance capability (Intel, 2018).

Evolutionary improvements in CPU and GPU design scale, lithography, have allowed for higher clock rates and respectively greater compute potential. However, these changes faced a variety of limiting factors, which led to the first revolutionary change, multicore CPUs. Multicore CPUs allowed a departure from the clock rate and instructions being the sole determining factor of a CPU's performance. Other changes in the manufacturing field, including tri-gate transistors, allowed more cores on each socket, reduced heat, and enabled higher clock speeds. All these innovations, which improved upon general performance, also improved AES performance and resultantly enhanced attack feasibility.

Attack Feasibility

Although feasibility is a component of every published attack, the purely attack feasibility literature is limited. The earliest estimation of AES attack feasibility cites Moore's Law and states that the first key length, 128-bit, should endure for around six decades as secure as an 80-bit key was in 2003 (Burr, 2003). In 2003, an 80-bit key was undeniably secure. Asides from the periodic review for usability, transitioning from DES and general AES research seemed to take precedence over AES attack feasibility. Given the lifespan of DES and the significantly improved key lengths of AES, the focus on transition, rather than immediate consideration of AES key lengths was appropriate.

Biryukov and Großschädl (2012) theorized an attack platform capable of completing attacks of the 2^{100} compute complexity in feasible time using GPU-like, special-purpose hardware. Although their work was published in 2012, it referenced 2010 technology. The attack referenced was the partial implementation attack using related keys with $2^{99.5}$ compute complexity (Biryukov & Khovratovich, 2010). Although this attack has assumptions and is based on a partial implementation, an improved method or simply time, may make their trillion US dollar theoretical system much more realistic in the near future.

Another attack feasibility study focused on the massive increase in the number of systems, in particular, mobile devices. It followed a similar framework as Distributed.net used to attack DES: just 10,000 commodity devices defeated DES only 20 years after its selection (Curtin & Dolske, 1998). Marculescu (2014) concluded that a trillion mobile devices could search 2^{75} keys per year. This assessment, places even the shortest AES key length of 128-bit far out of reach. These assessments are expected to continue to indicate that attacks are infeasible for years, but are still important for staying ahead of the necessary transition time. Furthermore, as these assessments indicate a closing margin of security, similar to the early 1990s assessments of DES, moving to the higher key variants of AES will be appropriate.

Future technologies such as quantum computing pose a great threat to encryption technologies, just as the aforementioned innovative technologies that increased the rate of compute beyond the rate that was initially projected. The massive revolutionary change that widespread or large-scale quantum computing will bring to the world will also have a great impact on cryptography. Algorithms that rely on factoring will be greatly reduced in attack complexity by Shor's algorithm (Bernstein et al., 2009). The reduction is so great that these methods will be rendered immediately and completely obsolete. Algorithms like AES do not rely

on factoring. However, Gover's algorithm still provides reduction in complexity, but not enough to immediately defeat the algorithm.

Conceptual Framework

The foundation for this research is the modern cryptographic principle of cryptanalysis. Combined with the lessons learned by the rise and fall of the previous standard, DES, the cryptographic community, led by NIST, selected the Rijndael algorithm as the new encryption standard. Lessons including key length, differential cryptanalysis, and transition time led this research to the gap addressed by this study. Despite attack feasibility literature emerging the in early 1990s, the feasibility of conducting a brute force attack against DES was not adequately tracked, calculated, or projected when it became evident that DES was no longer secure. Although NIST assesses AES periodically for five-year continued use, a practical, potentially real-time approach to determining the attack potential, is absent from the body of knowledge.

As the next generation standard, AES is cryptographically sound. Even before final selection, the candidates underwent extensive cryptographic and performance testing. AES is approaching its initial design time of 20 years, although initial projections of the shortest key length enduring for six decades, the world computing power has undergone massive change since AES took the stage. Changes, including multicore, multithreading, and AES hardware instructions, directly impact the CPU portion of processing power addressed in this study. To prevent these changes from compounding into a similar insecure situation that happened with DES, this research sought to determine the ratio between a widely available performance standard, FLOPS, and the AES operations, to potentially allow projections about the endurance of the standard and attack feasibility to accurately determine when key lengths need to be abandoned or a new standard is required.

Summary of Literature Review

This chapter detailed the literature surrounding the topic of AES attack feasibility. It included the foundation of the modern cryptographic era as the reason for open cryptographic standards and the paramount concept of cryptanalysis as the means of validating the security provided by cryptosystems. It advanced through the previous standard, DES, as the context for the selection of the current standard, AES, and demonstrated the study's concern regarding the potential for attacks to become feasible sooner than initially projected. It outlined the selection process and criteria for selection that eventually led to AES becoming the new standard.

The chapter review delved into the various forms of cryptanalysis conducted throughout the literature to validate the security provided by AES. It preceded to detailed explanations of the various attack types and tables depicting each attack with its associated performance data. It related the fields of attack feasibility and performance, demonstrating the intertwined nature of these fields, and identifying the gap which this research sought to fill. It concluded with the conceptual framework, which connected all of these categories of literature in the body of knowledge to form a foundation for the research questions. Chapter 3 will build upon this foundation to detail the method for the study. The method stems from the conceptual framework to answer the research questions.

CHAPTER THREE

This chapter presents the methods used to conduct the study. It begins by reviewing the research traditions from the body of knowledge that align with the conceptual framework. It continues to provide a review of the research questions and their associated hypotheses. The research design follows, which outlines the nature of the study and the approach used for collection, measurement, and analysis of the data. The research design includes the population and sampling information, in addition to the reasons why the sample was selected. This chapter also contains the detailed reasoning for the development of the collection mechanism and the selection of the survey instruments contained within the collection design. It concludes with specific information about the implementation of the collection design, with full code available in the appendices. The inclusion of fully detailed information and code allows the study to be repeated in the future.

Research Tradition(s)

This study contained research traditions from several fields. Benchmarks and the use of representative workloads are well established in the performance engineering field. The use of these procedures in the cryptography field for attack feasibility estimations is less common. In the case of DES, more detailed studies about attack feasibility and complexity were published as attacks reached the feasible point. Although every cryptographic attack publication briefly mentions feasibility, complexity, or performance, the more accurate techniques and dedicated research are less common. When studies do approach attack feasibility, they generally do so in one of two ways: they use a single or very limited set of hardware configurations, or they use theoretical, application-specific systems.

Considering attack feasibility as a form of validating the security provided by a cryptographic algorithm is a foundational concept of the modern cryptographic era. This concept

stems from Kerchkhoffs' *Cryptographie Militaire*, which was published in 1893 (Kahn, 1996). Kerckhoffs recognized the increased emergence of cryptosystems for military use and that the transformation of communication brought on by the telegraph would require new considerations for cryptography. One of these considerations was the need to verify the security provided by a cryptosystem. Kerckhoffs was justly critical of trends in cryptography at the time as the cryptographic techniques not only lacked validation but had not kept up with the pace of technology. Due to these concerns, Kerckhoffs introduced cryptanalysis in his literary work as a necessity to understand the security of a cryptosystem (Singh, 2000).

Cryptanalysis was among the considerations for DES, the predecessor to AES, when it emerged in the 1970s. The expected feasibility of attacks was assessed and found acceptable at that time. By 1993, however, the concern of the key length and the potential attack complexity reduction of differential cryptanalysis entered the literature (Biham & Shamir, 1993). In 1997, RSA Laboratories launched a cryptographic challenge to assess the security of currently in use encryption algorithms; DES was among them. The first success against DES was a distributed group using brute force that peaked around 14,000 personal computers (Curtin & Dolske, 1998). After 20 years as the workhorse of the cryptography, DES was now insecure; its 56-bit key was its downfall. Although the AES key lengths solved much of the problem with DES, at 128-, 192-, and 256-bit, assessments of key length endurance were part of the initial literature after selection.

Cryptanalysis and attack feasibility were even more present during the selection process for AES from 1997 to 2001. The cryptanalysis included the application of all known methods against each of the candidates, in addition to considerations for partial round implementations and their margin of security from feasible attacks (Ferguson et al., 2000). Attack feasibility considerations occurred throughout the process as well. The multiple, long key lengths were

designed to prevent the failures of DES. The early estimate for the shortest key length of AES, approximately 60 years, was made shortly after the selection (Burr, 2003).

Biryukov and Großschädl considered attack feasibility in 2012 using a theoretical, special-purpose approach based on the lithography and available technology in 2010. They concluded that a one trillion US dollar system built with GPU-like special purpose processors could complete an attack against AES in a year. The results of this current research may allow similar estimation about the number or cost of modern, general-purpose CPUs completing the same attack. Consequently, using this study as a template for a study of GPUs or GPU-like ASICs would expectedly yield closer to feasible results.

The study of performance and the identification of factors effecting performance, including the appropriate tuning, are integral to the performance component of this research. Even early benchmark studies had difficulty with external validity and tuning (Berry, 1992). Using one benchmark to predict the performance of another is a component of this study, which Gustafson and Todi used to rank a variety of benchmarks in 1998. Benchmarks and representative workloads stem from system planning, but have an important role in AES performance prediction and the design of the collection mechanism.

This study aimed to build upon these traditions. The consideration of attack feasibility and the inclusion of known attacks fulfils the component of cryptanalysis by testing the security provided by the AES algorithm. The practicality of the DES challenges approach is present in the benchmarking and use of general-purpose hardware. Despite the infeasibility of attacks, the theoretical component is present in the potential applications of the results. Since distributed and general-purpose hardware-based attacks are expectedly far from feasible, the contribution may

help realign the initial key-length durability estimations with the actual progression of technology in the last 20 years.

Research Questions and Hypotheses

R1: How correlated are the results of traditional benchmarks and AES benchmarks conducted on systems in the sample population?

$H0_0$: No statistically significant correlations exist between traditional benchmarks and AES benchmarks conducted on systems in the sample population.

$H0_A$: Statistically significant correlations exist between traditional benchmarks and AES benchmarks conducted on systems in the sample population.

R2: How do the hardware configurations of systems in the sample population effect the level of correlation between traditional benchmarks and AES benchmarks?

$H1_0$: The AES hardware instructions component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H1_A$: The AES hardware instructions component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H2_0$: The processor type component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H2_A$: The processor type component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H3_0$: The memory component of the hardware configurations of systems in the sample population has no statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

$H3_A$: The memory component of the hardware configurations of systems in the sample population has statistically significant effects on the level of correlation between traditional benchmarks and AES benchmarks.

Research Design

This study was quantitative in nature. It involved gathering of hardware information and benchmarks from a variety of systems using two established instruments from the literature. These instruments were included in a robust, highly-automated collection design to allow for the sampling to include the necessary diversity of configurations to purposefully represent the variety of systems in the population. Since the population was modern systems, human involvement was limited to the role of facilitators. Figure 4 depicts a simplification of the research design, instrumentation, and collection procedure.

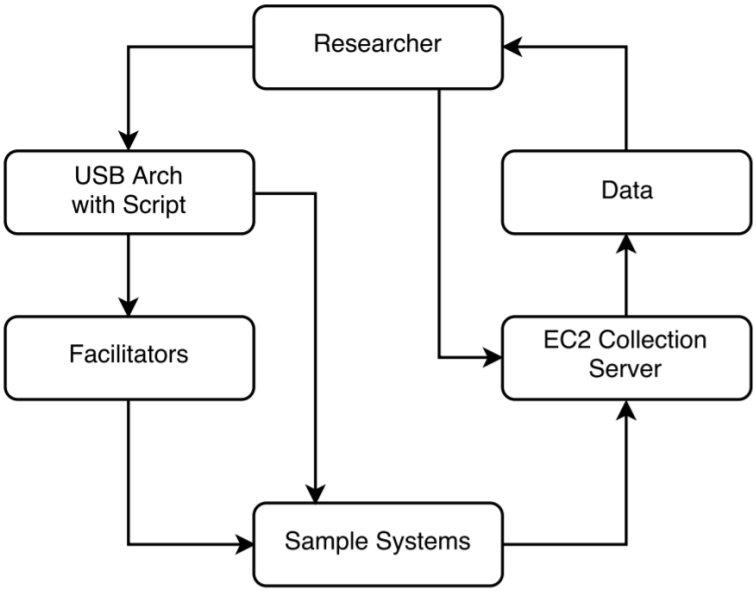

Figure 4. Simple Design by Daniel S. Hawthorne. Copyright 2018

Population and Sample

The population for this research involved the majority of modern computers, which are referred to as *systems* in this study that are able to conduct both floating point operations and AES operations. These systems are found almost everywhere and include the vast majority of laptops, desktops, and servers in use today. The systems are general-purpose and contain central processing units (CPUs) based on the x86_64 instruction set. Both AMD and Intel have used the x86_64 instruction set for central processing units (CPUs) since 2004 (AMD, 2018; Intel, 2018). Although CPUs are found almost everywhere, they are only one component of the global computing power. However, CPUs were the only component of global compute included in this research because the differences in architectures and instruction sets require specially tailored benchmarks and would be best suited for follow-on studies.

The population, and resultantly the sample, deliberately excluded four categories of computing platforms. The first two are ARM-based processor and GPUs. These robust platforms are able to conduct many types of operations, including floating point and AES, but the differences in benchmark techniques would have required separate scripts and introduced a layer of complexity and uncertainty to this study. The last two are field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs). FPGAs perform one specific task but may be reprogrammed to do another. ASICs possess remarkable performance for single tasks as each is designed for only one type of operation. The latter two cannot conduct both floating point operations and AES operations and were therefore excluded. Although they could not conduct both of the necessary operations for this study, these systems excel at performing AES and could pose the greatest threat to its continued use, as they did with DES and as Biryukov and Großschädl in 2012 more recently hypothesized (EFF, 1998b).

The sample was a subset of the population with four criteria for inclusion. The x86_64 instruction set was given as a condition of the population. The ability to boot to a USB device and a wired network connection to the Internet were requirements for inclusion based on the research design. Booting to the same operating system on a USB device reduced the variance of the results by eliminating the components of differences between operating systems, background services, and running programs. The network connection was necessary for the automatic submission of the results. To simplify the task of the facilitators, the collection required a wired network connection, as the task of establishing wireless connectivity without a desktop environment was a daunting for many users.

Sampling Procedure

The sampling procedure used by this study was unique. Unlike many other forms of research, the subjects of this study were computer systems. The components of interest in the study were the configuration of the hardware and performance of those systems. The hardware configurations of the same make and model generally have very little variability in performance. Aside from a few exceptions, which the study accounted for in the reliability section, computer systems in general have negligible differences in performance over time. The research questions and non-experimental approach also lent to the uniqueness of the sampling procedure. The study sought to characterize the performance of systems across a variety of hardware configurations. The study did not attempt to make inferences about the distribution of hardware configurations for systems in the entire population. These initial conditions ruled out random sampling or the need to try to collect representative numbers of hardware configurations. Instead, purposive sampling was used to collect a variety of hardware configurations.

This study utilized purposive sampling. Purposive sampling is a nonprobability sampling method where subjects are selected because they are typical or diverse (Vogt, 2007). For this

study, the exact sample size for each research question was determined during the data collection process. The sampling requirements for R1 and R2 were notably different. R1 is straight forward; it required a number of results from discrete systems. It did not require a set variety of systems, as long as the collected data was from discrete systems, in other words, not from the same computer. Separate systems of the same make, model, and configuration were still a valid inclusion as the results reinforced the reliability of the collection instruments and helped identify anomalies. The sampling requirements for R2 were more complex.

The sampling for R2 required a variety of hardware configurations. The variations were the descriptive independent variables indicated by H1, H2, and H3: AES instructions, processor, and memory respectively. It was possible that a determination on H1 might not be possible, given the prevalence of AES hardware instructions, which Intel introduced in 2010, but are now found on most CPUs (Gueron, 2010). However, if a determination was possible, it was expected that the presence or absence of AES instructions would have a statistically significant impact on the results. The target for completion of the collection phase was reaching an adequate variety of system configurations to make determinations on H2 and H3. The instrumentation and collection design enabled simple expansion of the facilitator audience as necessary to reach the target sample variety.

Instrumentation

The instrumentation and collection design expand the details of the simple design, included in Figure 4. This section describes in detail each of the elements depicted in Figure 5. At the core, the instruments were the *CryptSetup* and high performance Linpack (HPL) benchmark utilities, which were installed on the ArchISO Linux and executed by the collection script. However, the instrumentation and collection design were a more detailed process, which included the distribution, delivery, execution, and results collection of those benchmark

instruments. This section was organized in order of execution through a single repetition of the instrumentation and collection design, although, the development order was the opposite. The instrumentation and collection design process consisted of three scripts, two configuration files, and several other tasks. All of these elements are included in Figure 5, explained in this section, and the full code included in the appendices.

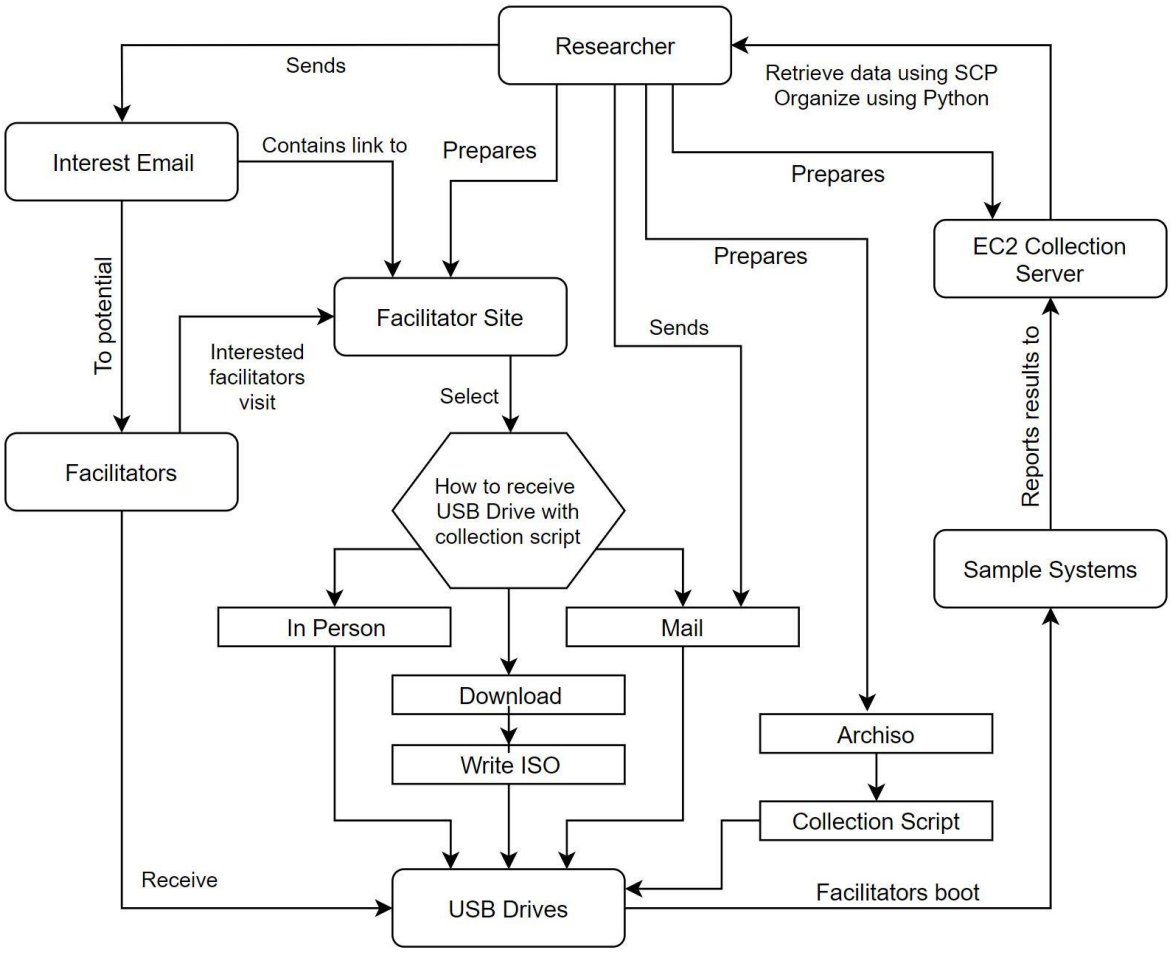

Figure 5. Instrumentation and Collection Design

The process began with the solicitation of interest from potential facilitators. Those that chose to participate received a prepared USB drive to use as a boot device for their system or systems. The drive contained a specifically remastered ArchISO Linux and the collection script.

Once booted, the script ran the aforementioned benchmarks, gathered system information, and submited the results to the collection server. The collection server contained the second script, which wrote those results to a file. Finally, a third script was combined with *secure copy* (SCP) to retrieve the results and organize the data.

Interest Email, Facilitator Site, and Facilitators

Upon completion of the Intuitional Review Board and receipt of approval to begin collection, the process began with the distribution of the facilitator interest email to professional colleagues and personal acquaintances who expressed interest in the research. The possibility for expansion of the audience remained until the completion of the preliminary data analysis. If necessary, the expansion would have begun by including computer industry contacts and posting the contents of the research interest email on various security and research forums. The email included a brief synopsis of the research and a link to the facilitator site where interested recipients could find more information and sign up to help facilitate.

The facilitator site (Appendix E) was a Google survey site that provided additional information about the research and collected some basic information from the facilitators. The upfront information included the goal of projecting attack feasibility, the benchmark pairs to be collected, and the limit of the collection – not accessing any other storage devices or gathering personal information. Following the initial information, the site included a link to the source code of the collection script to allow every facilitator to review, understand, and audit the design to their liking. The specific requirements of an x86_64 processor, the ability to boot to USB, and a wired Internet collection were included and explained to help ensure facilitators had systems that met the criteria for testing.

Following the above information, the facilitator site collected the name, email, and method to receive the USB drive from interested facilitators. Further fields including an option to

be recognized in the publication, phone number, and comments were included but completely optional. Additionally, if the facilitator elected to receive the USB drive by mail, the facilitator had to include a physical mailing address. If the facilitator chose the more advanced option to download the ISO, the final section contained additional instructions for writing the ISO file to a USB drive and an optional device ID field for facilitators that wanted to receive recognition. As Google surveys were submitted, the collection media was prepared and shipped or delivered based on the chosen method.

Live Linux, Collection Script, and USB Drives

Booting facilitator systems to a common operating system eliminated the variability of the operating system, background processes, and services. Arch Linux, a stable and lightweight distribution with a robust user repository containing the tools needed for this research was an appropriate foundation. As one of the lightest distributions, it had nothing extra to slow down the benchmarks while simultaneously leveling the operating system tasks for every sample system in the study. The initial configuration of Arch Linux was a persistent USB installation of the operating system. However, several compatibility issues arose when testing on platforms with differences, such as CPU type, from the platform upon which the persistent USB installation was built. The ArchISO live USB variant of Arch Linux was free of the aforementioned compatibility issues and served as a consistent platform across a variety of hardware types. Although remastering ArchISO to include the packages for this study was much more involved than simply installing them on a persistent install, the read only file system of the resulting remastered ISO helped reduce the chance of facilitator introduced errors.

Manjaro Linux, a user-friendly Arch-based distribution, was used as a development platform and for the initial testing environment for the benchmark script. As an Arch-based Linux distribution, Manjaro's environmental similarities with ArchISO and robust toolset made it

a sound choice for not only development but the remastering process of ArchISO. Remastering ArchISO, which typically contains the Arch installer and other simple packages for use as a live Linux implementation, allowed for the inclusion of the collection script, benchmark instruments, and all the required dependencies in the read only ArchISO. In addition to the development platform, three other systems were used as testing platforms to reduce the possibility of compatibility issues. Table 6 depicts the five test platforms in total, including a virtual machine, which were used for developing the benchmark script to ensure that the scripts functioned as expected in a variety of environments.

Table 6

Testing platforms

	CPU				
Type	Make/Model	Cores	Threads	Memory	AES NI
Laptop	Intel i5-4200M	2	4	4 GB	True
Virtual Machine	Intel i5-4200M*	1*	1*	1 GB	True
Desktop	AMD A8-9600	4	4	4 GB	True
Desktop	Intel G4400	2	2	4 GB	True
Desktop	Intel 6700K	4	8	16 GB	True

Note. Derived from AMD product resrouce center and Intel product specifications (Intel Corporation, 2018; Advanced Micro Devices, Inc., 2018). * Difference from published values due to virtualization.

Appendix C contains every detail of the final configuration and remastering process for the ArchISO. The process began with obtaining and mounting the Arch Linux ISO, which is typically burned to a disc or written to USB as bootable media. Once mounted, the contents of the ISO were copied and unpacked with *unsquashfs* so they were mutable for remastering. The script was copied from the host to the ArchISO file system followed by entering that file system as root using *arch-chroot*. The first task in the *chroot* environment was to change the script

permissions to include executable. The next series of steps comprised the installation of the dependencies for the script, most importantly *dmidecode* and *hpl*. The *dmidecode* utility is the non-benchmark component of the script, which allows for the collection of system information.

After the completion of the other *chroot* steps, which are detailed in Appendix C, the final tasks involved repackaging the file system and preparing it to be written to USB media. First, the new *squashfs* file system was created using *mksquashfs* followed by calculating the *sha512sum* of the file system, which is required for the generation of a bootable ISO. The volume identifier must also be an exact match, so the *isoinfo* was used to extract it from the original ArchISO file. With those preparatory steps complete, the *xorriso* tool was used to generate the USB-ready, bootable ISO. The ISOs were first tested on the virtual environment in Table 6, prior to being written to USB using *dd*. The order of development for the platform and script were reverse, since moving the script to the platform was the first platform step.

The data collection script served as a wrapper for the benchmark collection instruments high performance Linpack (HPL) and Cryptsetup. As both benchmarks run in memory, the system component of storage had no impact on the performance results. HPL measures performance in terms of floating-point operations per second (FLOPS). It is capable of running on systems ranging from single core, single thread to the largest supercomputers. Its prevalence in the performance literature made it an ideal choice. The benchmark results expectedly differed from the Intel optimized Linpack, as HPL is both double precision and CPU manufacturer agnostic. Cryptsetup is a standard Linux utility used for file system encryption. It included a benchmark very similar to the TrueCrypt benchmarks which was more popular in the literature until it was abruptly discontinued. The collection script also contained the system information collection commands and a web connection component, which automatically submitted the

results upon completion. As previously stated, the script gathered no personal data. Table 7 describes each field, including the system details and both benchmarks along with their terms or units.

Table 7

Data fields.

Field	Description	Units/Format
Date_time	The date and time of the collection	Y/m/d H:i:s
Uuid	Universally Unique ID	N/A
System_hash	PSK hashed with the Uuid	N/A
Device_ID	The serial number of the USB drive	N/A
CPU_model	The CPU model name given by lscpu	N/A
CPU_Sig	Stepping, model, family	N/A
CPU_arch	Architecture, sockets, cores per socket, threads per core	N/A
Memory	Total, used, free	MB
AES_Instructions	AES NI present and enabled	Boolean; present/not
AES_Bench	The AES benchmark result	MB/s
FLOPS_Bench	The FLOPS benchmark result	Giga-FLOPS

Appendix D contains the code, with detailed comments for replication or improvements upon this study. The script executes automatically upon boot and begins by messaging to the user and the construction and assignment of a few variables. It continues to test the Internet connection and connection to the collection server; wired DHCP connection is required and mentioned in the limitations and in the instructions to potential facilitators. To reduce the chance of incorrect results, the script conducted these tests without using ping, in case the network connection does not allow ICMP packets. If connectivity fails, the script stops as an automated

collection, since is not possible without connectivity. In the case of failure, the facilitators would be prompted to check the connection and try again.

Following the progression of the execution of the collection script, the collection of system information occurs first. The USB device identification is collected first using *lsblk* and the known volume identification from the remastering process. No other storage devices are accessed or identified. The system's universally unique identification (UUID) is gathered next. The UUID is used to delineate between duplicate configurations and the exact same system. It also serves as a key part for validation of the results, to ensure the results actually arrive from the collection script and not from another source.

The collection of the CPU model name, signature components, and architecture information follow the UUID. The CPU model name is the descriptive independent variable for H2. The stepping, model, and family comprise the signature components; the architecture, sockets, cores per socket, and threads per core comprise the architecture information. These fields are used to confirm the results are conducted on physical, non-virtual systems. A virtual platform would produce inconsistent results for the CPU model name. In the next fields, the script collects memory and AES instructions, whichare the descriptive independent variables for H3 and H1 respectively.

With the system information fields complete, the script continues to run the benchmarks starting with AES. The CryptSetup AES benchmark is conducted 10 times to produce 10 sample results, of which the average is reported. The repetition ensures systems produce similar results when retested. Too low a number, such as one, can have inconsistent results since, even on the very light weight distro, other system-level processes and events may affect the results.

The FLOPS benchmark using HPL is not as straight forward. HPL has numerous possible parameters and methods to optimize operation speed (Chen, 2011). For this study, the majority of the parameters were static to keep results as consistent as possible within the sample population. The problem size, however, varied from system to system as HPL documentation provides an equation based on free memory for problem size. Since the script already determined the free memory as part of the system information fields, it uses that value here. Additionally, HPL requires grid information, which is most applicable to many CPU systems and supercomputers; however, the script uses the CPU sockets, cores, and threads per core to form a simple grid size. The script uses *mpirun* to thread and execute the HPL benchmark. Once complete, the script parses the output file to obtain the results.

The last block of the script reports the results to the collection server. It concatenates each of the results together into a single string and formats the string for *curl*. The script then uses *curl* to submit the results to the collection server. As the script finalizes, it reports to the user the success or failure of the submission and offers to shut down the system.

Server Configuration, Collection Script, and Data Organization

From the submission of the results to the preparation of the data, the collection server is the center component of the research design. The detailed configuration is included in Appendix A for replication purposes. The collection server is an Amazon web services (AWS) elastic compute cloud (EC2) running Linux Amazon machine image (AMI). It is a “micro”-type instance with one virtual CPU, one GB memory, and eight GB storage. Although these specifications may seem limited, they are more than sufficient for the collection server; the free space could fit close to one million results. The configuration of the collection server is straight forward. It begins with installing and enabling the *httpd* and *php* services. The configuration continues with adding a user and giving permissions to the user to allow the results to be written

to a data folder. It concludes with the creating of the folder and the transfer of ownership to the web service user.

The collection script on the web server presents a very simple web form, to which the last section of the benchmark script uses *curl* to submit the results. The PHP script and associated HTML are included in Appendix B. The HTML is a simple form with a post method on submit that invokes the PHP portion. The PHP checks the system hash, which is the hash of the simple pre-shared key and the system's UUID. If the submitted system hash matches the server calculated system hash, the contents of the form fields, which are the benchmark script results and system specifications, are written to a text file in the data folder. Duplicate reports from the same system are appended to the previous results file for that system. The PHP posts a success message, which the benchmark script processes to make its announcement to the facilitator.

The data compilation script parses the files on the collection server and compiles the data into a useable spreadsheet format. Appendix F contains the Python code used for this script. The process begins by localizing a copy of the server data using secure copy (SCP). Once local, the script compiles each result into a row on a spreadsheet.

Validity

The validity of the research had three components: the validity of the instruments, the method used to validate the instruments, and validations to be applied to the data post collection. This section includes the validity considerations for the instruments and a description of the validations, which followed data collection. The instruments were both benchmark utilities, high performance Linpack (HPL) FLOPS benchmark and the Cryptsetup AES benchmark. The rest of the collection methodology, including the web survey, the collection script, and the collection web server, did not constitute instruments; rather they were a cohesive means of distributing the

instruments, running the benchmarks, and collecting the results. Although, they are not instruments, their validity was also ensured.

The validity of the non-instrument components of the research design was straight forward. During the development process, additional outputs from the collection script provided a verbose step by step log of the process. The results obtained from the server were verified as the same as those displayed by the collection script. Additionally, each of the commands were manually tested and compared with the results from the collection server to ensure the entirety of the benchmark script, collection server, and retrieval process was functioning as intended and valid.

The nature of benchmark instruments, which by design measure system performance, provides a foundation of validity. In contrast to research involving humans, system performance is mechanical. As many other FLOPS benchmarks exist, with a wide range of results on a single system, the external validity of HPL is largely limited to other HPL-based FLOPS results. Although other FLOPS benchmark projects, including SETI@home, have very large sample populations, approaching 150,000 in 2018 (University of California, 2018), the HPL-based results from this research had little in common with those results. The project results included the CPU makes, models, and FLOPS; however, the results were based on the Whetstone benchmark, which differs significantly from HPL. Furthermore, manufacturers also use different, highly-optimizes FLOPS benchmarks to measure their products performance in addition to publishing theoretical maximum performance numbers. The choice of HPL related to the attack feasibility component of this study. Although criticized as not being the most representative of modern workloads, HPL is the most commonly used benchmark for high performance systems like large

clusters and super computers, where the greatest capability to complete cryptographic attacks is present (Heroux & Dongarra, 2013).

The post-collection validations sought to identify the cause of outlier results and eliminate experimental errors. Although single-socket HPL results were not widely available, the prevalence and availability of FLOPS as a measurement of performance provided some external validity. If needed, external results could have been leveraged to help identify and eliminate inconsistent or erroneous results. Expected causes included temperature throttling and collection from within virtual environments; however, unexpected causes would also need to be investigated during this process.

Causes, such as temperature throttling, were a limitation that is partially mitigated by external validation. Temperature throttling was also partially mitigated by the instructions to facilitators as an identified limitation, as noted in Chapter 1. While the external performance data provided by the manufacturers might expectedly be best-case or over optimized, data from sources like SETI@home may be considered in conjunction with manufacturer data. Variance was expected between the results of the benchmark used by the collection instrument for this research, the manufacturer data, and other external sources, but the variance across similar systems in the sample was expected to be limited. All outliers were fully investigated, when comparing the level of variance between those data points.

Reliability

The reliability of the collection instruments, HPL and CryptSetup AES benchmark, was already largely determined through their prominence in research; however, the implementation of the benchmark script further ensured that repeated results were the same. HPL is recognized worldwide as a standard benchmark, which by definition produces repeatedly reliable results. Arguments against the reliability of HPL are sometimes made based on the highly tunable nature

of the parameters for the benchmark. To eliminate the variables of configuration over-optimization, standard, single-socket parameters were selected. The fields that do require customization, problem size and grid, were gathered from each system as the HPL documentation requires those parameters to be based on system memory and threads. The results from CryptSetup vary more than HPL. To limit the variance, the benchmark script took 10 samples and reported the average. The instrumentation section contains the details of this process.

The research design included the collection of extra system fields, which are not descriptive variables, but instead were used to determine atypical configurations, which were likely to produce invalid results. For example, a modern CPU with the AES instruction set disabled, hyperthreading/hyper transport disabled, or running the script in a virtual environment would expectedly produce results that differ from other CPUs of the same make and model. Situations exist where these configurations might be warranted, but they are by no means standard and would produce results outside of the norm and outside of the scope of this study. Table 7 contains in the instrumentation sections each of the fields and their descriptions.

The results themselves may demonstrate reliability in two ways. The first is repeat collection from the same system. During the testing and development of the benchmark and collection scripts, many duplicate results were amassed. Results from the same iteration of the design process were nearly identical. Duplicate results could also be present during the data collection portion of this research. Additionally, systems with identical hardware configurations could further the reliability or help to identify anomalous results. For example, if one of three different systems with the same model CPU, amount of memory, and presence of AES instructions has significantly different performance; the unexpected difference would prompt

further investigation as a potential experimental error. Conversely, if all three of those systems had similar results, albeit some degree of difference was expected, the similarities would reinforce the results. Since many system configurations are commonplace, the latter, reinforcing example was expected to occur.

Data Collection

Once approved to begin data collection, a few final preparatory steps were conducted before beginning the instrumentation and collection process. The preparation involved the facilitator site, collection server, and final verifications of the construct. Both the facilitator site and collection server were brought online during development and testing phase; however, if they were not, they would been brought online at this time. The collection server contained a variety of test data results, artifacts of development process. Those testing and development results were archived to ensure they remained separate from the results to be included in the study. As a form of final verification, the entire data collection process was conducted following the exact steps that the facilitators would use on locally available systems, including the testing platforms from Table 6, but excluding the virtual machine.

The process depicted in Figure 5 began with the distribution of the interest email containing the link to the facilitator site. The initial audience included professional colleagues and personal acquaintances who expressed interest in the research; however, the design of the study would have allowed for expansion as necessary during the data analysis. Once the interest email was sent, the duplication and verification of USB drives with the modified ArchISO and the collection script began to meet the anticipated facilitator audience. As the requests were received for in person and by mail receipt of the USB drives, the requests were promptly fulfilled. The download method was immediate and did not require any action or intervention for

the distribution of the collection device. In all cases of USB drive receipt, assistance was made available for facilitators to overcome any problems they encountered.

The facilitators received the USB drives with instructions describing booting to a Linux USB device. As the benchmarks, system data collection, and upload to the collection server were all automated in the instrumentation and collection design, the researcher remained in the support role, being available to answer questions. The researcher addressed any issues with the instrumentation and collection design but only modified the design if it precluded an entire category of systems. At regular intervals, the researcher connected to the collection server to check for results, retrieved the results using SCP, and organized the results into a spreadsheet using the Python script prior to including those results in the data analysis.

Data Analysis

The data analysis process began with the data collection. As the collection server received results, the results were periodically retrieved and validated followed by initial tests aligned with H0, H2, and H3. As sample size was not set at a specific number of systems, the results of the periodic tests helped to determine length of the data collection. H0 was tested using linear regression with the HPL FLOPS result as the independent variable and the CryptSetup AES benchmark result as the dependent variable. The study looked for significant two-tailed p-values of less than 0.001 for the presence of a relationship and a correlation of determination above 0.9 for that relationship to be considered strong. If a strong relationship was present, further tests included a predicted correlation of determination test to determine the viability of using FLOPS as a predictor of AES performance.

The initial tests for H1, H2, and H3 were conducted using multiple regression to determine the developing effects of the AES instructions, CPU, and memory. The same confidence levels were used as H0 with the aforementioned components as descriptive,

independent variables. H1 was expected to be accepted since the 2010 introduction of AES instructions greatly improved AES performance (Gueron, 2010). However, it was possible that the null would not be rejected due to all CPUs in the sample having AES instructions present. The initial tests ended with the collection and were repeated with the final set of results after the validation process.

Ethical Considerations

The study and the associated data collection did not involve participants in the traditional sense. The very limited human participation did not involve humans as the subjects of the research. Rather, the human involvement in the study was in the role of facilitation. Although the concerns addressed in the Belmont Report are primarily applicable for research with human subjects, the components of informed consent still have application to this study. Participating individuals and organizations received detailed information about every component of the facilitation process. This information contained many of the same core components from the Belmont Report, including information, comprehension, and voluntariness (U.S. Department of Health & Human Services, 1979).

Additional considerations for privacy and security were made. No personal information was gathered and local storage was never accessed by the benchmark collection script. The participating individuals and organizations had access to the source code of the benchmark script and configuration information of the Arch ISO Linux. The only means of tracking the number of systems surveys completed by an individual or organization was through the automatic submission of the USB drive serial number as part of the results submission.

The USB drive serial numbers were recorded with the associated individual or organization as an additional means of verification but also to allow for recognition. If the individual or organization desired recognition for their contribution to the study, they were able

to receive recognition since their system surveys contained the serial number of the USB drive that was sent to them. Individuals and organizations participating in the collection process also received information pertaining to the USB boot process. The information included instructions on how to boot to the USB media, a disclaimer about changing BIOS settings, and contact information in case of emergency. While one-time boot menus and booting to USB media does not pose a direct risk, user unfamiliarity with the BIOS or boot process could have introduced risk which was reduced through clarity of the documentation and availability of assistance during the data collection process, in addition to requesting that the test not be run on any operational or critical systems.

Summary of Chapter Three

This chapter covered method and supporting information. It tied in the research traditions found in the literature with the conceptual framework. It restated the research questions and hypotheses to ensure stand-alone value. This chapter used the research design as a blueprint for the sample selection criteria, process, and size. It selected the statistic test that was used to answer the research questions. It also detailed the collection mechanism, covered data analysis, and addressed the unique ethical considerations of the study. This chapter included enough details throughout for the study to be replicated in its entirety.

CHAPTER FOUR

This chapter covers the execution of the collection process from the receipt of Institutional Review Board approval through the analysis of the resulting data. It begins with discrete descriptive statistics pertaining to the facilitators. The limitations and validating controls precede the descriptive statistics and observations on each of the data points detailed in Chapter 3. It continues with the presentation of the results and the details of the statistics aligned with each of the research questions and hypotheses. This chapter concludes with a presentation of the findings and a discussion of the implications for the field.

Descriptive Statistics

The facilitator solicitation process began as soon as the Institutional Review Board approved the research. During the one-month collection period, 20 facilitators completed the Google Form in Appendix E acknowledging their role as facilitators and agreeing to participate. Of the 20, 6 had their collection devices shipped, 3 downloaded the ISO preparing their own USB devices, and 11 picked up the package locally. Two facilitators encountered issues with systems that had secure boot enabled or complicated BIOS menus and did not provide results. In total, 14 of the facilitators provided at least one result.

The one-month collection period yielded 44 results. Two of the results appeared to fall into the previously identified temperature limitation. The temperature limitation impacts the results when throttling occurs due to overheating. Result number 28 and result number 38 appeared to have been affected by temperature throttling. The internal control for the limitation, similar or identical systems, as computers, maintaining similar performance on identical tasks, helped eliminate one of these results. Result number 28 had identical specifics as two other samples. The similarity in the other results and lack of similarity with the overheating results

lead to the discarding of result 28. Table 8 depicts these similarities and differences given systems with the specifications: Intel Core i7-4940MX, 16GB RAM, and AES instructions.

Table 8

Temperature throttled result

Number	Status	AES	GFLOP	Ratio
24	Normal	2485.12	12.24	203.0327
28	Throttled	848.54	4.976	170.5265
29	Normal	2487.65	12.1	205.5909

Despite temperature throttling, the ratio was not as affected as it might have been. The throttling appeared to have a similar effect on both the benchmarks. The second temperature throttled result lead to the discovery of an unanticipated limitation, which was introduced by the sequential nature of the benchmarks in the collection script. The script runs the AES benchmark followed by the FLOP benchmark. Despite the very limited duration of the collection script, requiring less than five minutes from power on to shut down on the test platforms, throttling occurred during or more heavily during the second of the two benchmarks. The partial throttling appears to have skewed result 38. Unfortunately, this result did not have identically configured systems in the sample to compare it to. Since the results of AES appear normal for that generation of processor, yet the FLOP benchmark is on par with the CPUs five years older, this result was also not included. Both the throttled results were on mobile scale processors, which are known to throttle more frequently and aggressively than desktops.

Several examples in the data helped to reinforce the nature of computer systems, maintaining the same performance, as a valid assumption. One example was the identical desktop configurations, which produced results 14 and 15. The configurations were identical –

Intel i5-6500, 8GB RAM, and AES instructions – but the desktops were discrete, different systems. The results in Table 9 demonstrate the minimal, 0.16% difference between these identically configured systems.

Table 9

Identical system configuration results

Number	AES	GFLOP	Ratio
14	2720.82	11.63	233.9484
15	2722.88	11.62	234.327

The largest variations in identically configured systems were found on the Intel Core i7-6820HQ, 16GB RAM, and AES instructions: True. Table 10 depicts the differences.

Table 10

Identical system configuration largest variations

Number	AES	GFLOP	Ratio
5	2699.43	11.18	241.4517
42	2624	11.4	230.1754
43	2703.61	10.32	261.9777

The benchmarks were also run twice on single systems – results 9 and 30. The system was tested by separate facilitators working on the same group of systems. The configuration was Intel Core i7-6820HK, 32GB RAM, and AES instructions: True. Table 11 depicts the again minimal 0.14% difference.

Table 11

Same system results

Number	AES	GFLOP	Ratio

9	2705.37	12.13	223.0313
30	2703.86	12.14	222.7232

The categories, which relate to R2, have rich descriptive statistics. The first category aligns with H1, which considered the effects of AES instructions on the level of correlation. Prior to collection, it was unknown whether any systems without AES instructions would be present in the results. Since 2011, AES instructions have been included in most x86_64 CPUs (Gueron, 2010). Despite the prevalence of AES instructions on CPUs, systems without AES instructions were represented in the sample as depicted in Table 12. Interestingly, one of the systems without AES instructions, result 6, had a CPU, the Intel Core i5-3210M, which per specification had AES instructions available. This difference may have indicated that the instructions were either turned off in the BIOS or was due to a variant of the CPU without instructions.

Table 12

AES instructions

Systems without AES instructions	Systems with AES instructions
5	38

H2 related to the processor. Variations between manufacturer and scale, such as AMD versus Intel and mobile versus desktop, were expected and present in the sample. Although AMD was underrepresented, the results were not noticeably different than Intel, but this claim was further investigated as part of the statistic test aligned with R2. Table 13 depicts the distribution between CPU manufacturers in the sample.

Table 13

CPU manufacturer

AMD	Intel
4	39

In terms of scale, both general populations, mobile and desktop, were represented in the sample data as depicted in Table 14.

Table 14

CPU scale

Desktop	Mobile
19	24

H3 relateds to the amount of memory present in the system. A variety of amounts were present. The 6 GB amount was underrepresented, but is also uncommon as memory is manufactured in powers of two. Table 15 depicts the memory amounts.

Table 15

Memory amounts

4 GB	6 GB	8 GB	16 GB	32 GB
5	2	12	19	5

Presentation of the Data

As the results arrived, they were compiled using the data compilation script described in Chapter 3 and found in Appendix G. Appendix H contains the resulting, raw data. The date time field was only included to assist with validations and was not included after this step. Fields including the system hash, UUID, device ID and CPU signature were used to validate the data,

find results from the same systems, and confirm that no results originated from virtual machines. The previous section covered the observations and validation related to those fields. Since none of the results originated from virtual machines and all of the device IDs were valid, these fields were not included in subsequent tables as they lack statistical relevance. Appendix I contains the remaining data, including the numeric AES and GFLOP data, in addition to the data used to form categories for R2. Figure 6 depicts the distribution of the numeric portion in the results.

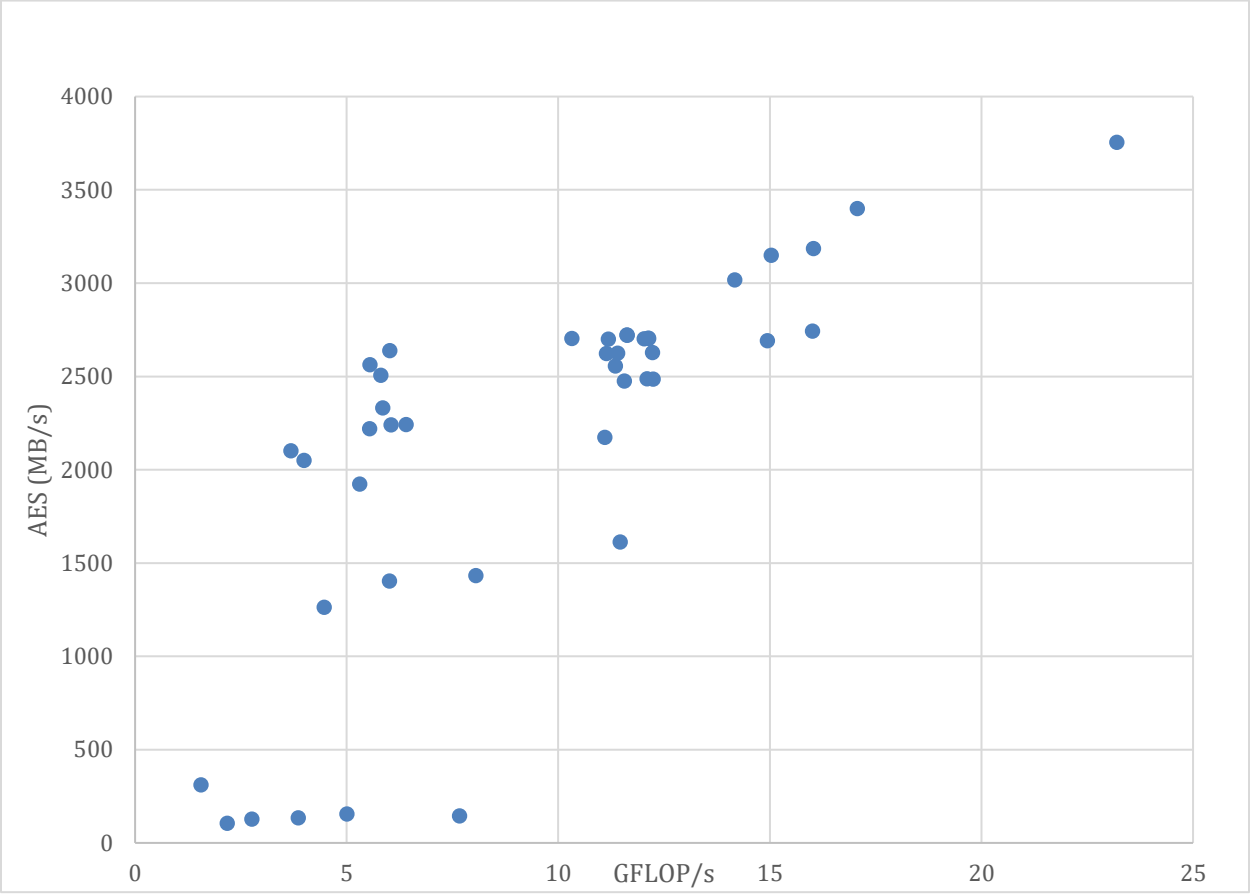

Figure 6. Results

The fields AES instructions, CPU, and memory are not included in the numeric portion but are required to answer R2. Appendix J contains the encoded results, where the CPU make and model are replaced with encoded values for manufacturer and scale. The AES instruction encoding is 0 for false and 1 for true. CPU is encoded 0 for AMD and 1 for Intel. The scale is 0

for mobile and 1 for desktop. Finally, memory is in order of the amount of memory present where 0 represents 4 GB, 1: 6GB, 2: 8GB, 3: 16 GB, and 4: 32 GB.

Presentation and Discussion of Findings

The first test, linear regression, examined the relationship between the traditional floating-point benchmarks and the AES benchmarks for the entire sample. Figure 7 depicts the results with a fit line. The data forms a recognizable pattern. The two-tailed p-value of significantly less than 0.001 (1.194e-8) indicates a statistically significant relationship between the AES and FLOP benchmarks. However, the correlation coefficient of 0.7486 and the resulting coefficient of determination of 0.5604 indicate that the relationship is weak. Given the inclusion of all categories of systems, especially those with and without AES instructions, the relationship is expectedly weak. The resulting equation in Figure 7 is valid for the sample but lacks precision.

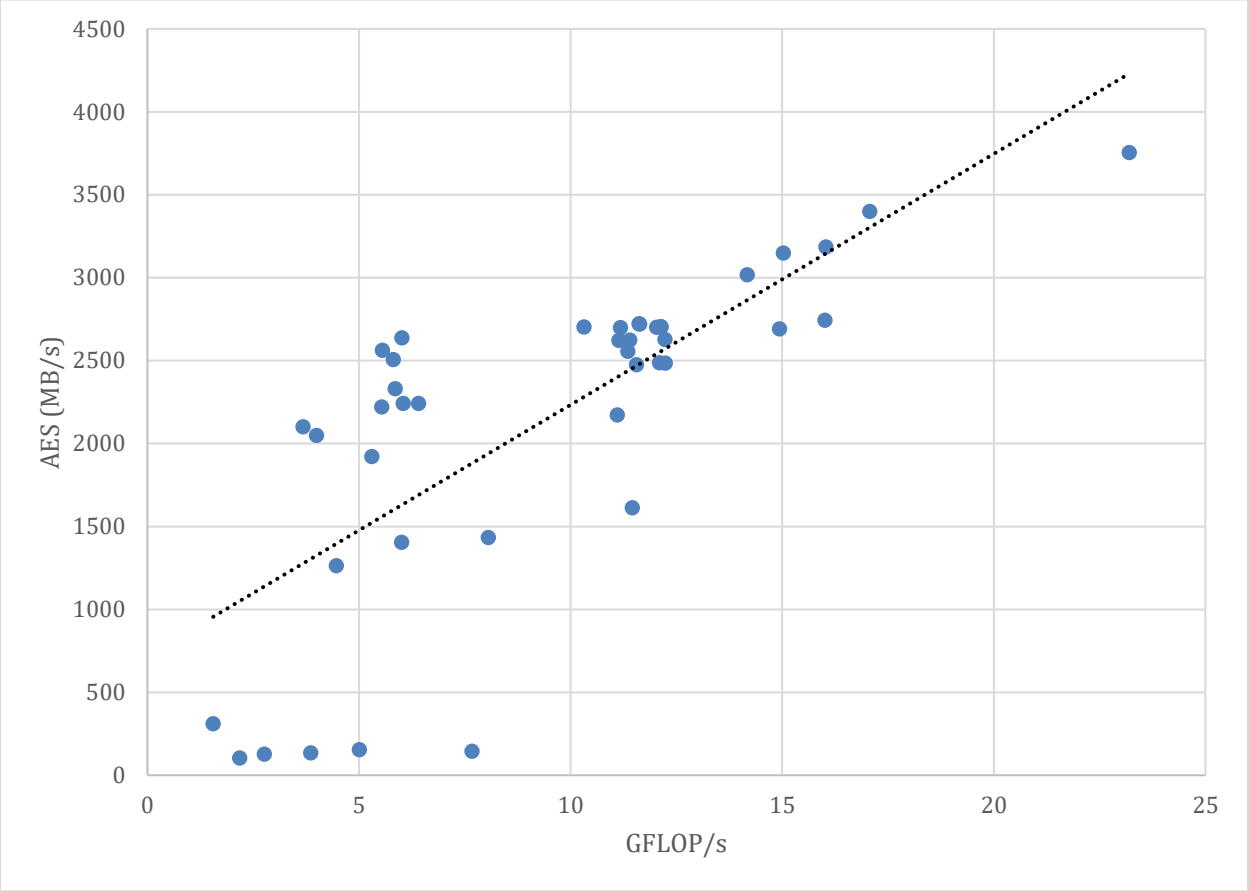

Figure 7. Results with fit line

Notably, five of the six lowest outliers, those less than 500 MB/s AES, did not have AES instructions.

The strong p-value leads to the rejection of the null hypothesis for $H0_0$, where no statistically significant relationship is present, and the acceptance of the alternate $H0_A$, where a statistically significant relationship is present. The first research question, how correlated are the results is answered in the relationship and coefficient of determination depicted in following equation.

$$y = 151.3x + 720.35$$

$$r^2 = 0.5604$$

The statistic tests for the second research question, how the hardware configurations effect the correlation between the benchmarks, were more involved. The multiple regression included the presence or absence of AES instructions, the CPU scale of mobile versus desktop, the CPU manufacturer, and amount of memory. The significance-F for the multiple regression is much stronger than that of the linear regression at 1.38E-15. Furthermore, the correlation coefficient of 0.9382 and resulting coefficient of determination of 0.8801 are much stronger than the results of the linear regression test used for R1. The effect of AES instructions is the strongest with a P-value of 7.46E-09. This impact was already observed in the Figure 7, as five of the six lowest points lacked AES instructions, but now confirmed by the multiple regression. CPU manufacturer is next with a P-value of 0.006924. Memory and scale are insignificant with 0.056246 and 0.643819 for P-values respectively.

These results led to conclusions for the hypotheses for research question two. For hypothesis one, the effect of AES instructions on the correlations was the strongest, which led to the rejection of the null hypothesis $H1_0$ and the acceptance of the alternate $H1_A$. Neither component of CPU, type nor scale, had significant effect on the relationship so the null hypothesis $H2_0$ could not be rejected. Similarly, the amount of memory did not have a significant impact so $H3_0$ could not be rejected either.

Summary of Chapter

This chapter presented the results from the collection process and the analysis of those results in the context of the research questions and hypotheses. The chapter began with a review of the collection process, which was followed by an outline of the facilitators and detailed descriptive statistics for the sample population. The descriptive statistics encompassed controls for validity. CPUs without AES instruction sets were expectedly limited and CPUs manufactured

by AMD were underrepresented, but the results data set was varied and proved useful otherwise. The chapter continued with the presentation of the results from the complete sample population, which formed a recognizable pattern, and was followed by the results as they pertained to each research question. The first research question utilized linear regression and found a weak relationship between AES and floating-point performance. The second research question utilized multiple regression and found a much stronger relationship where the presence or absence of AES instructions had the greatest impact. This chapter concluded with a presentation of the subset of the results with the greatest application as a predictive model.

CHAPTER FIVE

This dissertation examined cryptographic attack feasibility against the advanced encryption standard (AES) using a performance-based approach. The study was quantitative in nature utilizing both linear and multiple regression. The design involved the facilitated execution of benchmark pairs consisting of floating-point operations per second (FLOPS) and AES throughput. The results showed statistically significant correlations between floating point and AES performance. The correlations found through linear regression were significant but expectedly weak for the entire sample population given the inclusion of central processing units (CPUs) both with and without AES instructions.

The multiple regression showed that the presence or absence of AES instructions had the greatest impact on the strength of the relationship. The rest of the components tested by multiple regression had impacts, but none of them qualified as significant. The manufacturer of the CPU had the largest of these impacts. Since the tests ran in memory, the amount of memory was considered, but did not have a significant impact. Scale, mobile versus desktop, had a surprisingly small impact as well. The impact of AES instructions was expected and, since most modern CPUs have these hardware instructions, the subset of the results that excluded the older CPUs without AES instructions were the most applicable for projecting the capability of future systems.

Findings and Conclusions

The experiment involved the facilitated collection of system specifications and benchmark pairs. It took place over the span of a month where 14 facilitators provided benchmark pairs and system specifications from 44 systems. The process involved booting their systems to provided USB drives, containing the collection script, which collected system specifications and ran both AES and FLOPS benchmarks. The specifications involved the details

of the CPU, including AES instructions, amount of memory, and a unique identifier for each system in the sample. Before completion, the collection script reported the results to the web server used for this research.

Upon completion of the collection period, compilation and validation were followed by statistical analysis. Linear regression indicated a present albeit weak relationship for the entire sample population. As the Intel white paper on AES new instructions indicated, it was immediately evident in the results that the presence of AES instructions greatly influenced the ratio between FLOPS and AES performance (Gueron, 2010). Multiple regression confirmed that the largest factor in FLOPS as a predictor for AES performance is the presence or absence of AES instructions. The CPU manufacturer, CPU scale, and amount of memory did not have statistically significant impact on the results.

The relationships present in the results indicate a range of expected performance ratios on other platforms and including near future systems. Since the design of the AES brute force is mode agnostic and incredibly parallelizable, predicting AES performance at scale has universal implications for attack feasibility. Despite the results being a moderate range, they are applicable to form a lower and upper bound as a starting point for such approximations.

Limitations of the Study

The expected limitations were present in the study. Additionally, a few limitations were discovered in the process. The anticipated limitations relating to the element human of the facilitation process and temperature throttling were observed during the collection process. The mitigation technique eliminated two of the results that were erroneous from throttling due to overheating. Difficulty with the BIOS settings component of the facilitation process also caused several of the facilitators to be unable to provide system benchmark results. The mitigation for this limitation was availability for assistance, however, the mitigation was only partly successful;

a few facilitators were still unable to boot to the USB device. A related but unforeseen limitation was present in the collection mechanism; the sequential order of the benchmarks appeared to impact at least one result and should be further mitigated in follow-on research.

The delimitations, which constrained the scope of the study to a manageable level, also limited the external validity of the results. Only x86_64 CPUs were included, leaving the potential of ARM and GPU compute unassessed by this study. Unlike manufacturer optimized variants of Linpack, which expectedly have significant performance benefits for their respective CPUs, the use of high-performance Linpack (HPL) on single systems provides a stable benchmark across CPU manufacturers. HPL is not as common on individual systems, so relating the results to those already readily available for individual CPUs is not possible. HPL is the standard for the high-performance computing field and supercomputers where the concentrated performance potential will allow attacks to approach the feasible point first.

Implications for Practice

Since the presence of AES instructions had significant effect on the relationship between FLOPS and AES performance in addition to AES instructions being present in most modern CPUs, the subset of the results with AES instructions are most useful for projecting attack performance potential. The following equation depicts the results of the linear regression fit line.

$$y = 90.425x + 1543.9$$

$$r^2 = 0.5643$$

The analysis of the average ratio of AES throughput in MB/s to Giga-FLOPS of 268.6048 for systems with AES instructions is another useful result for the study. The above equation or average could be quickly applied to published performance measurements of the top supercomputers to provide rough estimates of their attack potential. Similarly, they could be applied to a botnet of known size or computational capability in addition to a variety of

theoretical systems or configurations. Lastly as a predictive measure, this study or the results of other follow-on studies might allow for projection of intersection power between increasing global compute availability and AES attack requirements.

Recommendations for Future Research

This dissertation may serve as a foundation for future research on performance-based attack feasibility. The lessons learned and limitations encountered in the process of planning and conducting this study may allow a future study with a revised collection construct to improve upon the accuracy of the results. Additional computational platforms such as graphics processing units (GPU) and ARM may also be the most fitting future topics, but the potential of application specific integrated circuits (ASICs) may warrant additional research as well. Future research may also benefit from employing custom AES benchmarks to emulate the different components of a brute force attack rather than the single component emulated in this study. In the predictive sense, the results of this study may serve as a starting point for estimating the attack potential of high-performance systems, including supercomputers and distributed systems.

GPUs lack AES hardware instructions. However, GPUs excel at highly parallel tasks and the designs of many AES attacks align well with this pattern. Furthermore, GPUs make up a significant portion of global compute. These platform advantages of GPUs may warrant follow-on research to assess their attack potential. Research on the attack capability of GPUs may also align with the work done by Biryukov and Großschädl (2012) on application-specific, GPU-like hardware. Although their work is an excellent addition to the field, the changes in computing performance since 2012 may warrant an updated assessment of the attack potential of an updated, theoretical AES supercomputer. Methods in the literature on performance measurement and prediction would support a similar study for projecting the expected capability of GPUs (Madougou, Varbanescu, Laat, & Nieuwpoort, 2016; Phillips & Fatica, 2014). In addition to

GPUs, the ARM architecture may deserve a separate but similar study. ARM is continuing to gain popularity and now includes AES instructions on many models. Unlike GPUs, the prevalence of ARM is in the sheer number of devices rather than the greater potential for each device.

The attack feasibility element of this research may augment current methods or serve as another technique for key length recommendations. As the computing environment continues to evolve, the scaling of this research and use of the results as a predictive mechanism when combined with the rate of change of available compute may be significant. This potential application and future study align with the National Institute of Standards and Technology (NIST) recommendations for key lengths and transitions (Barker & Dang, 2015; Barker & Roginsky, 2011; Barker & Roginsky, 2015).

Conclusion

This dissertation focused on AES performance, its relationship to general purpose performance, and what that relationship could mean in terms of attack feasibility. The topic was chosen to augment the field of cryptographic attack feasibility by providing an additional method and results. The results demonstrated a significant relationship between general performance and AES performance. Although the relationship was not overly strong, the results do provide a range of expected performance, which may help improve upon the community's capability to predict attack feasibility for AES key lengths. Finally, the study may serve as a foundation for future attack feasibility research and additional methods.

REFERENCES

Advanced Micro Devices, Inc. (2018). *Product resource center*. Retrieved from http://products.amd.com/en-us

Agosta, G., Barenghi, A., De Santis, F., & Pelosi, G. (2010). Record setting software implementation of DES using CUDA. *Proceedings of Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, (pp. 748-755). doi:10.1109/ITNG.2010.43

Alghazzawi, D. M., Hasan, S. H., & Trigui, M. S. (2014). Advanced encryption standard - Cryptanalysis research. *Proceedings of Computing for Sustainable Global Development (INDIACom), 2014 International Conference on*, (pp. 763-770). doi:10.1109/IndiaCom.2014.6828045

Bahrak, B., & Aref, M. R. (2008). Impossible differential attack on seven-round AES-128. *The Institution of Engineering and Technology - Information security, 2*(2), 28-32. doi:10.1049/iet-ifs:20070078

Balsamo, S., Di Marco, A., Inverardi, P., & Simeoni, M. (2004). Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering, 30*(5), 295-310. doi:10.1109/TSE.2004.9

Barker, E., & Dang, Q. (2015). *Recommendation for key management* (NIST Special Publication No. 800-57 Part 3 Revision 1). doi:10.6028/NIST.SP.800-57pt3r1

Barker, E., & Roginsky, A. (2011). *Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths* (NIST Special Publication No. 800-131A). doi:10.6028/NIST.SP.800-131A

Barker, E., & Roginsky, A. (2015). *Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths* (NIST Special Publication No. 800-131A Revision 1). doi:10.6028/NIST.SP.800-131Ar1

Baudron, O., Gilbert, H., Granboulan, L., Handschuh, H., Joux, A., Nguyen, P., . . . & Vaudenay, S. (1999). Report on the AES Candidates. *Proceedings from the Second Advanced Encryption Standard Candidate Conference.* Retrieved from http://csrc.nist.gov/archive/aes/round1/conf2/papers/baudron1.pdf

Bernstein, D. J., Buchmann, J., Dahmen, E., Ding, J., Hallgren, S., Micciancio, D., . . . Yang, B. (2009). *Post-quantum cryptography.* (D. J. Bernstein, J. Buchmann, & E. Dahmen, Eds.) Berlin, Germany: Springer-Verlag.

Berry, R. (1992). Computer benchmark evaluation and design of experiments: A case study. *IEEE Transactions on Computers, 41*(10), 1279-1289. doi:10.1109/12.166605

Biham, E., & Shamir, A. (1993). *Differential cryptanalysis of the data encryption standard.* New York: Springer-Verlag. doi:10.1007/978-1-4613-9314-6

Biryukov, A., & Großschädl, J. (2012). Cryptanalysis of the full AES using GPU-like special-purpose hardware. *Fundamenta Informaticae, 114*(3-4), 221-237. doi:10.3233/FI-2012-626

Biryukov, A., & Khovratovich, D. (2009). Related-key cryptanalysis of the full AES-192 and AES-256. *Advances in Cryptology ASIACRYPT 2009,* 1-18. doi:10.1007/978-3-642-10366-7

Biryukov, A., & Khovratovich, D. (2010). Feasible attack on the 13-round AES-256. *International Association for Cryptologic Research Cryptology ePrint Archive*. Retrieved from http://eprint.iacr.org/2010/257.pdf

Biryukov, A., Khovratovich, D., & Nikolić, I. (2009). Distinguisher and related-key attack on the full AES-256. *Advances in Cryptology - CRYPTO 2009,* 231-249). doi:10.1007/978-3-642-03356-8_14

Bogdanov, A., Khovratovich, D., & Rechberger, C. (2011). Biclique cryptanalysis of the full AES. *Proceedings of Theory and Application of Cryptology and Information Security, International Conference on the, in ASIACRYPT 2011: Advances in Cryptography*, pp. 344-371. doi:10.1007/978-3-642-25385-0_19

Bouillaguet, C., Derbez, P., Dunkelman, O., Fouque, P., Keller, N., & Rijmen, V. (2012). Low-data complexity attacks on AES. *Transactions on Information Theory, 58*(11), 7002-7017. doi:10.1109/TIT.2012.2207880

Burr, W. E. (2003). Selecting the advanced encryption standard. *IEEE Security & Privacy*, *99*(2), 43-52. doi:10.1109/MSECP.2003.1193210

Chen, R. (2011). *Automatic tuning of the high performance Linpack benchmark* (Master's thesis, Australian National University). doi:10.1.1.357.5368

Choy, J., Zhang, A., Khoo, K., Henricksen, M., & Poschmann, A. (2011). AES variants secure against related-key differential and boomerang attacks. *International Association for Cryptologic Research Cryptology ePrint Archive*. Retrieved from http://eprint.iacr.org/2011/072.pdf

Coppersmith, D. (1994). The data encryption standard (DES) and its strength against attacks. *IBM Journal of Research and Development, 38*(3), 243-250. doi:10.1147/rd.383.0243

Curtin, M., & Dolske, J. (1998). A brute force search of DES keyspace. *;login:, 23*(2), 1-8. Retrieved from http://www.interhack.net/pubs/des-key-crack/des-key-crack.pdf

Daemen, J., & Rijmen, V. (1999). *AES proposal: Rijndael*. Retrieved from http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf

Davis, R. M. (1978). The data encryption standard in perspective. *IEEE Communication Society Magazine, 16*(6), 5-9. doi:10.1109/MCOM.1978.1089771

Department of Health, Education, and Welfare. (1979). *The Belmont report.* Retrieved from https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/

Electronic Frontier Foundation. (1998a). *Cracking DES: Secrets of encryption research, wiretap politics & chip design.* Sebastopol, CA: O'Reilly & Associates, Inc.

Electronic Frontier Foundation. (1998b). *EEF DES cracker machine brings honesty to crypto debate*. Retrieved from https://www.eff.org/press/releases/eff-des-cracker-machine-brings-honesty-crypto-debate

Electronic Frontier Foundation. (1998c). *Frequently asked questions about the electronic frontier foundation's "DES cracker" machine*. Retrieved from https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., & Whiting, D. (2000). Improved cryptanalysis of Rijndael. *International Workshop on Fast Software Encryption.* Berlin, Germany: Springer-Verlag.

Floissac, N., & L'Hyver, Y. (2011). From AES-128 to AES-192 and AES-256: How to adapt differential fault analysis attacks on key expansion. *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC),* 43-53. doi:10.1109/FDTC.2011.15

Gueron, S. (2010). *Intel advanced encryption standard (AES) new instructions set.* Intel Corporation. Retrieved from http://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

Güneysu, T., Kasper, T., Novotný, M., & Paar, C. (2008). Cryptanalysis with COPACOBANA. *IEEE Transactions on Computers, 57*(11), 1498-1513. doi:10.1109/TC.2008.80

Gustafson, J. L., & Todi, R. (1998). Conventional benchmarks as a sample of the performance spectrum. *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, 517-523. doi:10.1109/HICSS.1998.649247

Heroux, M. A. (2013). *Toward a new metric for ranking high performance computing systems* (Sandia Report SAND2013-4744). Sandia National Laboratories. doi:10.2172/1089988

Intel Corporation. (2018). *Product specifications*. Retrieved from https://ark.intel.com/

Internet Live Stats. (2018). *Internet Users*. Retrieved from http://www.internetlivestats.com/internet-users/

Jayasinghe, D., Ragel, R., Ambrose, J. A., Ignjatovic, A., & Parameswaran, S. (2014). Advanced modes in AES: Are they safe from power analysis based side channel attacks. *Computer Design (ICCD), IEEE International Conference on,* 173-180. doi:10.1109/ICCD.2014.6974678

Johnson, D. B. (2000). AES and future resiliency: More thoughts and questions. *Third AES Candidate Conference*, 257-268.

Kahn, D. (1996). *The code-breakers: The comprehensive history of secret communication from ancient times to the internet.* New York, NY: Scribner.

Kelley, S. (2006). *Security implications of using the data encryption standard (DES)* (Internet Engineering Task Force RFC 4772). Retrieved from http://www.rfc-editor.org/rfc/rfc4772.txt

Khan, A. H., Al-Mouhamed, M. A., Almousa, A., Fatayar, A., Ibrahim, A. R., & Siddiqui, A. J. (2014). AES-128 ECB encryption on GPUs and effects of input plaintext patterns on performance. *2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD),* 1-6. doi: 10.1109/SNPD.2014.6888707

Khan, A. K., & Mahanta, H. J. (2014). Side channel attacks and their mitigation techniques. *2014 First International Conference on Automation, Control, Energy and Systems (ACES),* 1-4. doi:10.1109/ACES.2014.6807983

Madougou, S., Varbanescu, A. L., Laat, C. D., & Nieuwpoort, R. V. (2016). A tool for bottleneck analysis and performance prediction for gpu-accelerated applications. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops,* 641-652. doi:10.1109/IPDPSW.2016.198

Manavski, S. (2007). CUDA compatible GPU as an efficient hardware accelerator for AES cryptography. *IEEE International Conference on Signal Processing and Communications*, 65-68. doi:10.1109/ICSPC.2007.4728256

Marculescu, A. (2014). Mobile architecture for distributed brute-force attacks. *Journal of Mobile Embedded and Distributed Systems, 6*(1), 30-37.

Matsui, M. (1993). Linear cryptanalysis method for DES cipher. *Workshop on the Proceedings of Theory and Application of Cryptographic Techniques, in EUROCRYPT 1993,* 386-397. doi:10.1007/3-540-48285-7_33

McNett, D. (1998). *The secret message is.* Retrieved from http://www.distributed.net/images/6/69/19980224_-_PR_-_news-19980224.pdf

McNett, D. (1999). *US government's encryption standard broken in less than a day.* Retrieved from http://www.distributed.net/images/d/d7/19990119_-_PR_-_release-des3.pdf

Menascé, D. A., Almeida, V. A., & Dowdy, L. W. (2044). *Performance by design: Computer capacity planning by example.* Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE, 86*(1), 82-85. doi:10.1109/JPROC.1998.658762

Moradi, A., Mischke, O., & Parr, C. (2013). One attack to rule them all: Collision timing attack versus 42 AES ASIC cores. *IEEE transactions on computers,* 1786-1798. doi:10.1109/TC.2012.154

National Bureau of Standards. (1978). *Computer security and the data encryption standard* (NBS SP 500-27). Retrieved from http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nbsspecialpublication500-27.pdf

National Institute of Standards and Technology. (1997). *Announcing request for candidate algorithm nominations for the advanced encryption standard (AES).* Retrieved from http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm

National Institute of Standards and Technology. (1999a). *AES2 Conference Attendee Feedback.* doi:http://csrc.nist.gov/archive/aes/round1/conf2/feedback-summary.pdf

National Institute of Standards and Technology. (1999b). *Data encryption standard (DES)* (FIPS PUB 46-3). Retrieved from http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

National Institute of Standards and Technology. (2001). *Announcing the advanced encryption standard (AES)* (FIPS PUB 197). Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

National Institute of Standards and Technology. (2005). *DES transition plan.* Retrieved from http://csrc.nist.gov/groups/STM/common_documents/DESTranPlan.pdf

Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (2000). Report on the development of the advanced encryption standard (AES). *Journal of Research of the National Institute of Standards and Technology, 106*(3), 511-576. doi:10.6028/jres.106.023

Phan, R. C. (2004). Impossible differential cryptanalysis of 7-round Advanced Encryption Standard (AES). *Information Processing Letters, 91*(1), 33-38. doi:10.1016/j.ipl.2004.02.018

Phillips, E., & Fatica, M. (2105). A CUDA implementation of the high performance conjugate gradient benchmark. *5th International Workshop on Performance Modeling,*

Benchmarking and Simulation of High Performance Computer Systems (PMBS14), 68-84. doi:10.1007/978-3-319-17248-4_4

Richards, M. (2001). *AES: The making of a new encryption standard.* SANS Institute InfoSec Reading Room. Retrieved from https://www.sans.org/reading-room/whitepapers/vpns/aes-making-encryption-standard-740

Roback, E., & Dworkin, M. (1999). Conference report: First advanced encryption standard (AES) candidate conference. *Journal of Research of the National Institute of Standard and Technology, 104*(1). doi:10.6028/jres.104.007

RSA Laboratories. (1999). *DES Challenge III.* Retrieved from http://www.emc2.fm/emc-plus/rsa-labs/historical/des-challenge-iii.htm

RSA Laboratories. (n.d.). *The RSA laboratories secret-key challenge.* Retrieved from http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-laboratories-secret-key-challenge.htm

SANS Institute. (2001). *History of encryption.* SANS Institute InfoSec Reading Room. Retrieved from https://www.sans.org/reading-room/whitepapers/vpns/history-encryption-730

Schneier, B., & Whiting, D. (2000). A Performance Comparison of the Five AES Finalists. *Third Advanced Encryption Standard Candidate Conference*, 123-135. Retrieved from http://csrc.nist.gov/archive/aes/round2/conf3/papers/17-bschneier.pdf

Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., & Ferguson, N. (1999). *Performance comparison of the AES submissions.* Retrieved from http://csrc.nist.gov/archive/aes/round1/conf2/papers/schneier1.pdf

Shao, F., Chang, Z., & Zhang, Y. (2010). AES encryption algorithm based on the high performance computing of GPU. *Second International Conference on Communication Software and Networks*, 588-590. doi:10.1109/ICCSN.2010.124

Singh, S. (2000). *The code book: The science of secrecy from ancient Egypt to quantum cryptography.* New York, NY: Anchor Books.

Soleimany, H., Sharifi, A., & Aref, M. (2010). Improved related-key boomerang cryptanalysis of AES-256. *2010 International Conference on Proceedings of Information Science and Applications (ICISA)*, 1-7. doi:10.1109/ICISA.2010.5480302

Tech Power Up. (2018). *GPU specs database*. Retrieved from https://www.techpowerup.com/gpu-specs/

Thakur, J., & Kumar, N. (2011). DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International Journal of Emerging Technology and Advanced Engineering, 1*(2), 6-12.

TOP500. (2018). *TOP500 Lists*. Retrieved from https://www.top500.org/lists/

University of California. (2018). *Seti@Home: CPU performance*. Retrieved from https://setiathome.berkeley.edu/cpu_list.php

Vogt, W. P. (2007). *Quantitative research methods for professionals.* Pearson Education, Inc.

Wiener, M. J. (1993). Efficient DES Key Search. In W. Stallings, *Practical Cryptography for Data Internetworks*, 31-79. Los Alamitos, CA: IEEE Computer Society Press.

Yu, H., Xue-cheng, Z., Zheng-lin, L., & Yi-cheng, C. (2008). The research of DPA attacks against AES implementations. *Journal of China Universities of Posts and Telecommunications, 15*(4). doi:10.1016/S1005-8885(08)60412-4

Yuan, Y., He, Z., Gong, Z., & Qiu, W. (2014). Acceleration of AES encryption with OpenCL. *2014 Ninth Asia Joint Conference on*,*Proceedings of Information Security (ASIA JCIS),* 64-70. doi:10.1109/AsiaJCIS.2014.19

Zhao, X., Guo, S., Zhang, F., Wang, T., Zhijie, S., Liu, Z., & Gallais, J. (2013). A comprehensive study of multiple deductions-based algebraic trace driven cache attacks on AES. *Computers & Security, 39*(B), 173-189. doi:10.1016/j.cose.2013.07.002

Zhou, Y., & Feng, D. (2005). Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *International Association for Cryptologic Research Cryptology ePrint Archive*. Retrieved from http://eprint.iacr.org/2005/388.pdf

APPENDIX A

Web Server Configuration

The configuration is in bash script format to enhance its clarity, but the it is not intended to be executed script as it lacks error checking. Instead, the commands should be entered individually to confirm the success of each command.

```
#Distro: Amazon Linux AMI 2016.09.0 (HVM), SSD Volume, 64-bit
#Type: t2.micro variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon, 1 GiB RAM

#Setup Commands:
  #install apache and php, run apache with system start, confirm config
  sudo yum update -y # -y to skip additional prompts
  sudo yum install -y httpd24 php56
  sudo service httpd start
  sudo chkconfig httpd on
  chkconfig --list httpd # should show 2, 3, 4, 5

  #make group, add user, confirm config
  sudo groupadd www
  sudo usermod -a -G www ec2-user
  logout
  groups

  #set owner, set permissions, recurse permissions, confirm config
  sudo chown -R root:www /var/www
  sudo chmod 2775 /var/www
  find /var/www -type d -exec sudo chmod 2775 {} \;
  find /var/www -type f -exec sudo chmod 0664 {} \;
  ls -al /var/www

  #make the data directory, set permissions, confirm config
  sudo mkdir /srv/data
  sudo chown apache:apache /srv/data
  sudo chmod 777 /srv/data
  ls -al /srv/data
```

APPENDIX B

Web Server Data Collection Script

```
<?php
  //simple pre-shared key, used for validity check
  $psk = md5('What a journey');
  //date time for timestamps
  date_default_timezone_set("America/New_York");
  //if sumbit is posted
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
    //check if the system_hash is valid
    if (md5($psk.$_POST["uuid"]) == $_POST["system_hash"]) {
      //open the file for the system_hash, create if not found
      $results = fopen("/srv/data/".$_POST["system_hash"].".txt", "a");
      //write the results
      fwrite($results, "date_time ".date("Y/m/d H:i:s")."\n");
      fwrite($results, "system_hash ".$_POST["system_hash"]."\n");
      fwrite($results, "uuid ".$_POST["uuid"]."\n");
      fwrite($results, "device_id ".$_POST["device_id"]."\n");
      fwrite($results, "cpu_model ".$_POST["cpu_model"]."\n");
      fwrite($results, "cpu_sig ".$_POST["cpu_sig"]."\n");
      fwrite($results, "cpu_arch ".$_POST["cpu_arch"]."\n");
      fwrite($results, "memory ".$_POST["memory"]."\n");
      fwrite($results, "aes_inst ".$_POST["aes_inst"]."\n");
      fwrite($results, "aes_bench ".$_POST["aes_bench"]."\n");
      fwrite($results, "flops_bench ".$_POST["flops_bench"]."\n");
      //separate successive results from the same system
      fwrite($results, $result . "\n\n");
      //close the file, announce success
      fclose($results);
      echo "Submit successful!";
    } else {
      echo "Submit failed, invalid system_hash.";
    }
  }
?>
<!DOCTYPE html>
<html>
  <body>
    <form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
      system_hash <input type="text" name="system_hash"><br>
      uuid <input type="text" name="uuid"><br>
      device_id <input type="text" name="device_id"><br>
      cpu_model <input type="text" name="cpu_model"><br>
      cpu_sig <input type="text" name="cpu_sig"><br>
      cpu_arch <input type="text" name="cpu_arch"><br>
      memory <input type="text" name="memory"><br>
      aes_inst <input type="text" name="aes_inst"><br>
      aes_bench <input type="text" name="aes_bench"><br>
      flops_bench <input type="text" name="flops_bench"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

APPENDIX C

USB Linux Configuration

```
#Author: Daniel Hawthorne

#Distro: Arch Linux | Release: 2018.05.01 | Kernel: 4.16.5
#Media: SanDisk Cruzer Blade CZ50 8 GB USB 2.0
#Config: ArchISO x86_64

#Description: This document contains the steps used to modify the live
  #ArchISO to include the packages and scripts necessary for use in this
  #research.

#References:
  #https://wiki.archlinux.org/index.php/Archiso
  #https://wiki.archlinux.org/index.php/Remastering the Install ISO

#Note: These instructions are from an Arch Linux-based Distro.  They are
  #written as a bash script, but are not tested in that form.  Rather they
  #are designed to be executed sequentially.

#Step 1: Confirm required packages synced
  sudo pacman -S --needed cdrtools squashfs-tools arch-install-scripts \
    libisoburn syslinux

#Step 2: Get the ISO
  #Download and verify the ISO from you choice of mirrors at:
  #https://archlinux.org/download/
  #will move to below when a new version is released:
  #https://archive.archlinux.org/iso/

#Step 3: Mount the ISO, copy ISO contents, unmount the ISO, clean up mnt

  sudo mkdir /mnt/archiso
  sudo mount -t iso9660 -o loop \
    ~/Downloads/archlinux-2018.05.01-x86_64.iso /mnt/archiso
  #make sure ~/customiso does not already exist:
    sudo rm -r ~/customiso
  sudo cp -a /mnt/archiso ~/customiso
  sudo umount /mnt/archiso
  sudo rm -r /mnt/archiso

#Step 4: Unpack the file system
  cd ~/customiso/arch/x86_64
  sudo unsquashfs airootfs.sfs

#Step 5: Move the benchmark script to the target filesystem
  sudo cp ~/Downloads/bench.sh \
    ~/customiso/arch/x86_64/squashfs-root/etc/profile.d

#Step 6: Modify the custom ISO as root
  #enter ISO filesystem as root
  sudo arch-chroot squashfs-root /bin/bash
```

```
  #add a root password so pkgbuild will work
  passwd root #root

  #prepare the package manager
  pacman-key --init
  pacman-key --populate archlinux

  #add repo for yaourt
  echo -e "\n[archlinuxfr]\nSigLevel = Never\n" >> /etc/pacman.conf
  echo -e "Server = http://repo.archlinux.fr/\$arch" >> /etc/pacman.conf

  #sync package database
  pacman -Sy

  #may need to remove checkspace if error when getting packages
  nano /etc/pacman.conf
  #comment out CheckSpace

  #get packages
  pacman -S dmidecode binutils yaourt fakeroot make patch
  sudo -u nobody yaourt -S hpl #follow prompts to build/install

  #update the package list
  LANG=C pacman -Sl | \
    awk '/\[installed\]$/ {print $1 "/" $2 "-" $3}' > /pkglist.txt

  #clean package database
  pacman -Scc

  #clean bash history and exit chroot
  cat /dev/null > ~/.bash_history && history -c && exit

#Step 7: Create New filesystem
  #move package list
  sudo mv squashfs-root/pkglist.txt ~/customiso/arch/pkglist.x86_64.txt

  #remove old, make new, clean up
  sudo rm airootfs.sfs
  sudo mksquashfs squashfs-root/ airootfs.sfs
  sudo rm -r squashfs-root/
  sudo sha512sum airootfs.sfs | sudo tee airootfs.sha512

#Step 8: Make the new ISO
  cd ~
  #get iso label
  iso_label=$(isoinfo -i ~/Downloads/archlinux-2018.05.01-x86_64.iso -d \
    | grep 'Volume id:' | cut -d' ' -f3)
```

```
  #make the image (USB ready)
  sudo xorriso -as mkisofs \
    -iso-level 3 \
    -full-iso9660-filenames \
    -volid "${iso_label}" \
    -eltorito-boot /isolinux/isolinux.bin \
    -eltorito-catalog /isolinux/boot.cat \
    -no-emul-boot -boot-load-size 4 -boot-info-table \
    -isohybrid-mbr ~/customiso/isolinux/isohdpfx.bin \
    -output ~/arch-custom.iso \
    ~/customiso

  #test with virtual machine, if desired

#Step 9: Write the custom ISO (repeat as necessary):
  sudo fdisk -l #determine usb disk label
  sudo dd bs=4M if=~/arch-custom.iso of=/dev/sdb status=progress
```

APPENDIX D

Benchmark Script

```
#!/bin/bash

#Author: Daniel Hawthorne

echo -e "\nBenchmark script starting in 10 seconds."
echo -e "Press Ctrl+C to exit to Terminal.\n"

sleep 10
echo "Script started!"
echo -n "Setting up environment. "; sleep 1

test_url="http://www.google.com"
#collection server url, may change if taken offline
collection_server="http://ec2-107-21-89-87.compute-1.amazonaws.com"
#psk for validity check
psk=$(echo -n 'What a journey' | md5sum | awk '{print $1}')
#usb device_id
device_id=$(lsblk -d -o name,label,serial | \
  grep ARCH_201805 -m 1 | awk '{print $3}')

#check the connection
echo -ne "Done!\nChecking connection. "; sleep 1
conn=$(curl -s -I --retry 5 --retry-connrefused --url $test_url)

if [[ $conn = *"OK"* ]]; then

  echo -ne "Success!\nChecking results server. "; sleep 1
  rsc=$(curl -s -I --retry 5 --retry-connrefused --url $collection_server)

  if [[ $rsc = *"OK"* ]]; then
    echo -ne "Success!\nGathering system info. "; sleep 1
    #uuid
    uuid=$(sudo dmidecode | grep -i 'uuid' | awk '{print $2}' |\
      tr '[:upper:]' '[:lower:]')
    #system_hash
    system_hash=$(echo -n $psk$uuid | md5sum | awk '{print $1}')
    #cpu_model (model name)
    cpu_model=$(lscpu | grep -i 'model name' |\
      awk '{print substr($0, index($0,$3))}' | sed -e 's/ /_/g')
    #cpu_sig (stepping, model, family)
    cpu_sig=$(echo -n $(lscpu | grep -i 'stepping' | awk '{print $2}')"_"\
      $(lscpu | grep -i 'model' | grep -v 'name' | awk '{print $2}')"_"\
      $(lscpu | grep -i 'family' | awk '{print $3}'))
    #cpu_arch (architecture, sockets, cores per socket, threads per core)
    arch=$(lscpu | grep -i 'architecture' | awk '{print $2}')
    sockets=$(lscpu | grep -i 'socket(s)' | awk '{print $2}')
    cores=$(lscpu | grep -i 'core(s) per socket' | awk '{print $4}')
    threads=$(lscpu | grep -i 'thread(s) per core' | awk '{print $4}')
    cpu_arch=$(echo $arch"_"$sockets"_"$cores"_"$threads)
```

```
    #memory (used, free, total in megabytes)
    unalias free 2> /dev/null
    memory=$(free -m | grep -i 'mem'  | awk '{print $3 "_" $4 "_" $2}')
    #aes_inst
    if [ $(lscpu | grep -ci aes) -ge 1 ];\
      then aes_inst=true; else aes_inst=false; fi
    #aes_bench (in MiB/s [source review confirmed])
    aes_bench=0
    aes_bench_count=10
    echo -ne "Done!\nRunning AES benchmarks. "
    for ((loop=1;loop<=$aes_bench_count;loop++)); do
      aes_bench_next=$(cryptsetup benchmark --cipher aes |\
        grep aes | awk -F' ' '{print $5}')
      aes_bench=$(bc <<< "scale=2; $aes_bench+$aes_bench_next")
    done; sleep 2
    #divide by number of tests
    aes_bench=$(bc <<< "scale=2; $aes_bench/$aes_bench_count")

    #flops_bench setup
    echo -ne "Done!\nRunning FLOPS benchmark. "

    #size_dimensions=sqrt((free memory in bytes * 0.05) / 8)
    free_mem=$(free -b | grep -i 'mem' | awk '{print $4}')
    use_mem=$(bc <<< "scale=2; $free_mem*0.05")
    prob_size=$(bc <<< "scale=0; sqrt($use_mem/8)")

    #prepare the config file
    #reference: www.netlib.org/benchmark/hpl/tuning.html
    #           www.netlib.org/benchmark/hpl/faqs.html
    line1_2="config\nfile\n" #unused
    line3_4="HPL.out\n6\n" #output file, type
    line5_6="1\n$prob_size\n" #num prob sizes, prob size(s)
    line7_8="1\n16\n" #num block sizes, block size(s)
    line9="0\n" #process mapping (0 row-major, 1 column-major)
    line10_12="1\n$(($sockets*$threads))\n$cores\n" #num grids, P, Q
    line13="16.0\n" #residual threshold
    line14_21="1\n2\n1\n4\n1\n2\n1\n1\n"
    line22_23="1\n1\n" #num broadcast, broadcast type [0..5]
    line24_25="1\n0\n" #num look ahead, look ahead depth [0..2]
    line26_27="2\n64\n" #swap type [0..2], swap threshold
    line28_31="0\n0\n1\n8\n"

    #combine the config lines
    config="$line1_2$line3_4$line5_6$line7_8$line9$line10_12$line13"
    config="$config$line14_21$line22_23$line24_25$line26_27$line28_31"

    #write the config file
    echo -e $config | sudo tee /etc/hpl/HPL.dat >/dev/null

    #run the benchmark
    sudo -u nobody mpirun --oversubscribe \
      -n $(($sockets*$cores*$threads)) \
      /usr/bin/xhpl-ompi > ~/HPL.out
```

```
    #get the flops value
    flops_bench=$(cat ~/HPL.out | grep -m 2  Gflops -A2 | \
      awk END{print} | awk '{print $7}')

    #results_string
    echo -ne "Done!\nSubmitting results. "
    results_string=$(echo -n "system_hash="$system_hash"&"\
      "uuid="$uuid"&device_id="$device_id"&cpu_model="$cpu_model"&"\
      "cpu_sig="$cpu_sig"&cpu_arch="$cpu_arch"&memory="$memory"&"\
      "aes_inst="$aes_inst"&aes_bench="$aes_bench"&"\
      "flops_bench="$flops_bench"&submit=Submit" | tr -d '[:space:]')
    #submit results
    if [ $(curl -s -d $results_string \
      $collection_server"/index.php" |\
      head -n1 | grep -ci 'success') -ge 1 ];\
      then echo "Success!"; else echo "Failed."
    fi
  else
    echo "Failed. Proxy or filter blocking *.amazonaws.com? Server down?"
  fi
else
  echo "Failed. Wired connection? DHCP available? Cable attached?"
fi

#goodbye
echo "Script complete!"
echo "Press Enter to Shutdown or Ctrl+C to exit to terminal."
read -s
sudo shutdown now
```

APPENDIX E

Facilitator Google Form

Link to the facilitator Google Form:

https://www.tinyurl.com/dshawth-dcs

Dissertation Research Facilitation

Thank you for your interest in assisting with the facilitation of my dissertation research!

The goal of the research is to better understand, and possibly project, the cryptographic attack potential of general purpose hardware against the Advanced Encryption Standard (AES). The results of this research will be published in my dissertation and hopefully help the security community to better project key length vulnerabilities.

Please review the research design, requirements, and disclaimer below before checking the acknowledgement and continuing to the sign-up page. Thank you in advance for considering the study or assisting with its facilitation!

Respectfully,

Daniel Hawthorne

Research Design

The research involves the collection of performance pairs from a variety of computer hardware configurations. The pairs consist of a general purpose benchmark in terms of floating point operations per second (FLOPS) and an AES benchmark.

The benchmarks are conducted by booting computers to an Arch Linux based USB flash drive. The collection script automatically runs on boot, conducts the benchmarks, and submits the results to an EC2 instance.

The collection is limited to hardware and performance information. The collection script does not access any other storage devices or collect any personal information.

The source code is available for review at the link below:
https://github.com/dshawth/DCS

Requirements

A computer or computers (the more samples, the better) with the following:

1) An x86_64 processor (AMD and Intel, desktop and laptop processors since 2004)
2) The ability to boot to Linux on a USB device (May require enabling Legacy Boot Option ROM)
3) A wired internet connection

You will receive a USB drive that is ready to go or you can choose to download the binary and create one on your own. Power down, plug in, and boot to the drive. The script takes about 5 minutes to run and shuts down when complete. Detailed instructions will be provided with the USB drive.

Data Fields Collected

The collection script gathers the follow system information fields:

1) Universally Unique Identification (UUID)
2) Device ID (of the USB drive)
3) CPU: model name, signature (stepping, model, family), and architecture (sockets, cores per socket, threads per core)
4) Memory: Total, free, used
5) AES Instructions: Present/Not

The benchmark fields are:

1) Advanced Encryption Standard (AES) Benchmark
2) Floating point operations per second (FLOPS) Benchmark

Disclaimer

Booting to a live USB can be challenging. It may require you to make changes in your BIOS to change your boot order.

Changes in your BIOS should be done with caution, as the researcher cannot be held responsible for damages due to BIOS configuration mistakes.

If you have any questions or need help with the process, please reach me at daniel.s.hawthorne@gmail.com.

If you meet the requirements and agree to these terms, please continue and thank you in advance for your support!

Acknowledgement

☐ I meet the requirements, understand fields to be collected, acknowledge the risks of booting to USB, and consent to assist with the facilitation of this research.

Dissertation Research Facilitation

* Required

Facilitation Sign Up

Name (as you want it to appear in the publication where applicable) *

Your answer

Include name and number of result(s) in the final publication *

- (●) Yes, please.
- ( ) No, thank you.

Email *

Your answer

Phone number

Your answer

Comments

Your answer

How do you want to receive the USB drive? *

- ( ) Local pickup (West Point, NY)
- ( ) Mail (Please include your full mailing address below)
- ( ) Download the binary (Please include device ID below)

Complete Mailing Address (For mail option only)

Your answer

Download Instructions (For download option only)

Follow the link below to get the ISO:

https://github.com/dshawth/DCS/releases

LINUX:

Get the serial number of your USB drive for the device ID field below using terminal:

fdisk -l #to determine the sdX of your target device
lsblk -d -o name,label,serial #to get the serial

Write the USB using:

dd bs=4M if=~/Downloads/arch-custom.iso of=/dev/sdX status=progress

WINDOWS:

Get the serial number of your USB drive for the device ID field below using PowerShell:

gwmi Win32_USBControllerDevice |%{[wmi]($_.Dependent)} | Where-Object {($_.Description -like '*mass*')} | Sort Description,DeviceID | ft Description,DeviceID –auto

The serial number is right of the last \ in DeviceID.

Write the USB using Rufus with default options; when prompted choose the DD method.

Device ID (For download option only)

Your answer

APPENDIX F

USB Instructions

Dear Facilitator,

Thank you again for your willingness to help with my research pertaining to the feasibility of cryptographic attacks against the advanced encryption standard (AES)! The included USB drive is already imaged and ready for use. Please help me keep this research on a manageable timeframe by conducting the tests at your earliest convenience.

Shutdown each computer and attach the USB drive. Power on the system and enter the boot menu by tapping the key designated by the manufacturer (commonly F12) as soon as the computer begins to power on.

If you miss the boot menu, you may have to shut down and try again. If the USB drive does not appear on the list of bootable devices, you may have to enable legacy option ROMs in your BIOS. To enter the BIOS, tap the key designated by the manufacturer (commonly DEL or F2) as soon as the computer begins to power on.

Once you have successfully selected the USB drive, an Arch Linux boot menu will display; choose the first option and the boot process will continue. Once started, the script will collect the specs from your system, conduct the benchmarks, submit the results, and wait for you to confirm shutdown. Once powered off, remove the USB drive and repeat the process on other systems where possible.

The data fields and source are available on the survey form at https://tinyurl.com/dshawth-dcs.

Feel free to reach me with any questions at daniel.s.hawthorne@gmail.com.

Once complete with your system(s), the more the better, the USB drive is yours to keep!

Respectfully,
Daniel Hawthorne

APPENDIX G

Data Compilation Script

```
#Program: compile-data.py
#Author: Daniel Hawthorne
#Python: 3.6.2

import os #for files in folder

rawFolder = r"X:\Google Drive\School\Dissertation\Results\Raw"
fields = ["date_time", "system_hash", "uuid", "device_id", "cpu_model", \
"cpu_sig", "cpu_arch", "memory", "aes_inst", "aes_bench", "flops_bench"]

outFile = open("results.csv", 'w')

#write fields as headers
outFile.write(','.join(fields) + '\n')

for fileName in os.listdir(rawFolder):

    if fileName.endswith('.txt'):
        print(fileName)
        #open, read, close the file
        recordFile = open(os.path.join(rawFolder,fileName), 'r')
        fileLines = list(filter(None, recordFile.read().splitlines()))
        recordFile.close()
        lines = []
        for line in fileLines:
            #remove any commas from cpu strings
            line = line.replace(',', '_')
            #fields
            if len(line.split()) > 1:
                lines.append('_'.join(line.split()[1:]))
            else:
                lines.append('')
            #last line
            if line.startswith('flops_bench'):
                lines.append('\n')
                outFile.write(','.join(lines))
                lines = []
outFile.close()
```

APPENDIX H

Raw Results

date_time	system_hash	uuid	device_id	cpu_model	cpu_sig	cpu_arch
2018/05/17_10:35:36	4dfbd6a7b7e30c20c243d17c3dafa5cb	4c4c4544-0053-4d10-8031-b5c04f503232	4C530001150804105195	Intel(R)_Core(TM)_i5-4200M_CPU_@_2.50GHz	3_60_6	x86_64_1_
2018/05/17_10:53:48	d7f3ea5d9770ce0b826a1c5312b37b09	c1b8e060-d7e2-11dd-8eb0-704d7b2d05b4	4C530001150804105195	Intel(R)_Core(TM)_i7-6700K_CPU_@_4.00GHz	3_94_6	x86_64_1_
2018/05/17_11:12:04	5bcc9a1780b3cd4b98bba1ea07972ad6	03d502e0-045e-0528-9f06-860700080009	4C530001150804105195	AMD_A8-9600_RADEON_R7__10_COMPUTE_CORES_4C_6G	1_101_21	x86_64_1_
2018/05/17_11:26:48	45b0f1d91d17f9b18d89a92e9bb061ee	1e5c2c40-d7da-11dd-a6e8-2c4d54d85f4a	4C530001150804105195	Intel(R)_Core(TM)_i7-7700K_CPU_@_4.20GHz	9_158_6	x86_64_1_
2018/05/17_20:09:26	de20148de31143079a7eb0782a93e0b8	4c4c4544-0032-5a10-8038-b7c04f574432	4C530001150804105195	Intel(R)_Core(TM)_i7-6820HQ_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/05/24_10:14:31	f8c9542df4177659b8fcb0bf9b46bc65	4c4c4544-0020-2010-8020-a0c04f202020	4C530001150804105195	Intel(R)_Core(TM)_i5-3210M_CPU_@_2.50GHz	9_58_6	x86_64_1_
2018/05/24_14:06:04	c5f1f2c3746e88e6684d1c3e9d4e7db9	4c4c4544-0051-3910-804c-c7c04f4e4e32	4C530001150804105195	Intel(R)_Core(TM)_i5-7200U_CPU_@_2.50GHz	9_142_6	x86_64_1_
2018/05/24_14:25:09	5631c6a162718e2cd28e71e340042b8c	4c4c4544-0052-3510-8047-b5c04f523732	4C530001150804105195	Intel(R)_Core(TM)_m5-6Y57_CPU_@_1.10GHz	3_78_6	x86_64_1_
2018/05/28_21:18:00	9e5e5c36904f1fd245dc331a4e9ed078	4c4c4544-004d-5810-8038-b8c04f514332	4C530001200804105193	Intel(R)_Core(TM)_i7-6820HK_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/05/28_21:34:44	978ad4f56200102e09d184313dc483bf	4c4c4544-005a-5210-8050-b5c04f5a3332	4C530001200804105193	Intel(R)_Core(TM)_i5-7300HQ_CPU_@_2.50GHz	9_158_6	x86_64_1_
2018/05/29_22:05:35	173af41a127bd505e429e27932d3e598	21a98280-d7da-11dd-87d1-08626637239b	4C531001620804101070	Intel(R)_Core(TM)_i5-4570T_CPU_@_2.90GHz	3_60_6	x86_64_1_
2018/05/29_22:22:45	d275336c15f3fc5153d44a7b1a1b577e	4c4c4544-0035-5610-804c-c3c04f334c32	4C531001620804101070	Intel(R)_Core(TM)_i7-7500U_CPU_@_2.70GHz	9_142_6	x86_64_1_
2018/05/29_23:23:08	c1d6d95fedb8ffd62949427565381b79	d21772a0-72ba-11e3-88a2-305a3a8222dc	4C530001150804105195	Intel(R)_Core(TM)_i7-6700T_CPU_@_2.80GHz	3_94_6	x86_64_1_
2018/05/30_19:39:26	5da0933b47218205fa61c81086fdcc0a	88c11c80-d7da-11dd-aff7-305a3a00771a	4C530001150804105195	Intel(R)_Core(TM)_i5-6500_CPU_@_3.20GHz	3_94_6	x86_64_1_
2018/05/30_19:57:03	c418fe3fd031f3ebf142ebcba35c1d1b	295b6b60-d7da-11dd-9913-704d7b2e5e38	4C530001150804105195	Intel(R)_Core(TM)_i5-6500_CPU_@_3.20GHz	3_94_6	x86_64_1_
2018/06/01_18:16:09	03363a2ac12693b214182a411d1c1a89	031b021c-040d-05e6-1806-650700080009	4C530001150804105195	Intel(R)_Pentium(R)_CPU_G4400_@_3.30GHz	3_94_6	x86_64_1_
2018/06/02_17:07:36	ddfe6bf874d72ed2ba92bd207aceab5a	a0908ac6-3324-eb45-a32f-7b530e11728d	4C530001110804105303	Intel(R)_Core(TM)_i7-4710HQ_CPU_@_2.50GHz	3_60_6	x86_64_1_
2018/06/02_17:42:52	0ea8fd37c987e0a84c9e45247bc59cab	35304535-3439-3541-3946-3942ffffffff	4C530001110804105303	AMD_A8-3870_APU_with_Radeon(tm)_HD_Graphics	0_1_18	x86_64_1_
2018/06/02_17:48:08	73d2c70811efeb397b6aa4719407c570	644d0e80-b900-11dc-ae7a-e03f499f8964	4C530001110804105303	AMD_FX(tm)-8320_Eight-Core_Processor	0_2_21	x86_64_1_
2018/06/02_19:00:04	b3e5393a4c152b24e274967b4681daf0	038d0240-045c-051a-4b06-3f0700080009	4C530001150804105195	Intel(R)_Core(TM)_i5-6400_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/06/05_23:13:22	4ba558288eb42c39107dc0dda426070e	4c4c4544-0037-4210-8035-c8c04f425131	4C531001620804101070	Intel(R)_Core(TM)_i5-2400S_CPU_@_2.50GHz	7_42_6	x86_64_1_
2018/06/06_15:16:22	beb8e8373ac6f858bd9d0bdd6e1cc67c	44454c4c-3500-1043-8036-b3c04f584431	4C531001530804101071	Intel(R)_Core(TM)2_Duo_CPU_____T7500__@_2.20GHz	10_15_6	x86_64_1_
2018/06/07_06:30:36	355904f000fd84d5884d1af3365f55dd	d96d3c2a-7df6-11db-82cc-0011113186f6	4C530001260804101000	Intel(R)_Core(TM)2_CPU_____X6800__@_2.93GHz	6_15_6	x86_64_1_
2018/06/07_14:55:30	a02fd28b03a1c3c5ead1d0cbc140afe1	4c4c4544-0032-4210-804b-cac04f573532	070A7AED87BD3954	Intel(R)_Core(TM)_i7-4940MX_CPU_@_3.10GHz	3_60_6	x86_64_1_
2018/06/07_15:05:32	66db2b7ef947acd4e481c10210efb619	4c4c4544-004c-5310-8051-c6c04f534732	070A7AED87BD3954	Intel(R)_Core(TM)_i5-6300U_CPU_@_2.40GHz	3_78_6	x86_64_1_
2018/06/07_21:08:59	07f741d29304d8d1f22afbd0bd6a337a	3fbaf381-538a-11cb-834f-a1949e4827f5	4C530001260804101000	Intel(R)_Core(TM)_i7-4600U_CPU_@_2.10GHz	1_69_6	x86_64_1_
2018/06/07_22:01:00	ae37c2ee0d328312d11c7d81fb83db78	4a0d6f4c-2924-11b2-a85c-a3c4091f1442	4C530001260804101000	Intel(R)_Core(TM)_i7-6600U_CPU_@_2.60GHz	3_78_6	x86_64_1_
2018/06/08_10:33:57	9712eecf5b6e68e0e3a7fe8b19238271	4c4c4544-0032-3910-8054-cac04f573532	4C530001270804101000	Intel(R)_Core(TM)_i7-4940MX_CPU_@_3.10GHz	3_60_6	x86_64_1_
2018/06/08_10:53:47	af55e9e5f6bae00d243082d724ad05a5	4c4c4544-0032-3910-8050-cac04f573532	4C530001270804101000	Intel(R)_Core(TM)_i7-4940MX_CPU_@_3.10GHz	3_60_6	x86_64_1_
2018/06/08_11:28:30	9e5e5c36904f1fd245dc331a4e9ed078	4c4c4544-004d-5810-8038-b8c04f514332	4C530001270804101000	Intel(R)_Core(TM)_i7-6820HK_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/06/08_11:38:30	d03f87ce7c9567e5194a16bbbf7c8fa3	6891600c-4877-440a-bc49-b049ad7f12c7	4C530001270804101000	Intel(R)_Core(TM)_i7-6770HQ_CPU_@_2.60GHz	3_94_6	x86_64_1_
2018/06/08_12:00:40	b6ecc3b44f352783e57c0ecf6af48038	5113027f-7037-11e4-bb8f-38a21a2efcff	4C530001270804101000	Intel(R)_Core(TM)_i7-4600M_CPU_@_2.90GHz	3_60_6	x86_64_1_
2018/06/08_16:23:22	159fb278dcb9c616fb693ab155afc30a	03000200-0400-0500-0006-000700080009	4C530001150804105090	AMD_Phenom(tm)_II_X6_1045T_Processor	0_10_16	x86_64_1_
2018/06/09_19:16:13	ed7422ef35fbdcb6a72bc9b918cd1518	00000000-0000-0000-0000-448a5bce8647	4C530001260804105083	Intel(R)_Core(TM)_i7-4790K_CPU_@_4.00GHz	3_60_6	x86_64_1_
2018/06/13_18:07:23	7a39abda3e04d9a4276f73e33254ba80	27b02c60-d7da-11dd-956c-38d547aac192	4C530001160804105194	Intel(R)_Core(TM)_i7-6700_CPU_@_3.40GHz	3_94_6	x86_64_1_
2018/06/14_16:08:04	68cb09e52bfa08e63b16d844279a9355	038d0240-045c-05b8-7e06-a00700080009	4C530001210524108581	Intel(R)_Core(TM)_i7-6700K_CPU_@_4.00GHz	3_94_6	x86_64_1_
2018/06/15_17:06:15	f8f0cf9fd0a9909539b298ae85ab9917	41f1e380-4c54-c742-b8ef-df1a395882db	4C530001210524108581	Intel(R)_Celeron(R)_CPU__N3050__@_1.60GHz	3_76_6	x86_64_1_
2018/06/15_17:35:12	f50734cd14ba669b7979390cda8ab3e0	4c4c4544-0031-5910-8032-b5c04f483132	4C530001210524108581	Intel(R)_Core(TM)_i5-4300U_CPU_@_1.90GHz	1_69_6	x86_64_1_
2018/06/21_12:23:16	4ad2a47ac5250017884014ab561f01e7	aaffcdc0-a9b5-d359-951d-137db2ca81e8	4C530001150804105195	Intel(R)_Core(TM)_i5-2415M_CPU_@_2.30GHz	7_42_6	x86_64_1_
2018/06/28_09:50:37	def6e03aa246cd9837ac29066c966e5c	44ec9c00-d7da-11dd-a5d8-086266c5cece	4C530001290804105300	Intel(R)_Core(TM)_i5-4690K_CPU_@_3.50GHz	3_60_6	x86_64_1_
2018/06/28_11:27:28	39f653d247031aa1de8c1207c07aaa55	b1b3e073-3709-11e8-8a95-8c16455f56b3	4C530001290804105300	Intel(R)_Core(TM)_i5-8250U_CPU_@_1.60GHz	10_142_6	x86_64_1_
2018/07/12_11:37:48	0682cbbe5e36773756166ba6138d33cf	4c4c4544-0033-5a10-8038-b6c04f574432	4C530001130804105302	Intel(R)_Core(TM)_i7-6820HQ_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/07/13_15:48:50	720dba50bf57ccd413158cd281e8f248	4c4c4544-004c-5a10-8038-c3c04f574432	4C530001130804105302	Intel(R)_Core(TM)_i7-6820HQ_CPU_@_2.70GHz	3_94_6	x86_64_1_
2018/07/16_20:17:34	355455e3a77d07680d86e5ced0b34e89	73582780-d7da-11dd-b4b0-1831bfb4e152	4C530001150804105195	Intel(R)_Core(TM)_i7-8086K_CPU_@_4.00GHz	10_158_6	x86_64_1_

APPENDIX I

Results

result	cpu	mem	aes_inst	aes_bench	gflops
1	Intel Core i5-4200M	4 GB	TRUE	1922.36	5.305
2	Intel Core i7-6700K	32 GB	TRUE	3186	16.03
3	AMD A8-9600	4 GB	TRUE	1403.69	6.008
4	Intel Core i7-7700K	32 GB	TRUE	3400.2	17.07
5	Intel Core i7-6820HQ	16 GB	TRUE	2699.43	11.18
6	Intel Core i5-3210M	6 GB	FALSE	154.1	5.008
7	Intel Core i5-7200U	8 GB	TRUE	2330.57	5.856
8	Intel Core m5-6Y57	8 GB	TRUE	2101.33	3.681
9	Intel Core i7-6820HK	32 GB	TRUE	2705.37	12.13
10	Intel Core i5-7300HQ	8 GB	TRUE	2622.02	11.14
11	Intel Core i5-4570T	16 GB	TRUE	2240.96	6.046
12	Intel Core i7-7500U	8 GB	TRUE	2637.85	6.017
13	Intel Core i7-6700T	16 GB	TRUE	2701.22	12.03
14	Intel Core i5-6500	8 GB	TRUE	2720.82	11.63
15	Intel Core i5-6500	8 GB	TRUE	2722.88	11.62
16	Intel Pentium G4400	4 GB	TRUE	2505.69	5.807
17	Intel Core i7-4710HQ	16 GB	TRUE	2173.13	11.1
18	AMD A8-3870	16 GB	FALSE	134.62	3.858
19	AMD FX-8320	16 GB	TRUE	1612.13	11.46
20	Intel Core i5-6400	16 GB	TRUE	2474.42	11.56
21	Intel Core i5-2400S	6 GB	TRUE	1432.64	8.052
22	Intel Core 2 Duo T7500	4 GB	FALSE	104.71	2.181
23	Intel Core 2 Ext X6800	8 GB	FALSE	127.94	2.761
24	Intel Core i7-4940MX	16 GB	TRUE	2485.12	12.24
25	Intel Core i5-6300U	8 GB	TRUE	2220.39	5.54
26	Intel Core i7-4600U	8 GB	TRUE	2049.84	3.996
27	Intel Core i7-6600U	16 GB	TRUE	2562.98	5.549
28	Intel Core i7-4940MX	16 GB	TRUE	848.54	4.976
29	Intel Core i7-4940MX	16 GB	TRUE	2487.65	12.1
30	Intel Core i7-6820HK	32 GB	TRUE	2703.86	12.14
31	Intel Core i7-6770HQ	32 GB	TRUE	2627.38	12.23
32	Intel Core i7-4600M	16 GB	TRUE	2242.54	6.408
33	AMD Phenom II X6 1045T	8 GB	FALSE	143.96	7.667
34	Intel Core i7-4790K	16 GB	TRUE	2743.04	16.01
35	Intel Core i7-6700	16 GB	TRUE	3016.95	14.17
36	Intel Core i7-6700K	16 GB	TRUE	3148.33	15.03
37	Intel Celeron N3050	4 GB	TRUE	311.17	1.555
38	Intel Core i5-4300U	16 GB	TRUE	1236.97	1.28
39	Intel Core i5-2415M	8 GB	TRUE	1262.82	4.465
40	Intel Core i5-4690K	16 GB	TRUE	2690.83	14.94
41	Intel Core i5-8250U	8 GB	TRUE	2555.72	11.35
42	Intel Core i7-6820HQ	16 GB	TRUE	2624	11.4
43	Intel Core i7-6820HQ	16 GB	TRUE	2703.61	10.32
44	Intel Core i7-8086K	16 GB	TRUE	3754.83	23.2

APPENDIX J

Encoded Results

result	aes_bench	gflops	aes_inst	cpu	scale	mem
1	1922.36	5.305	1	1	0	0
2	3186	16.03	1	1	1	4
3	1403.69	6.008	1	0	1	0
4	3400.2	17.07	1	1	1	4
5	2699.43	11.18	1	1	0	3
6	154.1	5.008	0	1	0	1
7	2330.57	5.856	1	1	0	2
8	2101.33	3.681	1	1	0	2
9	2705.37	12.13	1	1	0	4
10	2622.02	11.14	1	1	0	2
11	2240.96	6.046	1	1	1	3
12	2637.85	6.017	1	1	0	2
13	2701.22	12.03	1	1	1	3
14	2720.82	11.63	1	1	1	2
15	2722.88	11.62	1	1	1	2
16	2505.69	5.807	1	1	1	0
17	2173.13	11.1	1	1	0	3
18	134.62	3.858	0	0	1	3
19	1612.13	11.46	1	0	1	3
20	2474.42	11.56	1	1	1	3
21	1432.64	8.052	1	1	1	1
22	104.71	2.181	0	1	0	0
23	127.94	2.761	0	1	1	2
24	2485.12	12.24	1	1	0	3
25	2220.39	5.54	1	1	0	2
26	2049.84	3.996	1	1	0	2
27	2562.98	5.549	1	1	0	3
29	2487.65	12.1	1	1	0	3
30	2703.86	12.14	1	1	0	4
31	2627.38	12.23	1	1	0	4
32	2242.54	6.408	1	1	0	3
33	143.96	7.667	0	0	1	2
34	2743.04	16.01	1	1	1	3
35	3016.95	14.17	1	1	1	3
36	3148.33	15.03	1	1	1	3
37	311.17	1.555	1	1	0	0
39	1262.82	4.465	1	1	0	2
40	2690.83	14.94	1	1	1	3
41	2555.72	11.35	1	1	0	2
42	2624	11.4	1	1	0	3
43	2703.61	10.32	1	1	0	3
44	3754.83	23.2	1	1	1	3