University of Liverpool

Simple cryptography

Detailed Proposal

Gubin Zhao

201536436

Contents

Statement of Ethical Compliance

I have categorized my project as follows:
Data Category A: I will not use any data derived from humans or animals in this project.
Participant Category 0: I will not involve human participants in any activity related to this project.
I undertake the individual project on Simple Cryptography, hereby confirming that I will strictly adhere to the ethical guidance provided by the university. My project involves the development of software to implement various simple ciphers for text encryption and decryption.
As this is an individual project, I take full responsibility for ensuring ethical compliance throughout its execution. While my initial proposal does not include human participants or data, I understand the importance of transparency and ethical considerations. Should there be any changes to my project's scope or activities, such as the addition of evaluation activities involving human participants or data, I commit to updating this statement in future assessments and consulting with my supervisor or relevant authorities before continuing.
I am committed to supporting the ethical standards set by the university and will ensure that all aspects of my project adhere to these guidelines.

Project Description

Cryptography is the process of securing information by transforming it (encrypting it) into an unreadable format. Only those who own the decryption key can decipher (decrypt) and read the information. This project focuses on implementing simple cryptographic methods, known as ciphers, to both encrypt and decrypt textual data. In addition, the project will delve into basic cryptanalysis techniques that attempt to decrypt data without the necessary encryption key. [1] [2]

1. Ciphers to be Implemented

The software will offer encryption and decryption functionality via three primary cipher methods:

a. **Shift Cipher (e.g., Caesar's Cipher):**
   This is one of the earliest and simplest forms of symmetric encryption. The idea is to shift each letter in the text by a fixed number of positions in the alphabet. For instance, a shift of 3 would transform the letter 'A' to the letter 'D'. [3]

b. **Permutation Cipher:**
   In this method, the positions of individual letters or groups of letters are shuffled (permuted) according to a specific system. This rearrangement of positions serves as the encrypted message. [3]

c. **Substitution Cipher:**
   Here, each letter or group of letters in the plaintext is replaced with another letter or group of letters. The substitution is consistent throughout the message. This means that if 'A' is replaced by 'X', then every occurrence of 'A' in the plaintext will become

'X' in the ciphertext. [3]

2. Cryptanalysis and Decryption [3] [4] [5]

The software will also supply tools to decrypt encoded messages. Two primary decryption methodologies will be explored:

a. Decryption with Known Key:

If the decryption key is available, the software will use it to quickly and efficiently revert the ciphertext into its original plaintext form.

b. Decryption without Known Key (Simple Cryptanalysis):

In situations where the encryption key is not known, the software will employ basic cryptanalysis techniques, primarily focusing on frequency analysis. Since certain letters or groups of letters appear more frequently in most languages (e.g., 'E' in English, this knowledge can be used to make educated guesses about the original message. This method will be most effective on relatively long texts where patterns become more evident.

For each type of cipher, the cryptanalysis methods without the known key would look as follows:

1) Shift Cipher (Caesar's Cipher):

- **Brute-force Method:** Test all 25 possible shifts (since the shift of 0 would give the original text) and see which yields a readable message. This is fast due to the limited possibilities.
- **Frequency Analysis:** Look at the frequency of single letters and compare it to typical letter frequencies in the language of the ciphertext. This method might be slightly slower than brute-force but offers a more educational insight into the nature of the cipher.

2) Permutation Cipher:

- **Pattern Recognition:** Search for patterns that could indicate common words or phrases. This is a more time-consuming process because there are many more permutations than shifts in a Caesar cipher.
- **Statistical Analysis:** Analyse the frequency of letter combinations and compare them to the expected frequencies in the target language. The effectiveness increases with the length of the ciphertext.

3) Substitution Cipher:

- **Frequency Analysis:** Examine single letter frequencies and look at digraphs (pairs of letters) and trigraphs (triplets), as well as common words. This is the primary method for breaking this cipher and can be quite time-consuming.
- **Codebook Method:** If certain words or phrases are known to be in the ciphertext, a 'codebook' or 'crib' can be used to match these and reveal parts of the key.

3. Testing and Evaluation

For the effectiveness of both the encryption methods and the cryptanalysis techniques, it is essential to test them on relatively long texts. This will not only ensure the robustness of the implemented ciphers but also gauge the effectiveness of the cryptanalysis approach, especially when the encryption key is unknown.

Aims & Requirements

1. Aims

1) Understand Basic Cryptographic Methods

- To comprehend and implement foundational cryptographic ciphers to securely transform plaintext into ciphertext and vice versa.

2) Facilitate Cryptanalysis Techniques

- To explore and apply basic cryptanalysis methods, particularly frequency analysis, to decrypt messages without the knowledge of encryption keys.

3) Evaluate the Effectiveness of Ciphers

- To test and evaluate the robustness of the implemented encryption techniques and the effectiveness of the decryption methods.

2. Requirements

1) Essential

a. Cipher Implementation

- Implement at least three distinct ciphers: Shift (e.g., Caesar's), Permutation, and Substitution.

b. Encryption & Decryption Interface

- A user-friendly interface to easily input plaintext/ciphertext and view the encryption/decryption results.

c. Key Input Mechanism

- Allow users to input or generate encryption keys for encoding and decoding processes.

d. Cryptanalysis Module

- Implement a module to attempt decryption without the knowledge of the encryption key, primarily utilizing frequency analysis and enumeration.

e. Testing Framework

- A built-in framework to test the ciphers on various text categories, measuring success rates, accuracy, and time efficiency.

2) Desirable

a. Multi-language Support

- The ability to handle and analyse texts from multiple languages, considering the unique frequencies and patterns of each language.

b. Visualization Tools

- Tools to visually represent the frequency distribution of characters in encrypted texts, aiding in cryptanalysis.

c. User Guide & Documentation

- A comprehensive guide to assist users in navigating the software and understanding the underlying cryptographic principles.

d. Feedback Mechanism

- An in-software system to collect user feedback for iterative improvements in future versions.

e. Advanced Cryptanalysis Techniques

- Beyond frequency analysis, integrate other basic cryptanalysis methods to improve the decryption success rate without known keys.

Key Literature & Background Reading

Developing a website for simple cryptography involves merging the disciplines of cryptographic principles with web development technologies. With a focus on Python and Flask, I delved into both the foundational cryptographic literature and resources specific to web-based application development. This section highlights the literature and resources consulted during the project's inception.

1. Cryptography Principles and Simple Ciphers

a. **Singh, S. (2000). The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.** [2]
   - This book supplies an extensive history of cryptography, touching upon simple ciphers like the Caesar cipher, explaining the fundamentals in an easily digestible manner.

b. **Stallings, W. (2005). Cryptography and Network Security: Principles and Practice.** [6]
   - A comprehensive text on various encryption methods, including those of simple ciphers. It also offers insights into the theoretical underpinnings and real-world applications of these ciphers.

c. **Douglas, R. (2005). Cryptography: Theory and Practice.** [3]
   - A comprehensive treatise on cryptographic algorithms and their mathematical underpinnings, Stinson's work offered a robust theoretical foundation to my project. It particularly helped in refining my understanding of substitution and permutation ciphers.

2. Cryptanalysis and Frequency Analysis

a. **Kahn, D. (1996). The Codebreakers - The Comprehensive History of Secret Communication from Ancient Times to the Internet.** [1]
   - This tome is an in-depth look at the history of code-breaking, detailing various techniques used throughout the ages. The chapters on frequency analysis will be of particular relevance.

b. **Lewand, R. (2000). Cryptological Mathematics. The Mathematical Association of America.** [7]
   - Offers mathematical techniques, including those behind frequency analysis, used in cryptanalysis. It dives into the why and how of breaking simple ciphers.

3. Flask and Web Development with Python

a. **Flask Web Development Documentation (Flask Documentation)**
   - The official Flask documentation was indispensable for understanding how to structure the web application, handle routes, and integrate cryptographic functionality seamlessly.

b. **Grinberg, M. (2018).** ***Flask Web Development: Developing Web Applications with Python*****.** [8]

- A step-by-step guide to building web applications using Flask. It covers best practices, database integration, and other advanced topics.

4. Cryptography Libraries in Python

a. API Documentation for Python's cryptography Library (Flask · PyPI)

- This resource was paramount in bridging the gap between cryptographic principles and their Pythonic implementation within the Flask environment.

5. Web Design & User Experience (for a web application perspective)

a. Krug, S. (2000). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. [9]

- While the primary focus is on cryptography, the user experience is crucial for any web application. Krug's book supplies insights on making web interfaces intuitive and user-friendly.

The breadth and depth of these resources ensure a holistic approach to the project. They provide the necessary theoretical foundation, historical context, technical know-how, and current research insights. As the project progresses, referring to these sources will ensure that design and development decisions are based on well-established principles and contemporary advancements in the field of cryptography.

Development & Implementation Summary

1. Development Environment

a. IDE: PyCharm

Reasons for Choosing:

- **Feature-Rich:** PyCharm is a comprehensive IDE tailored for Python development. Its features such as code completion, powerful debugging, and integrated testing simplify and accelerate the development process.
- **Integrated** Version **Control:** PyCharm seamlessly integrates with version control systems like Git, making it easier to track changes, branch, and merge during the development cycle.
- **Flask Support:** As I am developing a Flask-based web application, PyCharm offers dedicated support for Flask projects, easing the setup, and development process.
- **Extensible:** With a myriad of plugins available, PyCharm can be customized to fit the specific needs of this project.

b. Implementation Language: Python [10]

Reasons for Choosing:

- **Versatility:** Python is a general-purpose language that excels in various domains including web development and cryptography.
- **Libraries & Frameworks:** Python boasts a rich ecosystem of libraries which can be leveraged for cryptographic functions. Flask, being a Python micro-framework, makes web development straightforward and efficient.
- **Readability:** Python's clean syntax ensures the codebase remains readable and

maintainable, an essential factor for cryptographic projects where clarity can prevent security oversights.

2. Project Implementation

a. Cryptographic Module Development

- Starting with the simpler ciphers like Caesar's, before advancing to permutation and substitution ciphers.
- Utilize Python's **cryptography** library for any utility functions if needed.

b. Web Interface [7]

- Begin with setting up a basic Flask application.
- Create web forms for users to input text for encryption and decryption.
- Integrate the cryptographic modules with the web interface to enable real-time encryption and decryption.

c. Cryptanalysis Module

1) Caesar's Cipher (Shift Cipher):

- **Brute-force**: Decrypt using all 25 possible shifts due to its limited key space.
- **Frequency Analysis:** Use common letter frequencies in English (like 'e', 't') to guess the shift.

2) Permutation Cipher:

- **Known-plaintext Attack:** Use a segment of known plaintext and its ciphertext to deduce the permutation.
- **Frequency Analysis:** Look for patterns or common sequences in the ciphertext.

3) Substitution Cipher:

- **Frequency Analysis:** Compare letter frequencies in the ciphertext to standard English frequencies to make substitution guesses.
- **Pattern Words:** Identify words with distinct patterns (e.g., 'meet') in the ciphertext to determine substitutions.
- **Dictionary Attack:** Use guessed plaintext parts to reference a dictionary for possible word matches and further substitutions.

Incorporate this into the web interface, allowing users to attempt decryption without knowing the key.

d. Testing

- Use PyCharm's integrated testing tools.
- Begin with unit tests for individual cryptographic functions.
- Progress to integration tests for the entire Flask application.

3. Workflow Organization

1) **Requirements Gathering & Refinement**: Start with a clear understanding of the application's scope, its features, and functionalities.
2) **Prototyping:** Sketch a rough design of the web interface and flow of operations.
3) **Module** Development**:** Focus on one module at a time, starting with cryptographic functions, followed by the Flask interface, and finally, cryptanalysis.
4) **Integration:** Once individual modules are ready, work on integrating them.
5) **Testing:** Continuously test during the development phase. Ensure thorough testing before final deployment.

6) **Feedback &** Iteration**:** Regularly review the application's features and functionalities. Iterate based on feedback and any identified shortcomings.
7) **Deployment &** Documentation**:** Once satisfied with the functionality, deploy the application. Ensure comprehensive documentation for future reference and potential scalability.

Data Sources

1. User-Provided Text Data

- **Source:** Directly input by users into the web interface.
- **Obtaining Method:** Users will be prompted to enter text for encryption or decryption through a form on the web application.
- **Permission:** The text is provided willingly by the users, but it's vital to ensure they understand its use. A clear disclaimer will be placed, informing users not to input sensitive or personal information.
- **Confidentiality and Anonymity:** All text data entered by users will be processed in real-time without any storage, ensuring the confidentiality of the information. Anonymity will be preserved as users won't be required to provide any personal identifiers while using the encryption and decryption services.

2. Frequency Analysis Data (for Cryptanalysis)

- **Source:** Open-source datasets or publicly available texts.
- **Obtaining Method:** These texts can be retrieved from public domain repositories or libraries, ensuring they are free to use without any restrictions.
- **Permission:** Only publicly available texts or those with explicit permission for such use will be accessed.
- **Confidentiality and Anonymity:** Since these are public datasets, there's no personal information to be protected. However, ensuring these datasets are sourced from reputable platforms is essential.

Testing & Evaluation

To determine the effectiveness and robustness of the ciphers implemented in the "Simple Cryptography" project, a comprehensive testing and evaluation phase will be conducted. This phase is crucial for ensuring the software's reliability and for assessing the viability of the cryptanalysis techniques when faced with an unknown key.

1. Test Data Selection

To ensure a broad and comprehensive evaluation, the following categories of test data will be chosen:

a. **Random Texts:** These will serve to test the software's ability to handle texts without any discernible pattern.
b. **Literary Texts:** Excerpts from books, poems, or articles will be chosen to simulate real-world scenarios, as these often contain patterns inherent to the language.

c. **Controlled Texts:** Texts with known patterns, repetitions, or structures will be created. This will test the efficiency of the cryptanalysis techniques.

2. Testing Procedures

a. Encryption Tests

1) **Consistency:** The same plaintext, when encrypted with the same key, should always produce the same ciphertext.
2) **Unpredictability:** Different plaintexts, even with slight variations, should produce significantly different ciphertexts.

b. Decryption Tests with Known Key

1) **Accuracy:** Decrypting an encrypted message with the correct key should always return the original plaintext.
2) **Speed:** The software's ability to quickly decrypt long texts will be assessed.

c. Cryptanalysis Tests without Known Key

1) **Frequency Analysis Efficacy:** The software should be able to identify common patterns and frequencies in the encrypted text, giving clues to the potential content of the original message.
2) **Decryption Accuracy:** Even without the key, the software should be able to produce a plaintext that is close to the original or provides enough clarity to discern the intended message.

3. Evaluation Metrics

Post-testing, the following metrics will be used to evaluate the software:

a. Success Rate: The percentage of tests where the decrypted text matches (or closely matches) the original plaintext.

b. Time Efficiency: The duration taken to encrypt or decrypt texts of varying lengths.

c. Cryptanalysis Effectiveness [11]**:**

Chi-Squared Test [12]*:* To measure the deviation of the frequency distribution of the decrypted text from that of standard language texts.

Accuracy Score: A comparison of the n-gram frequency distributions of the decrypted and original texts

4. Feedback and Iteration

Based on the results from the testing and evaluation phase:

1) Any identified issues or weaknesses in the encryption and decryption processes will be addressed.
2) The software might undergo iterative improvements, especially in the cryptanalysis module, to enhance its decryption capabilities without the encryption key.

Project Ethics & Human Participants

1. Requirements Analysis

- **Activities:** Engaged potential users for feedback on application features.
- **Data Protection:** Feedback was recorded anonymously, with aggregated results to prevent individual identification.

2. Evaluation Phase

- **Activities:** Users will evaluate the application's functions and provide feedback.
- **Data Protection:** No personal identifiers will be included in feedback forms. Responses will be aggregated and anonymized.

3. Data Handling

- **Anonymization:** No personal details are recorded or stored.
- **Aggregation:** Data from participants is combined to ensure anonymity.
- **Storage & Deletion:** Data is securely stored with restricted access and will be deleted post-project.

Ethical Commitment:

All participants will be briefed on their role, the data collection process, and its use. They can opt out at any stage. This project adheres strictly to provided ethical guidelines, and any changes or exceptions will be discussed with the supervisor.

BCS Project Criteria

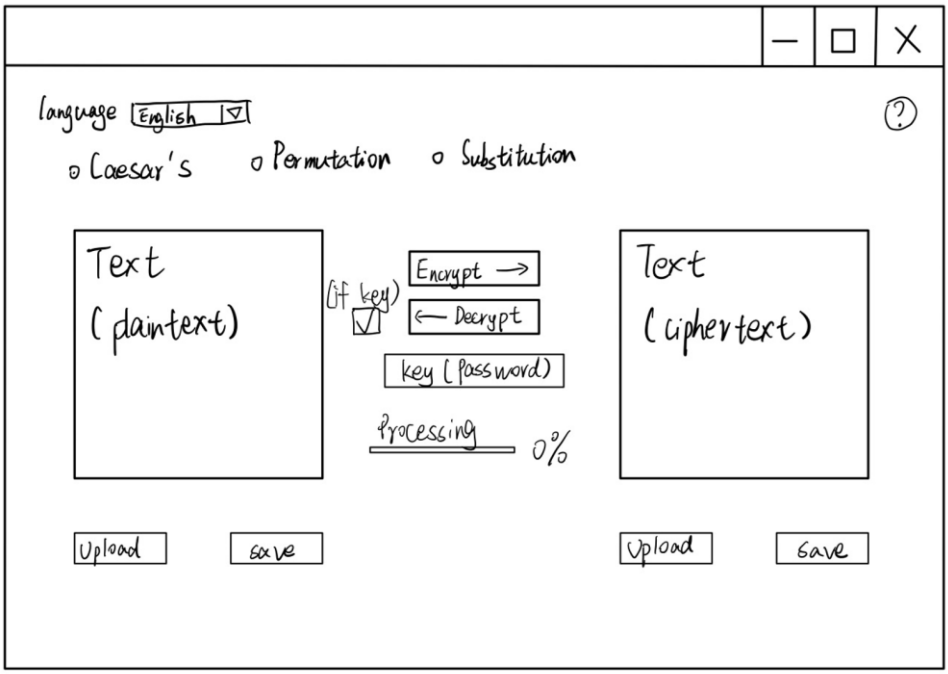
1. **Application of Practical and Analytical Skills:**
   - The project involves designing a cryptography-based web application using Flask and Python. This directly applies the practical programming and analytical problem-solving skills acquired during the degree programme. The process of encrypting, decrypting, and cryptanalysis embodies these skills.
2. **Innovation and/or Creativity:**
   - The combination of multiple simple ciphers in one user-friendly web interface is an innovative approach. Additionally, offering users the ability to decrypt without a known key using frequency analysis showcases creativity in solution design.
3. **Synthesis and Evaluation:**
   - The project synthesizes information from literature reviews, user requirements, and cryptography principles. The final product will be a holistic solution, followed by a rigorous evaluation phase to assess its effectiveness, usability, and security.
4. **Meeting a Real Need:**
   - In today's digital age, understanding encryption is pivotal. While sophisticated encryption methods exist, providing a platform that demystifies basic cryptography meets an educational and awareness need in the wider context.
5. **Self-Management:**
   - The project will be structured in phases, from requirements gathering to deployment. Utilizing tools like PyCharm for version control and task management ensures a well-organized, self-managed workflow.
6. **Critical Self-Evaluation:**
   - Throughout the project, regular reflection points will be established to assess progress, challenges, and areas of improvement. The project's conclusion will involve a comprehensive self-evaluation, addressing both successes and areas for potential enhancement.

These deliverables and methodologies are aligned with the BCS's criteria, ensuring the project not only meets academic standards but also professional ones. Other sections in

this document further elaborate on these points, solidifying the project's alignment with BCS requirements.

UI/UX Mockup

At this stage in the project's lifecycle, I have formulated an initial blueprint of my design intentions, as depicted in the attached mockup.

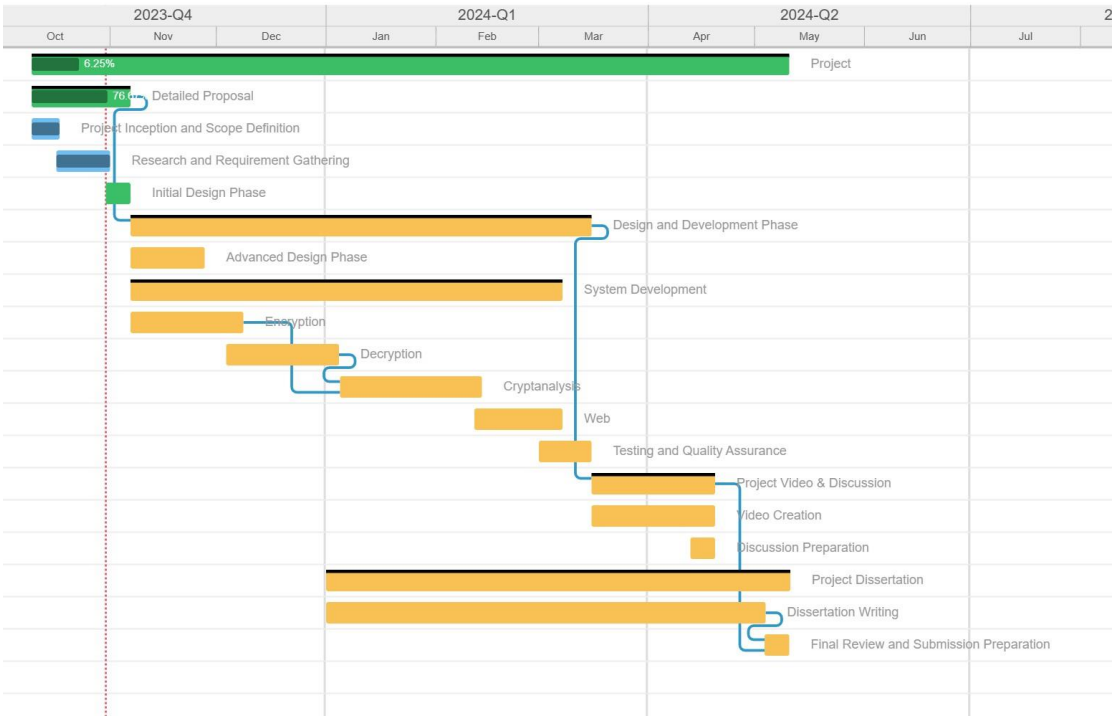


The proposed user interface prominently features:

- A language dropdown, allowing users to select their preferred language.
- Three encryption methods: Caesar's, Permutation, and Substitution, enabling users to choose the method that best suits their needs.
- Text input boxes for both plaintext and its resulting ciphertext, facilitating clear and straightforward data entry and retrieval.
- An 'Encrypt' and 'Decrypt' button, offering a clear pathway for users to secure their text or revert it to its original form.
- A key or password input option, determining to use random passwords when encrypting or use cryptanalysis when decrypting.
- A processing progress bar, ensuring users are kept informed of the application's status.
- 'Upload' and 'Save' options flanking both text boxes, granting users the flexibility to both import and export their data.

This design is my foundational blueprint, ensuring that my project is not just functional but also user centric. However, as with any early-stage mockup, elements might be refined as the development progresses. The primary aim remains to create an intuitive and efficient user experience.

Project Plan



1. **Detailed Proposal (Deadline: Mon 06 Nov @ 17:00)**
    - **Project Inception and Scope Definition (8 days)**
    - Define project objectives
    - Outline project deliverables
    - **Research and Requirement Gathering (15 days)**
    - Review of existing literature and solutions
    - Investigate user requirements
    - **Initial Design Phase (7 days)**
    - Draft preliminary design mockups
    - Gather iterative feedback *Dependencies:* None
2. **Design and Development Phase**
    - **Advanced Design Phase (21 days)**
    - Refine design based on proposal feedback
    - Finalize the user interface and system architecture *Dependencies*: Completion of Detailed Proposal
    - **System Development (130 days)**
    - Set up the development environment
    - Code and build the system(Encryption, Decryption, Cryptanalysis, Web)
    - Iterative testing and bug fixing *Can Overlap With:* Advanced Design Phase
    - **Testing and Quality Assurance (15 days)**
    - Unit testing
    - Integration testing
    - User acceptance testing *Dependencies:* Major completion of System Development
3. **Project Video & Discussion (Deadline: Fri 19 Apr @ 17:00)**

- **Video Creation (35 days)**
- Scripting and storyboarding
- Recording and editing
- Feedback and refinements
- **Discussion Preparation (7 days)**
- Finalize the main discussion points
- Rehearse the discussion presentation *Dependencies:* Testing and Quality Assurance

4. **Project Dissertation (Deadline: Fri 10 May @ 17:00)**
   - **Dissertation Writing (124 days)**
   - Compile all project details, research, findings, and analyses
   - Draft and revise the dissertation content
   - Proofreading and formatting
   - Writing *Can Overlap With:* System Development
   - **Final Review and Submission Preparation (7 days)**
   - Gather feedback on the dissertation
   - Make necessary revisions
   - Final proofreading and submission *Dependencies:* Project Video & Discussion and Dissertation Writing

Risks & Contingency Plans

Risks	Contingencies	Likelihood	Impact
Hardware failure (e.g., computer crash)	Hardware failure (e.g., computer crash)	Medium	High
Software failure (e.g., Python crash)	Use version control (e.g., Git). Save progress frequently.	Medium	Medium
Programming problems (bugs, errors)	Regular code reviews. Utilize unit tests.	High	Medium
Running out of time	Set interim deadlines. Prioritize tasks. Seek help if needed.	High	High
Inadequate encryption techniques	Research and understand the algorithms before implementation.	Medium	High
Flask integration issues	Have a test environment. Refer to Flask documentation.	Medium	Medium
Data loss (losing backups)	Maintain multiple backup sources, both local and cloud-based.	Low	Very High

Misunderstanding of encryption techniques	Regularly consult references and seek expert advice if unsure.	Low	High
Flask security vulnerabilities	Keep Flask and its dependencies updated. Regularly check for security updates.	Low	Medium
Incorrect cryptanalysis results	Double-check results. Validate with different methods.	Medium	Medium

References

[1] D. Kahn, The Codebreakers: The comprehensive history of secret communication from ancient times to the internet, Simon and Schuster, 1996.

[2] S. Singh, The code book: the science of secrecy from ancient Egypt to quantum cryptography, Anchor, 2000.

[3] D. R. Stinson, Cryptography: theory and practice, Chapman and Hall/CRC, 2005.

[4] R. E. Lewand, Cryptological mathematics, vol. 16, American Mathematical Soc., 2000.

[5] B. Schneier, Applied cryptography: protocols, algorithms, and source code in C, john wiley & sons, 2007.

[6] W. Stallings, *Cryptography and network security principles and practices,* Prentice Hall (New Jersey), 2005.

[7] A. Ronacher, *Flask documentation,* Welcome to Flask-Flask Documentation (2.0. x). https://flask …, 2021.

[8] M. Grinberg, Flask web development: developing web applications with python, " O'Reilly Media, Inc.", 2018.

[9] S. Krug, Don't make me think!: a common sense approach to Web usability, Pearson Education India, 2000.

[10] G. Van Rossum and F. L. Drake Jr, Python tutorial, vol. 620, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[11] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks and A. Heckert, A statistical test suite for random and pseudorandom number generators for cryptographic applications, vol. 22, US Department of Commerce, Technology Administration, National Institute of …, 2001.

[12] P. E. Greenwood and M. S. Nikulin, A guide to chi-squared testing, John Wiley & Sons, 1996.