NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

GRAPHICAL USER INTERFACE TO FACILITATE UNDERSTANDING OF ENCRYPTION ALGORITHMS

by

Elizabeth E. Huntoon

December 2018

Thesis Advisor:	Paul C. Clark
Co-Advisor:	Alan B. Shaffer

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE GRAPHICAL USER INTERFACE TO FACILITATE UNDERSTANDING OF ENCRYPTION ALGORITHMS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Elizabeth E. Huntoon			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The increasing complexity and contentiousness of the cyber domain require a well-educated Navy cyber force to defend its networks and key terrain. This demanding operational environment necessitates efficient and effective cybersecurity training and education, which can be maximized by leveraging newer methods and technologies. Cryptography is an essential element of cybersecurity, and is taught in the Naval Postgraduate School computer science curriculum through various methods, including a lab activity exploring symmetric encryption algorithms to reinforce critical concepts. This lab has heretofore been conducted using an older command line interface (CLI) style of computer interaction, requiring students to enter long commands into the CLI, which was prone to human error. This tended to shift students' focus more on command entry and less on the critical learning objectives of the lab activity. This thesis work has designed, developed, and implemented a web-based graphical user interface (GUI) application for use in completion of the symmetric cryptography lab activity, to increase learner comprehension and understanding of the fundamentals of encryption. The application was developed using principles of human-computer interaction and design, and was subsequently tested against the legacy system, showing that the GUI application performed equally as well for pedagogical purposes.

14. SUBJECT TERMS encryption, learning, testing, graphical user interface			**15. NUMBER OF PAGES** 83
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

GRAPHICAL USER INTERFACE TO FACILITATE UNDERSTANDING OF ENCRYPTION ALGORITHMS

Elizabeth E. Huntoon
Lieutenant Commander, United States Navy
BS, Pennsylvania State University, 2007
MS, Duquesne University, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2018

Approved by: Paul C. Clark
                 Advisor

                 Alan B. Shaffer
                 Co-Advisor

                 Peter J. Denning
                 Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The increasing complexity and contentiousness of the cyber domain require a well-educated Navy cyber force to defend its networks and key terrain. This demanding operational environment necessitates efficient and effective cybersecurity training and education, which can be maximized by leveraging newer methods and technologies. Cryptography is an essential element of cybersecurity, and is taught in the Naval Postgraduate School computer science curriculum through various methods, including a lab activity exploring symmetric encryption algorithms to reinforce critical concepts. This lab has heretofore been conducted using an older command line interface (CLI) style of computer interaction, requiring students to enter long commands into the CLI, which was prone to human error. This tended to shift students' focus more on command entry and less on the critical learning objectives of the lab activity. This thesis work has designed, developed, and implemented a web-based graphical user interface (GUI) application for use in completion of the symmetric cryptography lab activity, to increase learner comprehension and understanding of the fundamentals of encryption. The application was developed using principles of human-computer interaction and design, and was subsequently tested against the legacy system, showing that the GUI application performed equally as well for pedagogical purposes.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CLI	Command Line Interface
DES	Data Encryption Standard
ECB	Electronic Codebook
FIPS	Federal Information Processing Standards
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IRB	Institutional Review Board
NIST	National Institute of Standards and Technology
NPS	Naval Postgraduate School
OFB	Output Feedback
WWW	World Wide Web

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

A.   DISCUSSION

In the Naval Postgraduate School (NPS) course CS3600, Introduction to Computer Security, students have a lab assignment that focuses on symmetric encryption algorithms. This lab assignment requires students to use a Linux tool called OpenSSL, utilized via a command line interface (CLI). Students must type in long strings of commands, minimally 20 characters in length, and a single mistyped character will cause the encryption to fail, requiring the student to start over. Many students become so focused on the act of entering the correct strings of characters that they lose perspective on the purpose of the exercise and its learning objectives. This research was undertaken to design a graphical user interface (GUI) that alleviates this issue by providing students a concise interactive environment that clarifies and enforces the learning objectives of the lab assignment.

B.   AREA OF RESEARCH

Developing a GUI to replace the CLI on Linux systems.

C.   RESEARCH QUESTIONS

1.   Is it possible to create a GUI to replace CLI for exploring encryption modes?

2.   Does a GUI enhance student learning of concepts of encryption modes and their properties?

D.   SCOPE OF THE THESIS

The main focus of the study was on how to create and implement a GUI to potentially replace the CLI currently utilized during the "Exploring Symmetric Key Encryption Modes" lab in the CS3600 course. After the GUI was designed and implemented, its effectiveness was compared to the CLI through testing of human participants who were students of the class completing the lab assignment.

E.   METHODOLOGY

The study included the evaluation of suitable programming languages for implementing a GUI. After the ideal implementation was determined, the encryption tool was designed, developed, and tested to meet all requirements of the published lab. New lab instructions were created to reflect the new procedures using the GUI.

To evaluate the effectiveness of the GUI, two groups of test subjects completed the symmetric encryption lab using either the GUI or CLI. Following completion of the lab assignment, they were asked to fill out a post-test. The differences in scores between the two groups of users were used to gauge the effectiveness of the GUI versus the CLI for enhanced understanding of the material.

Although CS3600 students are required to complete this lab assignment as part of the course, participation in the post-test was voluntary. There were two sections of CS3600 which provided an ideal testing set up, one completing the lab with each method. Most students in both classes chose to participate.

F.   THESIS STRUCTURE

Chapter II provides a background description of AES encryption, and the operation of and differences between the four block encryption modes used during the lab. The chapter also covers basic tenants of human-computer interaction, specifically concerning systems used for learning, and design development processes. An evaluation of prior work discusses several observations of CLI and GUI in various environments.

Chapter III summarizes the development process of the design for the encryption tool using user analysis, task analysis, conceptual design, functional design, scenario design, and visual design. The chapter also includes storyboards and prototypes used in testing the design, and an overview of the decisions made and implementation of the final design.

Chapter IV discusses testing of the encryption tool by CS3600 students. It includes an analysis of the results of the human testing and conclusions drawn from these results.

Finally, Chapter V discusses conclusions on the utility of the encryption tool based on the human testing results, and makes suggestions for improvements to the GUI tool and future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II.   BACKGROUND

A.   ENCRYPTION

Proliferation of technology, and ubiquity of computers and automated systems have necessitated the implementation of security mechanisms to protect the massive amounts of information being used, stored, and transferred every day across the globe. Similarly, increasing computing power, as theorized by Moore's Law, necessitates that these security measures become increasingly complex to ensure that brute force intrusions and data exploitation remain impracticable. To meet these needs, encryption provides a method of encoding information for storage and transmission to ensure that only those in possession of a secret key can obtain the information. Methods of encryption have developed in complexity over many centuries to create the computer enabled algorithms in use today, namely the Advanced Encryption Standard (AES).

1.   Advanced Encryption Standard

The National Institute of Standards and Technology (NIST) is responsible for establishing standards for information processing that are disseminated and utilized by United States government entities [1]. Developed by IBM and adopted by NIST in 1977, the Data Encryption Standard (DES) was the first nationally recognized federal encryption standard. However, by 1997 DES, which used a 56-bit key, could be cracked in a few hours by specialized machines, and had become obsolete for use in secure government systems [2]. NIST sought to replace DES with an algorithm that could be used well into the next century, and in 1997 commenced a worldwide development process that resulted in the new Advanced Encryption Standard (AES) [3]. NIST specified that the AES algorithm must be capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits [1]. After four years of research, evaluation, conferences, and comments the Rijndael algorithm, developed by Doctors Joan Daemen and Vincent Rijmen of Belgium, was chosen as the AES encryption scheme [2]. AES was announced by Federal Information Processing Standards (FIPS) Special Publication 197 [1], which outlined its operation and implementation. The AES algorithm uses iterative rounds of

substitution, shifting, and mixing transformation operations, in conjunction with an input key, to provide the diffusion and confusion necessary to produce a secure ciphertext from a plaintext message. Because every element of the algorithm is public, per Kerckhoff's principle [3], it is essential to the security of AES that the keys remain both secret and large enough to make brute force cracking impractical.

2.   Block Encryption Modes

Symmetric block encryption algorithms, such as AES, can be implemented using a variety of modes, each providing various benefits and security considerations. The main considerations for the implementation of a certain cipher mode are security, efficiency, and fault-tolerance. For security, this involves making it as difficult as possible to analyze patterns and manipulate the deciphered text, while allowing encryption of more than one message with the same key. Efficiency can be equated to speed. The speed of the mode should be approximately equivalent to that of the cipher, while efficiency can be increased with parallelization and error recovery. Additionally, fault-tolerance has implications for recovery and preprocessing.

Some modes may require more processing overhead with the input plaintext, input key, or both. Block cipher modes require all inputs to be a multiple of the size of the block and thus introduce the need for padding when the input plaintext is not a multiple of the block size. Padding is most often achieved through a scheme that appends extra bits to the trailing end of the data with an indication that padding was added, such as the scheme specified in the NIST Special Publication (SP) 800-38A [4]. Stream ciphers, however, work on a smaller scale, usually bit-by-bit and do not require padding. Most modes also require an additional input data block called the initialization vector (IV). The IV is used to introduce randomness to the initial steps of encryption and decryption. NIST dictates that the IV need not be secret; however, it must be unpredictable.

Several modes have been developed for implementation with symmetric encryption algorithms. The modes authorized for use with AES are defined in NIST SP 800-38A [4] as Electronic Codebook, Cipher Block Chaining, Cipher Feedback, Output Feedback, and

Counter. The first four are discussed here, as they relate to the learning objectives for CS3600 and the lab created for the course.

a. Electronic Codebook (ECB)

ECB mode is as an example of the most straightforward application of encryption as shown in Figure 1. As described by Schneier [5], "In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of output blocks is the ciphertext." Similarly, for decryption the inverse cipher function is applied directly to the blocks of ciphertext to produce the plaintext. This direct application makes ECB quick, and the independent application to each block makes it possible to process several blocks in parallel or non-sequentially, making it a good choice for files that are accessed randomly, such as databases. However, the simplicity of ECB also contributes to its greatest security weakness.

Electronic Codebook (ECB) mode encryption
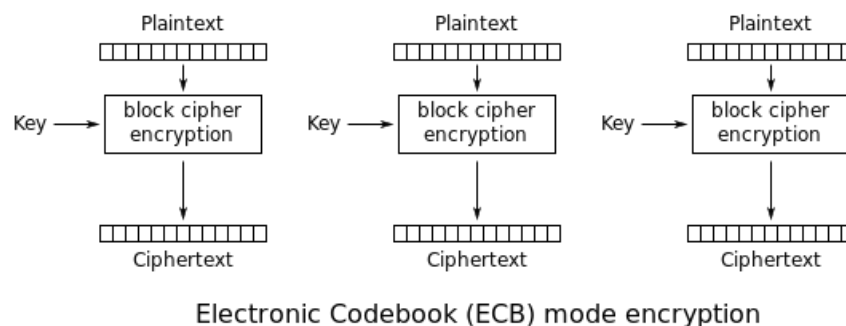
Figure 1.   Electronic Codebook (ECB) mode encryption. Source: [6].

Repeated identical plaintext blocks encrypted with the same key will produce the same result, which makes it possible to cryptanalyze ciphertext messages and distinguish patterns. Many messages and files have repeated patterns (headers, addressing information) that are identical. These patterns in a message can be analyzed, and with information about the plaintext, could be organized into a codebook, essentially making the encryption obsolete. ECB can be appropriate for short, random messages, such as ones that

disseminate keys, where the chances of redundancy are low. Longer messages, however, increase the likelihood of pattern recognition that can render the encryption ineffective.

ECB mode has better performance because it has the least amount of overhead, and it is also parallelizable. However, the ciphertext can be up to one block longer due to padding. ECB mode has some issues with fault-tolerance, namely that bit errors in the ciphertext will affect that entire block after decryption, and the addition of bits to the ciphertext will cause a fatal synchronization error as the blocks will not line up during decryption.

b. Cipher Block Chaining (CBC)

CBC improves on ECB by introducing randomness to the cipher input of each block so that patterns in the plaintext will not be revealed in the ciphertext. Randomization is initially introduced before the beginning of the cipher by combining an IV with the first plaintext block. Pattern recognition is further obscured by ensuring that even identical plaintext blocks are encrypted to produce different ciphertext. According to NIST SP 800-38A [4] this is accomplished by "combining ('chaining') the plaintext blocks with the previous ciphertext blocks." Decryption is accomplished in the same manner as encryption. CBC mode is still the same speed as the block cipher, however, because each ciphertext block is dependent on the one before it, encryption is not parallelizable. Figure 2 illustrates this dependence of each block on the one before it.

Cipher Block Chaining (CBC) mode encryption
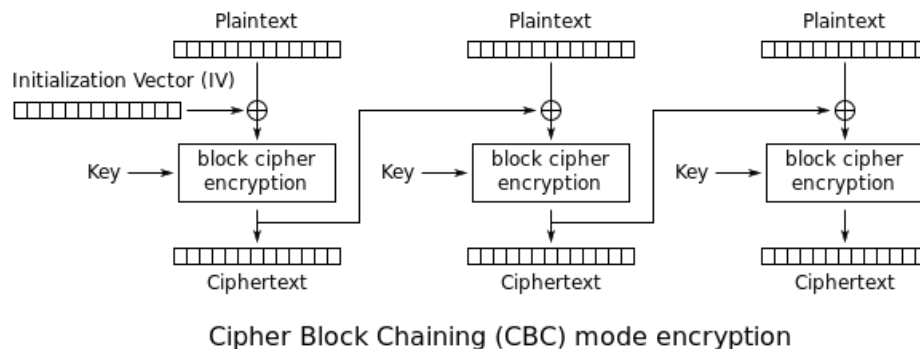
Figure 2.   Cipher Block Chaining (CBC) mode encryption. Source: [6].

Like ECB, CBC mode can also require padding, which causes the ciphertext message to be slightly longer than the plaintext. Encryption is often used to secure transmitted data. Because transmission increases the possibility of errors, error recovery is also considered a part of efficiency for encryption modes. CBC mode is largely recoverable as Schneier [5] notes: "A single-bit error in the ciphertext affects one block and one bit of the recovered plaintext. The block containing the error is completely garbled. The subsequent block has a 1-bit error in the same position as the error." Because it is also a block encryption, synchronization errors are still unrecoverable as with ECB mode, though these are rare. The added security and ease of implementation makes CBC ideal for most types of file encryption.

c.   Cipher Feedback Mode (CFB)

Unlike the previous modes, CFB can be implemented as a self-synchronizing stream cipher. This is accomplished by encrypting data segments of size s that are no larger than the block size. As defined by NIST [4], "The first input block is the IV, and the forward cipher operation is applied to the IV to produce the first output block. The first ciphertext segment is produced by XORing the first plaintext segment with the s most significant bits of the first output block." The remaining bits of the output block are discarded. The input to the cipher is then updated, the s most significant bits are discarded, and the s bits of ciphertext (output of the encryption XORed with the plaintext) are concatenated to the remaining bits. This can be thought of as input register. The register is updated with each subsequent ciphertext, which is illustrated in Figure 3. CFB decryption is accomplished in the same way, with the IV as the initial input and each successive input dependent on the previous output.

Initialization Vector (IV)

Cipher Feedback (CFB) mode encryption
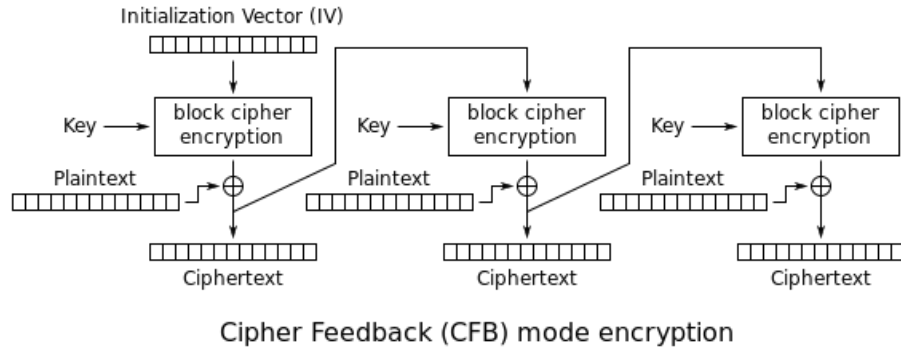
Figure 3.   Cipher Feedback (CFB) mode encryption. Source: [6].

Much like CBC mode, CFB conceals patterns in the plaintext and randomizes input. Keys can be reused for subsequent data encryptions, as long as a new IV is used. The speed of CFB is the same as the block cipher, but the ciphertext will be the same size as the plaintext, since with smaller data segments there is no need for padding. As with CBC, subsequent blocks are dependent on their predecessors, so encryption is not parallelizable. The use of the input register causes interesting results to errors. Schneier [5] notes, "The first effect of a single-bit error in the ciphertext is to cause a single error in the plaintext. After that, the error enters the shift register, where it causes ciphertext to be garbled until it falls off the other end of the register." Similarly, CFB will self-recover from synchronization errors as soon as the error leaves the shift register. CFB mode can be used for encrypting files, but it can also be used in network applications where input must be encrypted individually as it is transmitted.

d.   *Output Feedback Mode (OFB)*

A block cipher can also be implemented as a synchronous stream cipher using OFB mode. According to NIST [4] this is accomplished through "iteration of the forward cipher on an IV to generate a sequence of output blocks that are XORed with the plaintext to produce the ciphertext, and vice versa." Similar to CFB, in OFB the IV is the initial input, is processed in data segments smaller than the block size, and the input register is updated sequentially. By contrast, in OFB the output of the encryption, not the ciphertext, is used to update the input as shown in Figure 4. Because subsequent inputs are not dependent on

the ciphertext output, it is possible to preprocess this mode in order to create a keystream before data is introduced.

Output Feedback (OFB) mode encryption
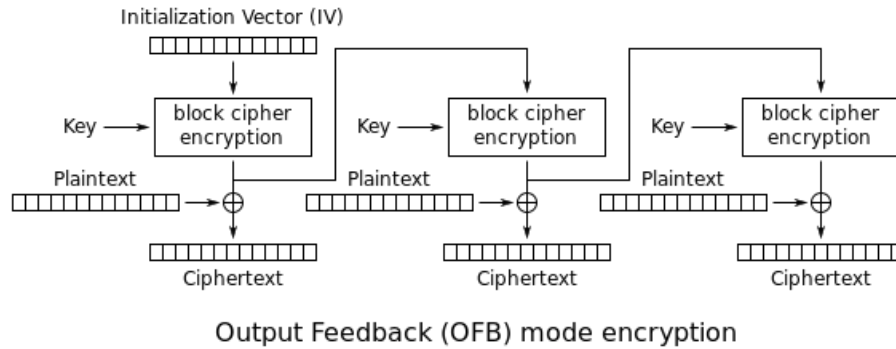
Figure 4.   Output Feedback (OFB) mode encryption. Source: [6].

Like other modes using IV inputs, OFB conceals plaintext patterns and randomizes the input, so keys can also be reused for subsequent encryptions, as long as a different IV is used each time. OFB is just as fast as the block cipher, the cipher and plaintexts are the same size, and preprocessing provides extra efficiency. Another benefit of the input not depending on the ciphertext is that OFB is extremely fault tolerant for bit changes, thus a ciphertext bit error only effects the corresponding bit in the decrypted plaintext. However, a synchronization error will be fatal. Because the key stream is generated without the ciphertext, should the key and ciphertext get out of synch at the receiver it is impossible to recover and continue decryption. Schneier [5] explains, "OFB is most often used in high-speed synchronous systems where error propagation is intolerable. OFB is also the mode of choice if pre-processing is required. OFB is the mode of choice in an error-prone environment, because it has no error extension."

B.   HUMAN-COMPUTER INTERACTION

This thesis involved developing a GUI program to be used by students in the completion of a lab activity on symmetric encryption modes. The design and development of this program necessitated integration of basic principles of human-computer interaction

11

to facilitate learning during the activity. The functionality of the system was well defined by the requirements of the activity, however, usability and learnability were also important considerations.

1.   Usability

Users are varied and complex and because of this there are numerous considerations during design and implementation of a system. These elements have been closely reviewed by many organizations in studies devoted exclusively to optimizing human-computer interaction. The International Organization for Standardization (ISO) is an international organization of professionals from a variety of fields that strives to create standards to maximize quality, efficiency, and safety. The ISO Publicly Available Specifications (PAS) on ergonomics of human-system interaction [7] codified several of the essential elements of human-computer interaction during development, including life cycle involvement, integration of human factors, and human-centered design. These basic considerations ensure that systems reflect the needs of the people who will utilize the system, properly execute the human-system processes, and ensure *usability* of these systems by incorporating data and processes into the system implementation. While these are valid considerations, they are high level concepts that must be executed at lower levels of design.

To be successful beyond simple consideration of basic human requirements, the system must work properly and be a positive experience for the user. Ritter et al. [8] argue that while functionality is the first consideration, the usability of a system is often an afterthought for many designers when it should be just as important. They state, "Usability issues need to be considered with respect to the task that is being carried out and the human operator's capabilities." A system can be functional and can complete all specified requirements, however, if it is beyond the comprehension or abilities of the user it will not be successfully implemented. A system that is too complex, with too many options or steps for completion, will confuse and frustrate the user. There needs to be a balance between ease of use and completeness of function. Ideally these goals support one another, as a more functional system should be more usable and easier to learn.

2. Learnability

Of specific concern is how people not only use the system but learn it and learn from it. Ritter et al. [8] say that "*Learnability* is how easy the system is to learn." Systems that are less complex and more easily recognizable to the user are easier to learn. Systems that utilize graphics and process flow similar to other practiced systems will be understood and adapted to more quickly than novel systems. Additionally, the user's ability to learn and use a system can be demonstrated by how well the system engages their memory and attention. Memory can be defined in many ways, but for learning and interaction with systems, designers consider memory to refer to recognition and recall.

The easiest way to understand the difference between recognition and recall is the difference between multiple choice and fill-in-the-blank questions on a test [9]. Recognition memory is analogous to multiple choice selection, and is easier to develop and lasts longer than recall. Therefore, for the usability and learnability of a system it is generally better to utilize prompts or options that will be easily recognized by the user. By limiting the amount that the user must remember, the system becomes easier to learn and use. Similarly, limiting strains on the user's attention, or "the number of items that can be processed at the same time," [8] will have a positive impact on how easily the system will be understood. Overwhelming the users with text, graphics, or options strains their cognitive capacity, and may obscure which elements are most critical and inhibit learning. Interfaces that have fewer objects allow the user to focus their attention, aiding learning of the system and allowing them to use it more effectively. Therefore, an effective system must capture and maintain the user's attention, and must be easy to remember. If it has too many steps or choices, user manuals or other aids become necessary.

C. SOFTWARE DEVELOPMENT PROCESS

Software development is a relatively new process, compared to other more traditional industries. As described in the *Universal Principles of Design* [9], most software products go through several stages of development, namely requirements, design, development, and testing. The different models of development approach these stages with different priorities on completion and timing.

1. Linear Models

The first model to define a disciplined approach to software development was the Waterfall method, which defines each stage of development as a phase, where one phase must be completed before the next can begin. As described by Westfall in the *Certified Software Quality Engineer Handbook* [10], "a well-defined set of requirements is specified before design begins, design is complete before coding begins, and the product is tested after it is built." There may be as many steps to the Waterfall process as the designers desire, though it will generally start with the identification of a problem or opportunity that necessitates a new system. Because each phase is completed before the next can begin, phases cannot go back, requirements must be defined at the outset of the project, and there is little leeway for the addition of functionality later in the process. The Waterfall model is best suited for well-defined problems that have little chance of changing between requirements definition and implementation. Any changes to requirements will necessitate a new project after the completion of the original.

The V- and W-models are variations on the Waterfall model. As defined by Westfall [10], the V-model, "highlights the relationship between the testing phases and the products produced in the early life cycle phases." The inclusion of testing for each phase of the planning process allows verification that all elements of the design are included and correctly implemented. Westfall goes on to describe the W-model as two crossing Vs, representing the phases and testing for the development organization, and those of an "independent verification and validation organization that is responsible for independently analyzing, reviewing, and testing the work products of each phase of the software cycle." This type of testing and verification can be especially crucial in systems where failures could be catastrophic or fatal. Even with the addition of progressive testing, linear models are best suited for static problems with rigid requirements.

2. Iterative Models

As its name suggests, iterative models have steps that are repeated several times, allowing refinement of requirements, design, code, tests and other details. There are many varieties of iterative models—also considered agile development or prototyping—

including spiral, test-driven, feature-driven, and rapid application development (RAD). Iterative prototyping supports an interleaved rather than a linear form of project development. As Boyle [11] notes, "prototyping allows the team to visualize and evaluate system possibilities." Using this method, the system is built a little at a time, distributing the design, code writing, and testing throughout the various iterations. The various iterative models are based on how the iterations are organized, usually to pass various test cases, to introduce a new element of functionality, or to build subsets of the overall design sequentially or quickly in parallel. According to Lidwell et al. [9], the iterative process is preferred in cases where the requirements are more dynamic or less exact. This allows functionality to be added throughout the development process, though it can also result in scope creep. To avoid the addition of unnecessary features it is essential to test and evaluate the system throughout development to ensure the user's requirements are met without overwhelming the project with unnecessary features.

3.   Testing

All development models stress the importance of testing at some point, though ideally more than once, during the process. Per Boyle [11], software testing can be broken down into two categories: formative, which is conducted during development; and summative, which is completed at the end of a project. The results of formative testing are used to guide further development, identifying gaps, inconsistencies, errors, and other issues that will need to be corrected before completion, but allowing incremental progress. Conversely, summative testing is a final review of the completed system. Both types of testing are meant to evaluate the effectiveness and usability of a system, and if it achieves the specified objectives and is easy to use. While it can be relatively simple to determine whether a system meets functionality requirements, usability can be more ambiguous. Testers may have trouble describing exactly how a system does or does not meet their usability expectations. As pointed out by Ritter et al. [8], "Ravden et al. specify the following usability criteria: visual clarity, consistency, informative feedback, explicitness, appropriate functionality, flexibility and control, error prevention and control, and user guidance and support." These criteria, once identified, can provide guidance for testers to

evaluate usability. It is far easier to ask for feedback on error messages than expect users to identify the need for a message.

D.   PREVIOUS RELATED WORK

Though there appears to be no work specifically assessing learner concept comprehension comparing CLI and GUI, some general statements can be made about human-computer interaction, as well as the usability and learnability of different interfaces. Some previous studies [12], [13] have made observations on the comparison of use and utilization, and concluded that, in general, a GUI is better utilized by beginners and a CLI for experts.

It is now possible for users to interact with computers and computer systems in a variety of ways, including keyboards, mice, microphones, and touchscreens. However, as noted by Ritter et al. [8], for input, typing is still the most common method of interaction. Similarly, most output to the user is transferred via text, and even when using GUIs a majority of computer feedback is text. Nielsen's [14] critique of the Mac interface evaluates the advantages and limitations of computer interaction exclusively by text. He points out that text and language provide a rich, understandable, and repeatable syntax. Commands can be combined and executed to produce specific and powerful results. However, CLIs are limited by what text the computer can understand, and the user must know and adhere to the commands of the operating system. Most CLIs are not robust enough to correct errors in spelling or grammar. He goes on to emphasize that the CLI was optimal for the user when it was created: "The command-line interface was suited for a user community dominated by programmers who had the expertise, interest, and time to understand how the many different system features could be combined for optimal use through mechanisms like piping and shell scripts." Before the development of GUI, CLI was the only option for all users who were forced to memorize commands. As Ritter et al. [8] point out, novices will require systems that have easily recognizable actions like those provided by a GUI, while experts will prefer to recall commands and keystrokes without needing to consult or "wade through" choices. As beginners become experts they will begin to learn these shortcuts and optimize efficiency.

Sivakumar et al. [12] conducted a study examining students in a distributed environment interacting with a lab via network connections, where a CLI was chosen over a GUI for relative benefits to students' learning. The environment necessitated simplicity and speed of transmission where CLI shows an obvious advantage. Where remote student interaction with a GUI was necessary, it involved a much larger bandwidth capacity. Students were conducting labs for an internetworking class and conducted all communications with virtual networks by CLI. "Since all options and operations are invoked in a consistent manner, the CLI allows greater flexibility and control and is, therefore, easier to learn and use." The course developers chose to use a CLI for lab completion by distance learners, while resident students used a combination of GUI and CLI. Surveys were conducted comparing the online experience to that of the in-person laboratory. The CLI-only remote lab was considered easier and more flexible to use, while the combination lab was deemed more accessible, interactive, and state-of-the-art. Additionally, the educators noted that there was no degradation to the learning outcomes between the two interfaces. While this study did not measure and compare the use of CLI and GUI in a learning environment, it demonstrated the considerations of educators and designers.

In a study comparing the use of CLI and GUI for implementations of the R program by ecologists, Thioulouse and Dray [13] noted: "GUIs make the learning phase smoother for beginners, and can be used in education to introduce students to CLI mode." They also observed that a GUI is better utilized than a CLI for occasional users, as many in this study were. However, there were notable disadvantages in the GUI, for example users had more difficulty remembering actions that led to a result, while a CLI can hold a history of commands. They also observed that GUIs are made for "personal and instant use." GUI programs run better on newer systems and may perform worse over distributed systems. Finally, a CLI offered advantages in repeated operations such as scripting and allowing sharing of code. While this study was limited in scope and scale, the observations can clearly be broadened to other application interfaces.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PROGRAM DEVELOPMENT

The initial design and evaluation of the GUI for the CS3600 Exploring Symmetric Key Encryption Modes lab were completed as part of course work for CS3004, Human-Computer Systems Interaction. The quarter-long project required students to design a system of their choice, with the restriction that it had to solve a problem (i.e., it could not be a game for fun), and it had to help accommodate three user tasks related to solving the stated problem. The three tasks as required by the CS3600 lab assignment are to encrypt a file, decrypt that file, and edit a file in hexadecimal format. Though the hexadecimal editing feature was superfluous to the end product it is included for completeness of the design process. The main concept behind the development process was user experience (UX) design, which is an iterative process with multiple levels of analysis, prototyping, and formative assessments. The users and tasks were thoroughly analyzed on several levels to create requirements for inclusion into the design. These requirements were incorporated into a low fidelity model presented in a printed format that was tested for completeness and usability. From that assessment a high-fidelity model was created and tested, and then feedback from this testing was incorporated into the final design. The steps and detailed results of this UX design process are described more fully in the following subsections.

A. USER ANALYSIS

The first task was to analyze the user. Who would be using the system? How will they use the system? What are their abilities and constraints? Answering these questions about the potential users allowed the design of the system to be limited in scope to the necessary functions while ensuring that all essential elements and considerations were included.

(1) User Characteristics

The primary user of the system is a CS3600 student completing the Exploring Symmetric Key Encryption Modes lab. Students of CS3600 may be computer science or cyber systems students, or may be from another focus area taking the class to satisfy a cyber elective requirement. The student will use the system to execute encryption and

decryption of files using various modes of AES symmetric key encryption for educational purposes. The student will have an indeterminate amount of prior knowledge of the subject matter. The student will use the system infrequently as the lab is a one-time requirement for the course.

(2) User Skills

A baseline general computer skill level can be assumed for all students who use the system. Prerequisites include the ability to interact with a GUI by using a cursor and keyboard, make selections via a drop-down menu or radio button, select and type in various boxes, make selections from a hierarchical list (i.e., files from a drive), open and close files, and possess general competency and knowledge of their operating system. No formal training should be necessary to operate the system.

(3) Conclusions

Students using the system are part of a military institution, and thus it can be assumed that there will be no extenuating physical or mental conditions that will affect the design. There should be no need for audio or haptic responses, or special interfaces to facilitate operation. The user should not require any special skill or instruction to interact with the system, but the system should provide meaningful error messages, for example, possible errors might include incorrect key or IV length. Ideally user-induced errors would be prevented through design, however, a message must convey the exact error, and suggest methods of correction.

B. TASK ANALYSIS

The task analysis was conducted using hierarchical task analysis. This type of analysis focuses on physical or observable actions starting with a user goal and examining each of the tasks required by the user to achieve that goal [15]. A hierarchical task analysis analyzes each task by its goal, preconditions, subtasks, exceptions, and other relevant features. By going into such depth, the next step of the design was informed on the essential elements and necessary features and feedback. The tasks were identified to fulfil

assignment requirements for the class during which the initial design took place; this includes features that were not incorporated into the final design.

(1)   Task 1: Encrypting/Decrypting a file

The goal of this task is to transform a file using an encryption algorithm either from a plaintext to ciphertext or ciphertext to plaintext. The precondition for beginning this process is that the user must have an input file created or downloaded. This goal was broken down into several subtasks: select input file, create output file, select encryption or decryption, select mode, provide key, determine if IV is needed, and provide IV if needed. The following exceptions, or situations that would lead to an undesired result, were identified: choosing the wrong file as input, overwriting an existing file with the output, and choosing an inappropriate key or IV based on length or language. In addition to these basic steps, it was also determined that there needs to be a way to clear selections to start over because the action will need to be repeated several times with different modes and files.

(2)   Task 2: View a file

The second task's goal is to view an opened file. The precondition for this goal is that encryption or decryption has taken place, otherwise the user could only view the original file. The subtasks for this goal are to choose, open and close the file. The only identified exception for this goal is the user selecting a non-viewable file due to an issue or error during the encryption or decryption process. The other relevant feature of this task is to provide a method to compare two versions of a file, e.g., before and after encrypting, to allow the user to identify anomalies or errors.

(3)   Task 3: Edit a file

The final task's goal is to view a hexadecimal representation of an encrypted file to allow it to be edited. For the learning objectives of the lab assignment, it is necessary for the file to be encrypted as a precondition. The following subtasks were identified: choose a file, open the file, change specified bits in the file, save the changed file with a new name, and close the file. This element of the exercise is intended to show the student what happens

when bit errors are induced in the ciphertext, so an exception should occur if the user fails to change the filename of the modified file to prevent overwriting the initial ciphertext. The modified ciphertext (i.e., the purposefully corrupted ciphertext) would need to be decrypted, so another relevant feature of the task is that the new file needs to be in a format that can be decrypted.

C. CONCEPTUAL DESIGN

Conceptual Design is a method of modeling how the user thinks about a task or activity, which informs the visual design at a later stage of the design process [16]. To conceptualize the tasks and activities of the user, the methods of achieving the user's goal were broken down into the objects they will use, the attributes of these objects, the relationships between the objects, and actions upon the objects, attributes, or relationships.

(1) Objects

To achieve the goals of encrypting, decrypting, and editing files, there are four objects that need to be considered by the user: file, mode, key, and IV. These are the most basic elements of the task.

(2) Attributes

Each of the objects is made up of several attributes that may be static or dynamic. A file has attributes of size, location, and type (i.e., text, image, etc.). The mode has one attribute, representing one of the following block encryption modes: ECB, CBC, CFB, OFB. The key and IV each have attributes for language (ASCII or hexadecimal) and length in bits.

(3) Relationships

There is a relationship between the mode, key, and IV because they are combined to act upon the file and create a new file. There is also a different relationship between the mode and IV, such that ECB mode does not require an IV, while all other modes do require an IV.

(4)   Actions

Actions that can be performed on objects were identified as: open a file, create a new file, create a key, create an IV, and delete all object values. The actions performed on attributes include: change mode type, change key length or language, and change IV length or language. There are limited actions possible on relationships; determining whether an IV is necessary for the selected encryption mode may change the relationships of the elements in the encryption process. For example, choosing the ECB mode removes the necessity of an IV object, and thus changes the relationship of objects needed to create a file.

D.   FUNCTIONAL DESIGN

Once the users and tasks were established and understood, the next step was to perform a functional analysis of the system to identify necessary functions to be incorporated into the design. For each of the tasks, the functions and potential methods for their implementation were identified.

For the file encryption and decryption task the following functions and methods were identified:

- Select the input file by typing a file name or selecting a file from a displayed list in a browser.

- Create the output file by typing a new file name.

- Select the encryption or decryption option via a drop-down menu, button, or other selection method.

- Select the encryption mode by drop-down menu, button, or other selection method.

- Provide the key by typing.

- Determine if an initialization vector (IV) is needed based on the selected encryption mode, which would determine whether the selection of an IV is possible or not.

- Provide the IV by typing (when necessary).

The view file and edit file tasks had fewer functions, as shown below. The view file task consists of only the first and last functions: opening and closing.

- Open the file specified by the user.

- Change specified bits, first by selecting bits with the cursor or arrow keys, and then changing the bit(s) by typing.

- Save changed file with a new name by typing.

- Close the file.

E.   SCENARIO DESIGN

Formative assessments were required throughout the design process. To accomplish this, volunteer subjects were asked to complete a scenario using low or high-fidelity prototypes of the encryption tool created to mimic the intended design and operation of the system. The scenario was essentially the steps of the CS3600 symmetric encryption lab, requiring the user to encrypt and decrypt files in various modes. The lab also has a requirement to edit ciphertext and view the results after decrypting. To simplify the scenario for testing, the instructions were narrowed down to three basic functions:

1.   Encrypt a text file.

2.   Decrypt an encrypted text file.

3.   Edit an encrypted text file.

During scenario testing each participant was asked to talk through their actions and choices. The participants were allowed time at the beginning of the assessment to familiarize themselves with the interface.

F. VISUAL DESIGN

The focus of the visual design was a simple interface with only the essential fields and functions present. The system was designed to be easy to use and aesthetically pleasing, which would ideally make it more desirable for the user and lead to more interaction and learning. Among the various design principles utilized in the visual design, the principles of uniform connectedness and affordance were implemented to imply the order and use of the elements of the GUI. The objects on screen in the GUI were meant to seem related by virtue of uniform connectedness, the principle that the visual similarities of the various elements would allow them to be perceived as connected [9]. In addition to how the design would make the user think, the system had to be easy to learn and use, this was accomplished through affordances. Lidwell et al. [9] describe affordance as "a property in which the physical characteristics of an object or environment influence its function." In the system design, the use of boxes and buttons abided by this principle. Where an input was meant to be typed, a text box was provided; similarly, where there were a limited number of selections, radio buttons were used. The interface was designed to enhance learning and discourage errors.

G. STORYBOARDS

To achieve the three task objectives for the system, very simplistic flow was developed using tabs. Each objective had its own tab on the interface to clearly delineate a user task. Figures 5, 6, and 7 show the storyboard walkthroughs for encryption, decryption, and hex editing, respectively. As mentioned above in Scenario Design, these storyboards were used by test subjects to verify the low-fidelity prototype, to validate the design.
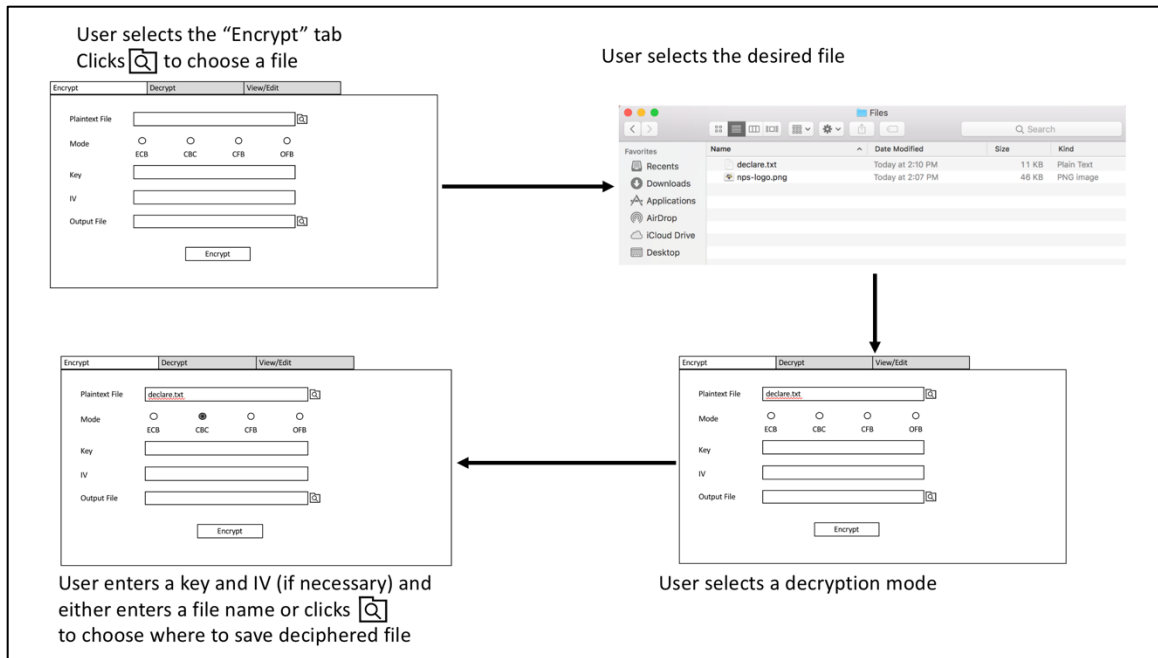
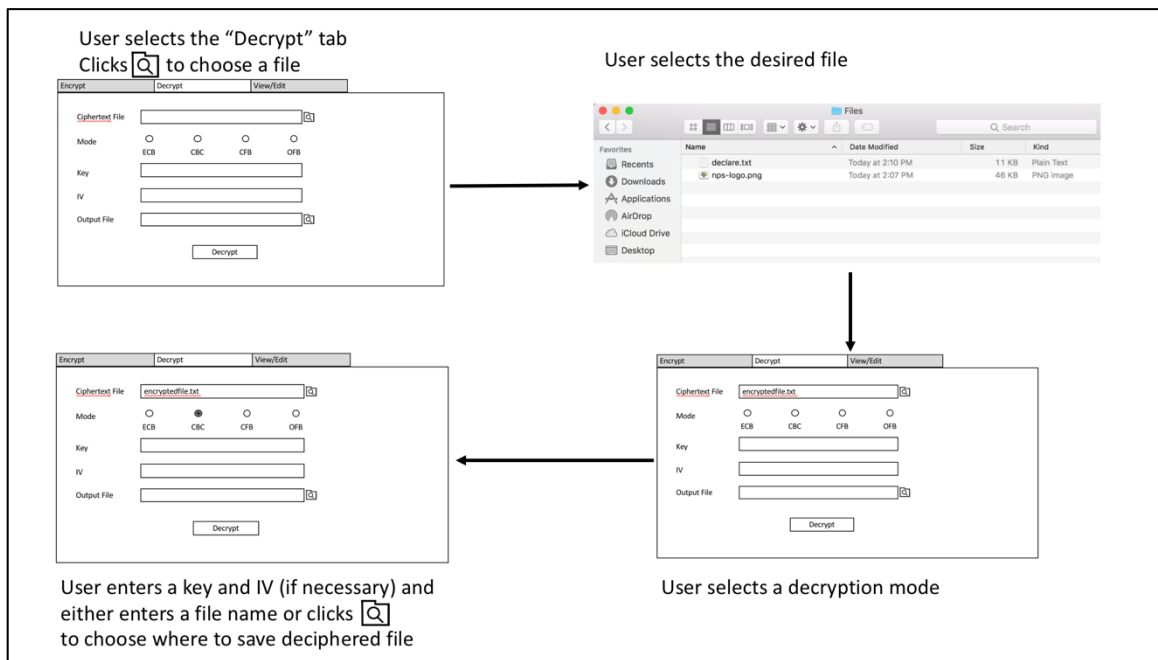Figure 5.   Storyboard to encrypt a text file.

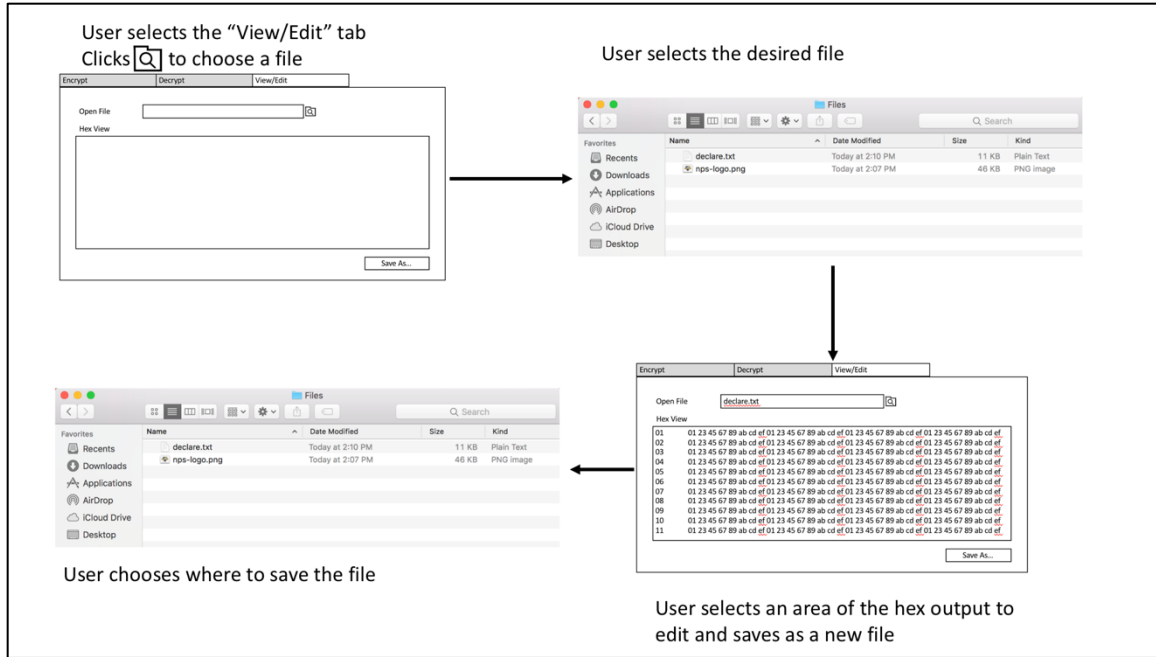Figure 6.   Storyboard to decrypt an encrypted text file.

Figure 7.   Storyboard to edit an encrypted text file.

H.   LOW-FIDELITY PROTOTYPE EVALUATION

The low fidelity prototype was created using Microsoft PowerPoint to model the visual elements of the system design. Slides were created to simulate the possible interactions between the user and the interface. These slides were printed and presented to test subjects, with the start page shown in Figure 8 used to begin their tasks. Selections were made by users pointing with a finger to simulate the click of a mouse. Participants were given time to familiarize themselves with the interface, however, only one participant chose to "click through" all the tabs before starting.

All participants were able to complete the three tasks without help, although two asked questions about the function of the IV input, because they either did not know or did not remember for what it was used. From this observation it became clear that a help function or instructions were needed for the input fields. Testing also revealed the necessity of changing the file saving method to reduce risk of error, because the tested prototype introduced the possibility of overwriting the original input text file, which would defeat the purpose and cause problems later in the interaction.

27

Figure 8.   Low fidelity prototype start page.

I.   HIGH-FIDELITY PROTOTYPE EVALUATION

The high-fidelity prototype was created using PowerPoint and implemented to simulate the process of encrypting, decrypting, and editing plaintext and ciphertext files as presented in the low fidelity prototype. In other words, rather than using printed slides on paper, as in the low-fidelity prototype, the high-fidelity prototype used an interactive PowerPoint presentation.

The high-fidelity prototype was designed to allow test users to accomplish each task with every possible combination of orders of input. The inputs required for encryption were a plaintext file, mode, key and initialization vector, and output file. Similarly, the inputs required for decryption were a ciphertext file, mode, key and initialization vector, and output file. The only input required for the view/edit functionality was a file to open. As a design choice to reduce possible errors, the encrypt and decrypt buttons were only accessible once all input elements were completed. For testing purposes, the filenames used for input and output were provided, and there was no typing functionality implemented for the encryption and decryption tasks. For any type-able option in the program, a fixed response was provided.

28

The PowerPoint prototype offered little opportunity for error checking, e.g., comparing input and output files to ensure they were different, and ensuring keys and IVs were 32 characters in length and in hexadecimal, therefore these potential user errors were covered by static responses. The view and edit functions were type-able. The high-fidelity prototype testing was done on a Windows laptop and participants interacted with the PowerPoint presentation specifically designed and organized to simulate the system GUI.

During user testing, the purpose of the system was described to the subject, i.e., it was described as a GUI to encrypt and decrypt files using the AES algorithm. The start screen of the presentation simulation is shown in Figure 9. Participants were allowed to click through the interface to familiarize themselves with the system, however, none of the participants chose to do so. The first task was to encrypt the text file "declare.txt." The second task was to decrypt the file "encryptedfile.txt." The third task was to edit three bytes of "encryptedfile.txt," and save it as a new file. Testers were observed as they completed these task, including the steps taken, the ordering of the steps, and the way they chose to navigate through the GUI. The assessment was not timed.

Figure 9.   High fidelity prototype start page.

While most testers quickly figured out they were meant to search for a file using the magnifying glass file icon, two participants tried to type in a response. This required intervention by the test administrator to explain that typing was not an option in the prototype, at which point the students asked how they were able to find the file, and were thus directed to the icon. These same two users also had questions about the different encryption modes because they were unfamiliar with AES encryption. Finally, these two users also had issues trying to click the encrypt or decrypt buttons before all inputs were completed and would have benefitted from an error message to explain what was needed to complete the request.

Two other testers had issues understanding the difference between the "Save As…" button and the "Encrypt" and "Decrypt" buttons. They felt that "Save As…" implied the action of saving and not just choosing a file, and that it should come after the "Encrypt" and "Decrypt" action buttons. They suggested renaming the "Save As…" button to more accurately reflect the actions that would result. Therefore, the button was changed to "Name File" as shown in Figure 9.

Though the prototype was developed so that any order of input could be accepted, all testers went from top to bottom to fill-in each required piece of information. Some chose to click through the different modes after they had added the plaintext or ciphertext file. All users followed the optimal, and arguably the most efficient, top to bottom order of operations. The top down order ensured all required information was present before submission and that only required portions appeared (i.e., the IV was not filled in before selecting the mode, which could possibly invalidate the need for an IV). For simplicity, some of the options for feedback on errors were removed, so the two users who had issues with this were able to easily figure it out when it was pointed out that information was missing. All users were able to quickly learn the system, such that any problems experienced with the encryption task were not repeated with the decryption task.

Each participant was asked for overall comments on the appearance and design and the functionality. The open-ended question was meant to determine if there were any possible oversights or improvements that could be addressed; specific or prompted questions could have limited the responses. There were no general comments on the overall

design; all comments were specific to functionality with which the individual user had issues. Specifically, the two users who had problems with the prototype test had suggestions for functionality of the file input and help prompts. Separately from the comments, observation of user interaction revealed additional supplementary changes, specifically: adding an explanation that to decrypt a file the same parameters (mode, key and IV) from encryption must be used; adding an explanation of hexadecimal representation on the view/edit tab; and error testing for only hexadecimal in the edit window.

J.   FINAL DESIGN

The final design and implementation were informed by feedback from prototype testing and bounded real-world performance and limitations. The end product keeps many of the elements presented in the prototypes while improving features when possible and simplifying design where applicable.

1.   Language Selection

Several implementation options were considered for the final design of the GUI. First, the Java language was considered for system implementation. However, the development process proved cumbersome and difficult. After consulting several books and online tutorials, Professor Thomas Otani was consulted, the author of several Java reference and instruction books.   He advised that the nature of the program, specifically the encryption and file saving interactions, would be more feasible if implemented as a web application. With his recommendation, the final design was completed using Django, a high-level Python web framework that utilizes both Python and Hypertext Markup Language (HTML) coding to perform complex operations with minimal code. The resultant web application was hosted on a web server, and accessible via the World Wide Web (WWW).

The HTML code was supplemented with a bootstrap design template for a cleaner, more professional appearance. Such templates are available opensource and provide HTML and Cascading Style Sheets (CSS) scripts to assist developers with rapid implementation. Encryption and decryption were executed using a Python package called

Pycryptodome, which was chosen over other Python encryption packages for its support of the AES algorithm and padding utility. The greatest challenge during programming was ensuring all input information was in the proper format for use in the Python functions because passing incorrect information to the function would cause failure. Specifically, after the file was uploaded to the server it had to be read and saved as a byte string before being passed to the encryption or decryption function. Conversely, the resultant ciphertext or plaintext had to be saved in a readable format after the encryption or decryption function was complete. The flexibility of the platform necessitated design decisions based on requirements and ease of implementation. For example, the Django framework offered myriad possibilities for data input organization but ultimately an HTML form was used for its consistent feel and simple retrieval method to the Python functions.

2.   Visual Design Decisions

The homepage, shown in Figure 10, provides an overview of the system and was kept intentionally simple. The bootstrap template for Django and HTML uses blue as the button color, so these colors were changed for the encrypt and decrypt buttons to make them distinct, where red implies encryption and green implies plaintext. The difference in color increases visual interest by drawing attention to the two possible choices, and has the potential to draw upon previous knowledge and mental models on the relationship between these colors and the functions and texts they produce. These color themes were replicated through the rest of the website. The system can be navigated from the homepage, or by using a pulldown menu at the top of each page.
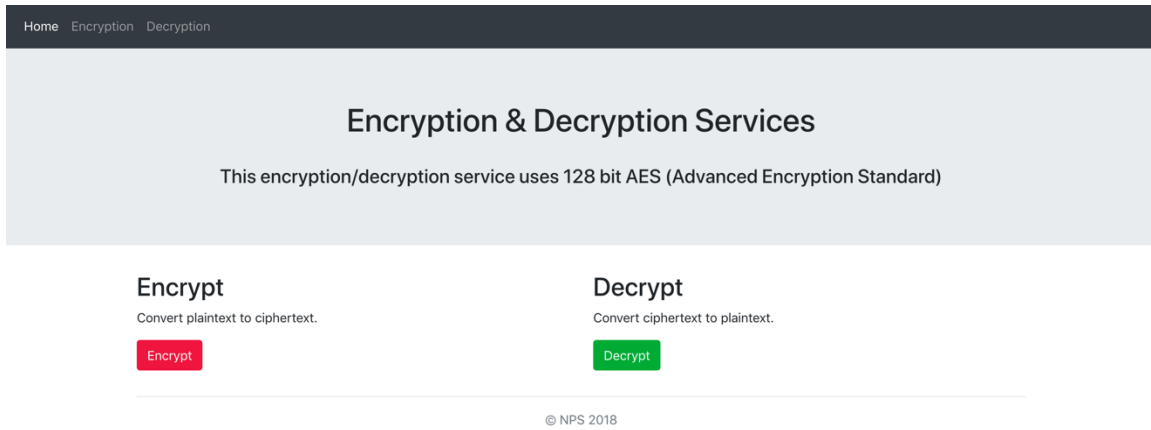
Figure 10.   Final design home page.

The encryption and decryption pages, shown in Figures 11 and 12, respectively, were intentionally designed to be similar to provide consistency and imply analogous functionality. The option to type in a filename was removed to reduce possible errors and mental load on the user, thus all files are selected using a file browser. The file icon was replaced with text to reduce confusion, based on feedback during the prototype testing. Default key and IV inputs were added to show their necessary length and format. The Python encryption package used did not necessitate hexadecimal entry so there were no restrictions on the content of input key or IV, other than length.

Figure 11.   Final design encryption page.

Figure 12.   Final design decryption page.

3. Error Prevention and Feedback

In response to feedback from testers, help messages were added for the key and IV for both the encryption and decryption pages, as shown in Figures 13, 14, 15, and 16. These brief descriptions identify and attempt to avoid unintentional mistakes from the user.
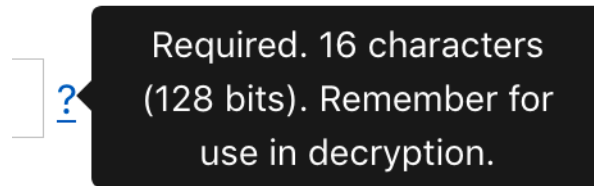
Figure 13.   Encryption key help dialogue.
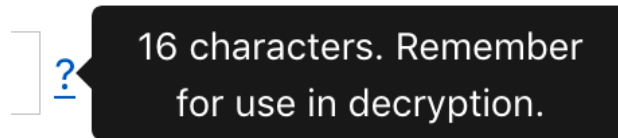
Figure 14.   Encryption IV help dialogue.
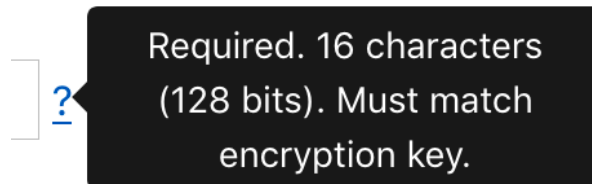
Figure 15.   Decryption key help dialogue.
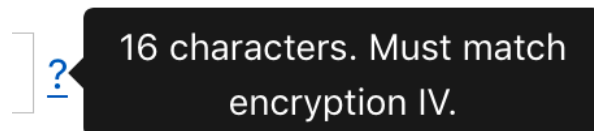
Figure 16.   Decryption IV help dialogue.

Error reduction was further accomplished by utilizing HTML specifications for entries. For example, HTML tags can be written with specific attributes such as "required," "minlength" and "maxlength." Because of these attribute specifications, error messages appear if any of the inputs are empty or unselected when the "Encrypt" or "Decrypt" button is attempted. Similarly, error messages display the required entry length for the key and IV, to ensure the input is neither too long nor too short.

Additionally, the user does not choose the name for the file that results from the encryption or decryption function. Instead, the naming convention is handled on the server by the Python "FileSystemStorage" function to ensure there are no duplicate files shown in Figure 17.
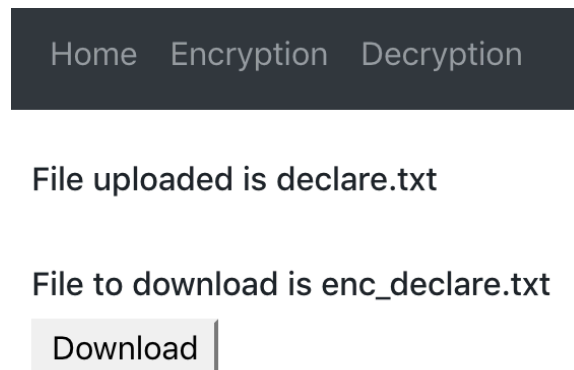
Figure 17.   Download page.

Despite every effort to eliminate errors, two issues were identified and still have not been remedied: 1) the server did not recognize filenames with spaces for download, and 2) mismatching keys and IVs between the encrypted and decrypted input cause errors on the server. These were addressed by a customized error page listing the possible problems and solutions, as shown in Figure 18.

There was an error with your encryption/decryption.

Please try again.

Possible issues:

Space character in file name

Key and/or IV do not match between encryption and decryption.

Figure 18.   Error page.

4.   Conclusion

Though designing the system as a web application presented unique challenges, the result was an application platform that could be accessed easily by all students, regardless of location or operating system. The choice of Django as the programming language presented a steep learning curve for development, but offered flexible and customizable implementation that was easy to test, and that was readily adaptable to additional and clarified requirements throughout programming.

THIS PAGE INTENTIONALLY LEFT BLANK

IV.   TESTING AND RESULTS

A.   TEST DESCRIPTION

A five-question post-test was created based on the learning objectives of the CS3600 Symmetric Encryption Lab to compare student comprehension between the CLI and GUI methods for completing the lab. The questions are shown below with correct answers bolded:

1. What kind of cipher is AES? (All that apply): Stream, **Block**, **Symmetric**, Asymmetric.

2. Which mode(s) of AES encryption require an Initialization Vector? (All that apply): ECB, **CBC**, **CFB**, **OFB**.

3. How many bits are in an AES block? (All that apply): 64, **128**, 192, 256.

4. How many bits are in an AES key? (All that apply): 64, **128**, **192**, **256**.

5. Which mode(s) use chaining or feedback during encryption or decryption? (All that apply): ECB, **CBC**, **CFB**, **OFB**.

For grading, since some questions had multiple correct answers among the four options, each option was evaluated individually, i.e., a point was earned for each option that was correctly selected or correctly left unselected, for a total of twenty possible points.

The test was designed to be taken after the completion of the Symmetric Encryption Lab. A pre-test was considered to determine a baseline of knowledge before performing the lab, however, this idea was dismissed primarily because the small number of questions left little option of variability between the pre- and post-tests. Posing the same questions before the lab would introduce priming of the subject that could affect results by changing the attention and memory used during the task. It was determined that students would be focused on finding the correct answers and not on completing the assignment as intended.

B. SUBJECT POPULATION

The test subjects were all resident NPS students taking CS3600 during the Fall 2018 quarter. No demographic data was collected on the students. However, it is known that the students came from several curricula, including computer science and cyber systems operations, and varying backgrounds, including United States military, international military, and civilian. There were two sections of CS3600 during the testing, roughly of the same size, which allowed one section to serve as the control (CLI) group, and the other to serve as the experiment (GUI) group. The testing regimen was evaluated by the NPS Institutional Review Board (IRB) and deemed exempt from human subject restrictions. During the GUI tool evaluation, testers adhered to all appropriate protocols for exempt human testing, including verbal consent and anonymization of data.

C. EXECUTION AND LIMITATIONS

Testing for both sections of CS3600 was coordinated with the instructors to take place after the completion of the section of instruction related to the Symmetric Encryption Lab. Both groups were provided with the same post-test, to be taken after they completed the lab. A verbal consent statement was read to each group, and no subjects voiced any concerns or declined to participate immediately after the statement. Students had one week to complete the lab, after which time the tests were collected. Of the 22 students from the CLI group, 16 participated by returning their tests. Similarly, of the 25 students in the GUI group, 21 returned their tests for grading. Both groups of tests were graded using the same rubric and standards.

Accuracy of the testing may have been limited by several factors. Firstly, as with any control and test group from different populations there are inherent variances in personal experience and abilities. Next, because the two test groups had different instructors there may have been variation in the focus and emphasis of their instruction prior to completing the lab and testing. The testing environment was not strictly controlled, although students were instructed to only use what they remembered from classroom instruction and the lab experience to complete the post-test, and not to reference any other learning materials. Still, there was no way to ensure that all students complied with this

instruction, which may have affected some of the results. Additionally, the lab and post-test were not timed and students completed either one version of the lab or the other, so there is no way to know if either the CLI or GUI was easier for each group of students to execute and understand. Finally, small sample sizes often contribute to large variances in statistics from small differences in results. A larger sample size of students and a greater number of test questions could have provided more information about the two groups or made the differences between their scores more significant.

D.   DATA

All data was summarized and analyzed with Microsoft Excel. The scores were calculated out of a possible 20 points. The average score for the group that completed the lab using the CLI was 16 (or 80%). The average score for the group using the GUI was 16.857 (or 84.3%). The boxplot in Figure 19 compares the test results of both groups showing the maximums, minimums, means, and quadrilles.
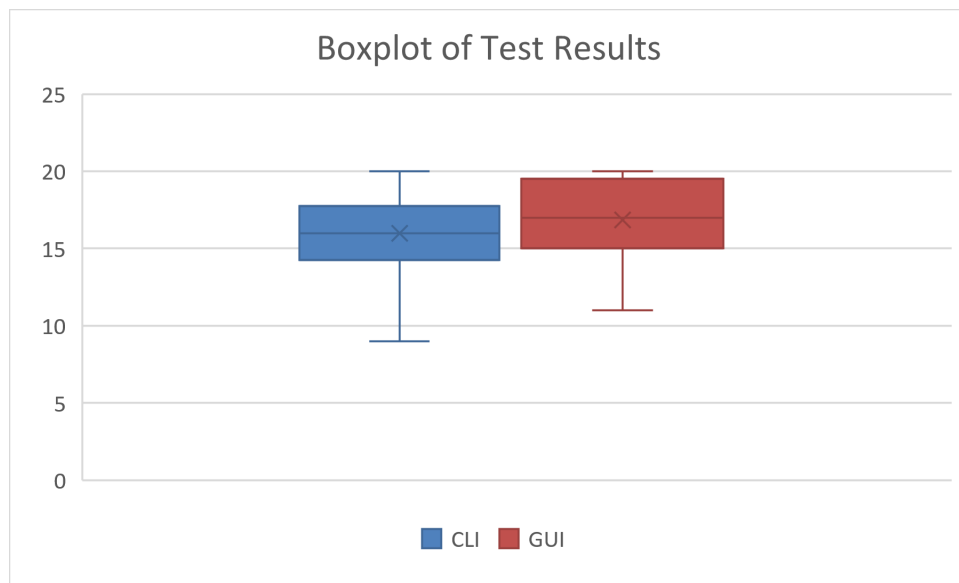
Figure 19.   Boxplot of test results.

This initial review of the results showed that the GUI group performed slightly better overall. The results were also compared on a question and option basis as seen in

Figures 20 and 21, respectively, and the averages of each answer are compared and averaged together in Table 1.
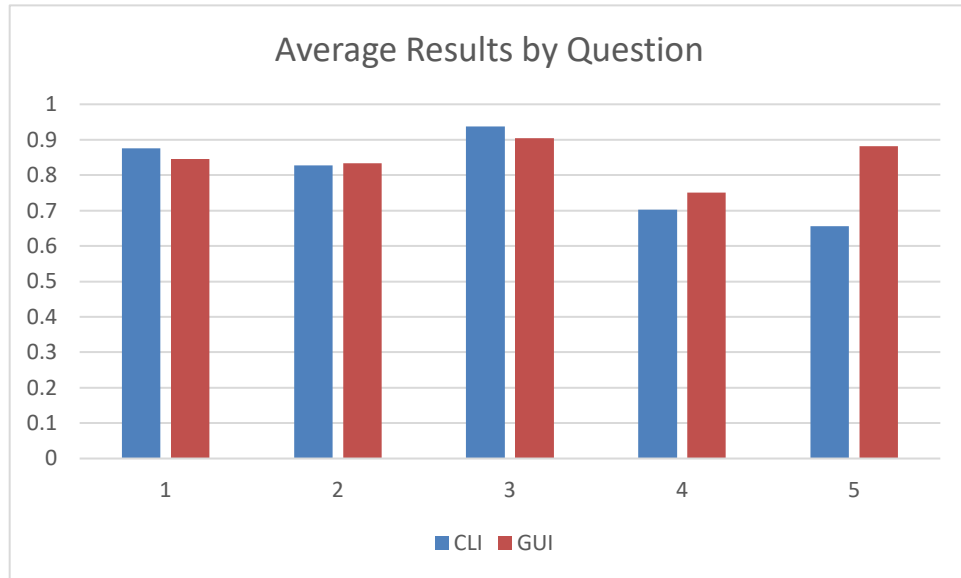
Figure 20.   Average results by question.
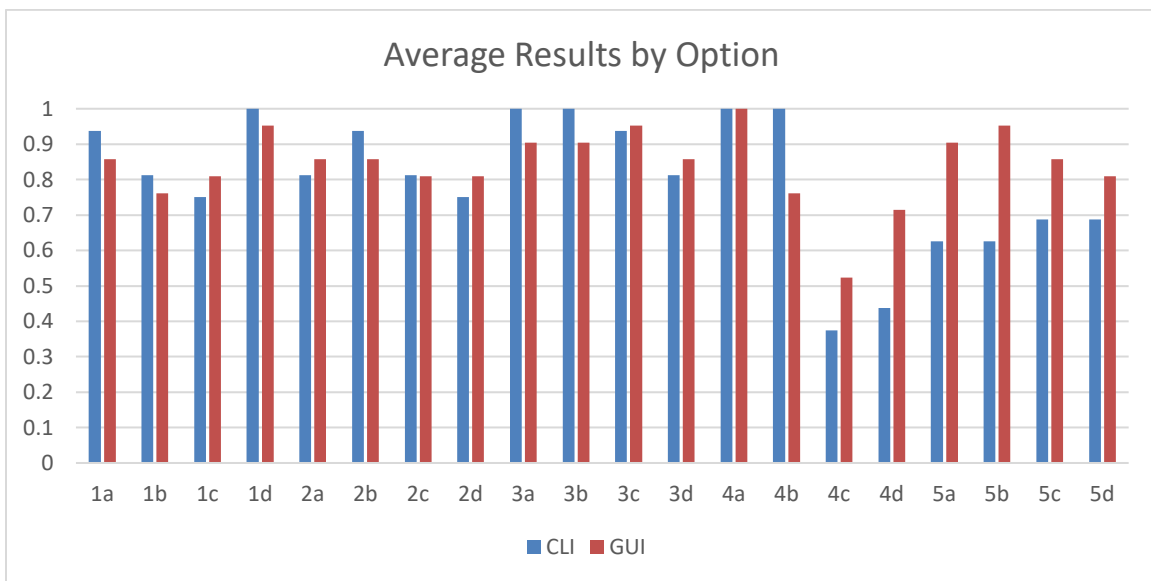
Figure 21.   Average results by option.

Table 1. Question average scores with difference and overall average.

	Q1	Q2	Q3	Q4	Q5
CLI	0.875	0.828	0.938	0.703	0.656
GUI	0.845	0.833	0.905	0.75	0.881
Difference	0.03	-0.005	0.033	-0.047	-0.225
Average	0.86	0.8305	0.9215	0.7265	0.7685

Both groups performed similarly (within 5%) on all but question 5 (Which mode(s) use chaining or feedback during encryption or decryption?), where the GUI group performed slightly better, a difference of 22.5%. Question 4 (How many bits in an AES key?) showed the most surprising results. Both groups showed the worst performance on this question: 70.3% for the CLI group, 75% for the GUI group and 72.65% overall. However, when broken down by option, all participants got the first option correct and 88% got the second option correct. Compared with the second two options where the overall averages were 44.94% and 57.61%, respectively, this may show that neither group fully understood the various options for key size using AES. This is less surprising from the GUI group because the tool was limited to only 128-bit key size. The CLI group, however, had the option to work with several key sizes during the lab. This variance could possibly be attributed to the different instruction received before lab completion.

Since the two test groups were of different sizes, further data analysis was required to determine the validity of differences in their scores. This was accomplished by performing a t-Test using the data analysis functionality in Microsoft Excel as shown in Table 2.

Table 2. t-Test: Two-sample assuming unequal variances.

t-Test: Two-Sample Assuming Unequal Variances		
	CLI	GUI
Mean	16	16.85714286
Variance	8.666666667	7.828571429
Observations	16	21
Hypothesized Mean Difference	0	
df	32	
t Stat	-0.896338096	
P(T<=t) one-tail	0.188383005	
t Critical one-tail	1.693888748	
P(T<=t) two-tail	0.37676601	
t Critical two-tail	2.036933343	

The significance threshold was set at .05 and the *p* value of the two-tail t-Test for the data set was 0.37676601. Because the *p* value was greater than .05, the difference in the two test groups is not statistically significant. This shows that neither group performed significantly better or worse than the other.

E.   DISCUSSION

While a quick look at the raw data may hint that the GUI group performed better than the CLI group, deeper analysis of the data shows that the difference was not statistically significant. This is likely a combination of the small number of participants in the test groups, and the relatively small amount of data collected, both of which contribute to large variations in the data set. Because neither group performed significantly better, it can be said that, based on the test used with the two groups, the GUI did not improve or impede learning or comprehension of the lab objectives.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

Advances in technology allow and require continual improvements in processes, operations, and even education. Capitalizing on studies in design and human-computer interaction can create better interfaces and facilitate better learning. By creating usable tools for learning, improved student attention and retention can be better realized. The GUI encryption tool envisioned and created for this thesis was meant to serve this purpose, to remove the mental load of complex computer interaction from students so they could focus exclusively on the learning material.

The two thesis questions were:

1. Is it possible to create a GUI to replace the CLI for exploring encryption modes?

2. Does a GUI enhance student learning of concepts of encryption modes and their properties?

The answer to the first question is yes, all modes explored in the original symmetric encryption lab were captured in the GUI and students were able to interact with the encrypted and decrypted files to adequately complete the lab assignment.

The second question is less concrete. The data did not show a significant difference between the results of the group completing the lab using the CLI and the group using the GUI. So, the GUI did not necessarily enhance student learning, but it did not inhibit learning and could be a viable alternative to the CLI.

The original motivation for the creation of a GUI was to eliminate the tedious and confusing CLI entry and allow students to focus more on the intent of the lab and less on precise typing and exact input. To this end the GUI has met its purpose: students were provided a streamlined and usable interface that allowed them to complete all required tasks of the lab with limited chance of error. On the other hand, it could also be argued that exclusive use of the CLI in this lab would lead to greater expertise in the CLI interactions

that occur in later lab assignments in CS3600 and in other courses throughout the computer science curriculum. While students could potentially benefit from increased exposure to CLI, the purpose of the lab is to increase their understanding of encryption algorithms, not to become experts in command-line interactions. Regardless, even students who completed the GUI version of the lab were required to utilize the CLI for portions and received the tangential benefits of CLI interaction.

Overall the GUI is a useful tool for student learning and comprehension and should replace the legacy CLI lab instruction. Further improvements to the GUI will only increase its efficacy.

B.   FUTURE WORK

Work to improve the GUI in the future should first focus on construction of a standalone application, ideally one that could be included in the CS3600 virtual lab environment. An application would make the GUI more accessible, eliminating the need for a network connection, and the overhead and maintenance of hosting it on a web server.

The GUI could further be improved to enhance learning with some minor modifications to the existing design and implementation. Firstly, allowing students to select different key sizes would provide them increased interaction and comprehension of the flexibility and power of the algorithm. Similarly, the ability to choose from more than one algorithm (i.e., something other than the default AES-128 algorithm) would allow students to experiment and increase their understanding of symmetric key encryption.

In the area of block encryption modes, since the Counter mode was not included in the original implementation, this could also be added to allow more experimentation with encryption modes. In addition, to better illustrate the difference in complexity between various block modes, a timer or other feedback after encryption would allow students to compare how long each mode takes to complete. Because these processes happen so quickly students may not notice the variance on their own, and this feedback could provide further insight into the difference between the modes.

Other improvements to the app could include changing how files are downloaded and saved on the host computer, as this seemed to be a problem for some students. Finally, a beneficial change to the key and IV input would provide padding and eliminate the need for a specific input length.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. DJANGO CODE

A.   VIEWS.PY

```python
from django.http import HttpResponse
from django.shortcuts import render, render_to_response
from django.core.files.storage import FileSystemStorage
from Cryptodome.Cipher import AES
from Cryptodome.Util.Padding import pad, unpad
from django.core.files.base import ContentFile

def home(request):
    return render(request, 'home.html')

def my_encode(request):
    return render(request, 'encode.html')

def my_decode(request):
    return render(request, 'decode.html')

def encrypt(request):
    file_tag = "file_to_encrypt"

    if request.method == 'POST' and request.FILES[file_tag]:
        mode = request.POST["mode"]
        key = request.POST["key"]
        iv = request.POST["iv"]
        myfile = request.FILES[file_tag]

        data = myfile.read()

        fs = FileSystemStorage()

        filename = fs.save(myfile.name, myfile)
        uploaded_file_url = fs.url(filename)

        ciphertext = encode(data, key, mode, iv)

        enc_file = ContentFile(ciphertext)
        enc_filename = fs.save("enc_" + myfile.name, enc_file)
        download_file_url = fs.url(enc_filename)

        return render(request, 'download.html', {'uploaded_file_url': uploaded_file_url,
                            'download_file_url': download_file_url})

    return render(request, "not_ok.html")

def decrypt(request):
    file_tag = "file_to_decrypt"

    if request.method == 'POST' and request.FILES[file_tag]:
        mode = request.POST["mode"]
        key = request.POST["key"]
        iv = request.POST["iv"]
        block_size = 16
        myfile = request.FILES[file_tag]

        data = myfile.read() ## convert byte to string
```

49

```python
        fs = FileSystemStorage()

        filename = fs.save(myfile.name, myfile)
        uploaded_file_url = fs.url(filename)

        text = decode(data, key, mode, iv)

        dec_file = ContentFile(text)
        dec_filename = fs.save("dec_" + myfile.name, dec_file)
        download_file_url = fs.url(dec_filename)

        return render(request, 'download.html', {'uploaded_file_url': uploaded_file_url,
                            'download_file_url': download_file_url})

    return render(request, "not_ok.html")


def download(request):
    filename = request.GET['file_to_download']

    ## use application/octet-stream for content_type vs text/plain
    with open(filename, 'rb') as f:
        response = HttpResponse(f.read(), content_type='application/octet-stream')
        response['Content-Disposition'] = 'attachment; filename=' + filename
        response['Content-Type'] = 'application/octet-stream; charset=utf-8'
        return response

def encode(data, key, mode, iv) :
    key = key.encode()
    iv = iv.encode()
    block_size = 16

    if mode == 'ecb':
        data = pad(data, block_size, style='pkcs7')
        encryption_suite = AES.new(key, AES.MODE_ECB)
        enc_data = encryption_suite.encrypt(data)
    elif mode == 'cbc':
        data = pad(data, block_size, style='pkcs7')
        encryption_suite = AES.new(key, AES.MODE_CBC, iv)
        enc_data = encryption_suite.encrypt(data)
    elif mode == 'cfb':
        encryption_suite = AES.new(key, AES.MODE_CFB, iv)
        enc_data = encryption_suite.encrypt(data)
    elif mode == 'ofb':
        encryption_suite = AES.new(key, AES.MODE_OFB, iv)
        enc_data = encryption_suite.encrypt(data)
    return enc_data


def decode(data, key, mode, iv):
    key = key.encode()
    iv = iv.encode()
    block_size = 16

    if mode == 'ecb':
        decryption_suite = AES.new(key, AES.MODE_ECB)
        dec_data = decryption_suite.decrypt(data)
        dec_data = unpad(dec_data, block_size, style='pkcs7')
    elif mode == 'cbc':
        decryption_suite = AES.new(key, AES.MODE_CBC, iv)
```

50

```python
        dec_data = decryption_suite.decrypt(data)
        dec_data = unpad(dec_data, block_size, style='pkcs7')
    elif mode == 'cfb':
        decryption_suite = AES.new(key, AES.MODE_CFB, iv)
        dec_data = decryption_suite.decrypt(data)
    elif mode == 'ofb':
        decryption_suite = AES.new(key, AES.MODE_OFB, iv)
        dec_data = decryption_suite.decrypt(data)

    return dec_data

def mypagenotfound(request):
    return render(request, 'error.html')

def myservererror(request):
    return render(request, 'error.html')

def mypermissiondenied(request):
    return render(request, 'error.html')

def mybadrequest(request):
    return render(request, 'error.html')
```

B.   HOME.HTML

```html
<html lang="en">
{% load staticfiles %}
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">


    <title>Encryption Home Page</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
crossorigin="anonymous">


  </head>

  <body>

    <header>
      <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
            <a class="nav-item nav-link active" href="{% url 'home' %}">Home <span class="sr-
only">(current)</span></a>
              <a class="nav-item nav-link" href="{% url 'encode' %}">Encryption</a>
```

```html
        <a class="nav-item nav-link" href="{% url 'decode' %}">Decryption</a>
      </div>
    </div>
  </nav>
</header>

<main role="main">

  <section class="jumbotron text-center">
    <div class="container">
      <h1 class="jumbotron-heading">Encryption & Decryption Services</h1> <br>
        <h4>This encryption/decryption service uses 128 bit AES (Advanced Encryption Standard) </h4>
      <!--p class="lead text-muted">This encryption/decryption service uses 128 bit AES (Advanced
Encryption Standard).</p>
        <!--p>
         <a href="mailto:eehuntoo@nps.edu" class="btn btn-primary my-2">Email Me</a>
        </--p> -->
    </div>
  </section>

  <div class="container">
    <!-- Example row of columns -->
    <div class="row">
      <div class="col-md-6">
        <h2>Encrypt</h2>
        <p>Convert plaintext to ciphertext.  </p>
        <p><a class="btn btn-danger" href="{%  url 'encode' %}"role="button">Encrypt</a></p>
      </div>
      <div class="col-md-6">
        <h2>Decrypt</h2>
        <p>Convert ciphertext to plaintext. </p>
        <p><a class="btn btn-success" href="{% url 'decode' %}" role="button">Decrypt</a></p>
      </div>
    </div>

    <hr>

  </div> <!-- /container -->

</main>

<footer class="text-muted">
  <div class="container text-center">
    <p>© NPS {% now 'Y' %}</p>
  </div>
</footer>

<!-- Bootstrap core JavaScript
================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQlAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"
integrity="sha384-cs/chFZiN24E4KMATLdqdvsezGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"
integrity="sha384-uefMccjFJAlv6A+rW+L4AHf99KvxDjWSu1z9Vl8SKNVmz4sk7buKt/6v9Kl65qnm"
crossorigin="anonymous"></script>
```

```
      </body>
    </html>
```

C.   ENCODE.HTML

```html
<html lang="en">
{%  load staticfiles %}
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Encoding Page</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
crossorigin="anonymous">

  </head>

  <body>

    <header>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
          <a class="nav-item nav-link" href="{% url 'home' %}">Home</a>
          <a class="nav-item nav-link active" href="{% url 'encode' %}">Encryption <span class="sr-
only">(current)</span></a>
          <a class="nav-item nav-link" href="{% url 'decode' %}">Decryption</a>
        </div>
      </div>
    </nav>
    </header>

    <main role="main">
    <div class="col-md-4">
     <h3 class="text-muted">Encryption Service</h3>
     <form action="{%  url 'encrypt' %}" method="post" enctype="multipart/form-data">
      {% csrf_token %}
      File to encrypt: <input type="file" name="file_to_encrypt" required> <br><br>
      Mode: <br>
      <input type="radio" name="mode" id="ecb" value="ecb" onclick="changecolor('grey')"
required>ECB<br>
      <input type="radio" name="mode" id="cbc" value="cbc"
onclick="changecolor('black')">CBC<br>
      <input type="radio" name="mode" id="cfb" value="cfb"
onclick="changecolor('black')">CFB<br>
      <input type="radio" name="mode" id="ofb" value="ofb"
```

```html
onclick="changecolor('black')">OFB<br><br>
        Key: <input type="text" name="key" value="0000000000000000" required minlength="16" maxlength="16">
        <a href="#" data-toggle="tooltip" data-placement="right" title="Required. 16 characters (128 bits). Remember for use in decryption.">?</a><br><br>
        IV: <input type="text" id="iv" name="iv" value="0000000000000000" minlength="16" maxlength="16">
        <a href="#" data-toggle="tooltip" data-placement="right" title="16 characters. Remember for use in decryption.">?</a><br><br>

        <button type="submit"  class="btn btn-danger">Encrypt</button>

   </form>
   </div>

   </main>

   <footer class="text-muted">
     <div class="container text-center">
        <p>© NPS {% now 'Y' %}</p>
     </div>
   </footer>

   <!-- Bootstrap core JavaScript
   ================================================== -->
   <!-- Placed at the end of the document so the pages load faster -->
   <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQlAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js" integrity="sha384-cs/chFZiN24E4KMATLdqdvsezGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ" crossorigin="anonymous"></script>
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9Vl8SKNVmz4sk7buKt/6v9Kl65qnm" crossorigin="anonymous"></script>
   <script>
     function changecolor(newcolor) {
        var elem = document.getElementById('iv');
        elem.style.color = newcolor;
     }
   </script>
   <script>
     $(document).ready(function(){
     $('[data-toggle="tooltip"]').tooltip();
     });
   </script>

   </body>
</html>
```

D.   DECODE.HTML

```html
<html lang="en">
{%  load staticfiles %}
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

```html
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Decryption Page</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
crossorigin="anonymous">
  </head>

  <body>

  <header>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <!-- <a class="navbar-brand" href="{% url 'home' %}">Home</a> -->
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
          <a class="nav-item nav-link" href="{% url 'home' %}">Home</a>
          <a class="nav-item nav-link" href="{% url 'encode' %}">Encryption</a>
          <a class="nav-item nav-link active" href="{% url 'decode' %}">Decryption <span class="sr-
only">(current)</span></a>
        </div>
      </div>
    </nav>
  </header>

  <main role="main">
  <div class="col-md-4">
    <h3 class="text-muted">Decryption Service</h3>
    <form action="{% url 'decrypt' %}" method="post" enctype="multipart/form-data">
     {% csrf_token %}

      File to decrypt: <input type="file" name="file_to_decrypt" required> <br><br>
      Mode: <br>
      <input type="radio" name="mode" id="ecb" value="ecb" onclick="changecolor('grey')"
required>ECB<br>
      <input type="radio" name="mode" id="cbc" value="cbc"
onclick="changecolor('black')">CBC<br>
      <input type="radio" name="mode" id="cfb" value="cfb"
onclick="changecolor('black')">CFB<br>
      <input type="radio" name="mode" id="ofb" value="ofb"
onclick="changecolor('black')">OFB<br><br>
      Key: <input type="text" name="key" value="0000000000000000" required minlength="16"
maxlength="16">
      <a href="#" data-toggle="tooltip" data-placement="right" title="Required. 16 characters (128
bits). Must match encryption key.">?</a><br><br>
      IV: <input type="text" id="iv" name="iv" value="0000000000000000" minlength="16"
maxlength="16">
      <a href="#" data-toggle="tooltip" data-placement="right" title="16 characters. Must match
encryption IV.">?</a><br><br>

      <button type="submit"  class="btn btn-success">Decrypt</button>

    </form>
```

```
        </div>

      </main>

      <footer class="text-muted">
        <div class="container text-center">
          <p>© NPS {% now 'Y' %}</p>
        </div>
      </footer>

      <!-- Bootstrap core JavaScript
      ================================================== -->
      <!-- Placed at the end of the document so the pages load faster -->
      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQlAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"
integrity="sha384-cs/chFZiN24E4KMATLdqdvsezGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ"
crossorigin="anonymous"></script>
      <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"
integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9Vl8SKNVmz4sk7buKt/6v9Kl65qnm"
crossorigin="anonymous"></script>
      <script>
        function changecolor(newcolor) {
          var elem = document.getElementById('iv');
          elem.style.color = newcolor;
        }
      </script>
      <script>
        $(document).ready(function(){
        $('[data-toggle="tooltip"]').tooltip();
        });
      </script>

  </body>
</html>
```

E.   DOWNLOAD.HTML

```
<html lang="en">
{% load staticfiles %}
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>File Download</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
crossorigin="anonymous">
  </head>

  <body>
```

```html
<header>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-item nav-link" href="{% url 'home' %}">Home</a>
        <a class="nav-item nav-link" href="{% url 'encode' %}">Encryption</a>
        <a class="nav-item nav-link" href="{% url 'decode' %}">Decryption</a>
      </div>
    </div>
  </nav>
</header>

<main role="main">
  <div class="col-md-4">
    {% if uploaded_file_url %}
      <br><h6> File uploaded is {{ uploaded_file_url }}</h6> <br>
      <h6> File to download is {{ download_file_url }}</h6>

      <form action="{% url 'download' %}" >
        <input type="hidden" name="file_to_download" value= "{{ download_file_url }}">
        <button type="submit">Download</button>
      </form>
    {% endif %}
  </div>

</main>

<footer class="text-muted">
  <div class="container text-center">
    <p>© NPS {% now 'Y' %}</p>
  </div>
</footer>

<!-- Bootstrap core JavaScript
================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQlAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js" integrity="sha384-cs/chFZiN24E4KMATLdqdvsezGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9Vl8SKNVmz4sk7buKt/6v9KI65qnm" crossorigin="anonymous"></script>

</body>
</html>
```

F.   ERROR.HTML

```html
<!DOCTYPE html>

<html lang="en">
{% load staticfiles %}
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Error</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-
9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
crossorigin="anonymous">
  </head>

  <body>

    <header>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
          <a class="nav-item nav-link" href="{% url 'home' %}">Home</a>
          <a class="nav-item nav-link" href="{% url 'encode' %}">Encryption</a>
          <a class="nav-item nav-link" href="{% url 'decode' %}">Decryption</a>
        </div>
      </div>
    </nav>
    </header>

    <main role="main">
      <div class="col-md-4">
        <h4>There was an error with your encryption/decryption.</h4><br>
        <h4>Please try again.</h4><br>
        <h6>Possible issues:</h6>
        <h6>Space character in file name</h6>
        <h6>Key and/or IV do not match between encryption and decryption. </h6>
      </div>

    </main>

    <footer class="text-muted">
     <div class="container text-center">
       <p>© NPS {% now 'Y' %}</p>
     </div>
    </footer>

    <!-- Bootstrap core JavaScript
    ================================================== -->
    <!-- Placed at the end of the document so the pages load faster -->
```

```html
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"
integrity="sha384-cs/chFZiN24E4KMATLdqdvsezGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"
integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9Kl65qnm"
crossorigin="anonymous"></script>

  </body>
</html>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. LAB POST-TEST

Correct answers marked in bold.

1. What kind of cipher is AES? (All that apply)
   a. Stream
   b. **Block**
   c. **Symmetric**
   d. Asymmetric

2. Which mode(s) of AES encryption require an Initialization Vector? (All that apply)
   a. ECB
   b. **CBC**
   c. **CFB**
   d. **OFB**

3. How many bits are in an AES block? (All that apply)
   a. 64
   b. **128**
   c. 192
   d. 256

4. How many bits are in an AES key? (All that apply)
   a. 64
   b. **128**
   c. **192**
   d. **256**

5. Which mode(s) use chaining or feedback during encryption or decryption? (All that apply)
   a. ECB
   b. **CBC**
   c. **CFB**
   d. **OFB**

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1]   *Federal Information Processing Standards (FIPS), Publication 197*, 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[2]   NIST, *ADVANCED ENCRYPTION STANDARD (AES) Fact Sheet*, 2002. [Online]. Available: https://web.archive.org/web/20020211162045/http://csrc.nist.gov:80/encryption/aes/round2/aesfact.html.

[3]   H. Tipton and M. Krause, *Information Security Management Handbook,* Sixth Edition. Boca Raton, FL, USA: Auerbach Publications, 2007.

[4]   M. Dworkin, "NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation, Methods and Techniques." National Institute of Standards and Technology, Dec 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

[5]   B. Schneier, *Applied Cryptography: Protocols, algorithms, and source code in C*, Second. New York, NY, USA: John Wiley and Sons, Inc., 1996.

[6]   "Block cipher mode of operation," *Wikipedia*. Accessed Sep 28, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[7]   *Ergonomics of human-system interaction - Specification for the process assessment of human-system issues*, ISO - International Organization for Standardization, 2003.

[8]   F. E. Ritter, E. Churchill, and G. D. Baxter, *The Basics of Human-System Interaction: What System Designers Really Need to Know about People.* University Park, PA, USA: Penn State University, 2011.

[9]   W. Lidwell, K. Holden, and J. Butler, *Universal Principles of Design*. Glocester, MA, USA: Rockport Publishers, 2003.

[10]   L. Westfall, *Certified Software Quality Engineer Handbook*. Milwaukee, WI, USA: ASQ Quality Press, 2009.

[11]   T. Boyle, *Design for multimedia learning*. New York, NY, USA: Prentice Hall, 1997.

[12]   S. C. Sivakumar, W. Robertson, M. Artimy, and N. Aslam, "A web-based remote interactive laboratory for Internetworking education," *IEEE Trans. Educ.*, vol. 48, no. 4, pp. 586–598, Nov. 2005. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1532367

[13]   J. Thioulouse and S. Dray, "Interactive Multivariate Data Analysis in *R* with the ade4 and ade4TkGUI Packages," *Journal of Statistical Software*, vol. 22, no. 5, Sep. 2007. [Online]. Available: https://core.ac.uk/download/pdf/25968385.pdf

[14]   D. Genter and J. Nielsen, "The Anti-Mac User Interface," *Communications of the ACM,* vol. 9, no. 8, pp. 70-82, Aug. 996. [Online]. Available: https://www.nngroup.com/articles/anti-mac-interface/.

[15]   "Requirements Gathering," class notes for Human Computer Interaction, Dept. of Computer Science, Naval Postgraduate School, Monterey, CA, USA, winter 2018.

[16]   "Conceptual Design," class notes for Human Computer Interaction, Dept. of Computer Science, Naval Postgraduate School, Monterey, CA, USA, winter 2018.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
    Ft. Belvoir, Virginia

2. Dudley Knox Library
    Naval Postgraduate School
    Monterey, California