CIS7 Project Documentation Guide

In the documentation, provide at least 2 pages (single-space) that contains the following components of your course project:

1. Team name, members.
   - Paramvir Singh
2. Project Information and details: (30 points)
   - What problems are you solving in this project?
   - In a Vigenère Cipher code, I would address several problems. Firstly, I would provide a robust encryption algorithm that prevents frequency analysis attacks by introducing a keyword-based shifting mechanism. This ensures that patterns in the plaintext are disguised in the ciphertext, making it harder for adversaries to decrypt the message. Secondly, I would ensure the implementation allows for longer encryption keys, enhancing resistance to brute-force attacks. Additionally, I would aim to optimize the code for efficiency, ensuring that encryption and decryption processes are performed quickly and accurately.
   - What solutions are you implementing in the project?
   - In the Vigenère Cipher project, I am implementing several solutions to enhance the encryption process. Firstly, I am developing a robust algorithm that utilizes a keyword-based shifting mechanism, effectively disguising patterns in the plaintext and making it resistant to frequency analysis attacks. Secondly, I am incorporating the ability to use longer encryption keys, improving the security against brute-force attacks. Additionally, I am optimizing the code for efficiency, ensuring fast and accurate encryption and decryption operations. Furthermore, I am creating comprehensive documentation and user-friendly interfaces to facilitate easy usage and understanding of the Vigenère Cipher code, catering to both novice and experienced users.
   - Provide explanation of calculations and algorithm implementation.
   - Calculations that are needed Key Expansion, Shifting, and Encryption. In Key Expansion the keyword is repeated to match the length of the plaintext. In Shifting each letter in the plaintext is shifted by a corresponding letter in the expanded key. In Encryption The shifted letters are then combined to form the ciphertext.
   - The algorithm implementation takes into account the calculation of shifting values, handling of modular arithmetic, and proper handling of expanded keys. These steps ensure the encryption and decryption operations are performed accurately and consistently.
   - What is the program objectives? Explain how your program is interacting with the user and its purpose.
   - The objective of the program is to provide a user-friendly interface for encrypting and decrypting messages using the Vigenère Cipher algorithm. It interacts with the user by accepting input for the plaintext message, keyword, and the desired operation (encryption or decryption). The purpose of the program is to simplify the usage of the Vigenère Cipher, allowing users to secure their messages by easily encrypting and decrypting them. By providing a straightforward interface and automating the encryption and decryption

process, the program aims to enhance users' privacy and confidentiality in their communication.

-

  - How is discrete structures implemented in the C++ program?

- In a C++ program, discrete structures can be implemented using various data structures and algorithms. For example, graph theory concepts can be implemented by creating classes for vertices and edges, and using adjacency lists or matrices to represent the relationships between them. Set theory and combinatorics can be implemented by utilizing standard C++ libraries or by designing custom classes and functions to handle operations such as unions, intersections, permutations, and combinations. Logic and Boolean algebra can be implemented through the use of logical operators and conditional statements. Additionally, recursion can be employed to solve problems involving recursion formulas or recursive sequences.
  - What are the limitations of the program?
- The Vigenère Cipher program has several limitations that should be taken into account. Firstly, it can be vulnerable to known-plaintext attacks if an attacker has access to sufficient plaintext and corresponding ciphertext pairs. Additionally, the security of the cipher relies on the length and randomness of the keyword, making short or predictable keywords susceptible to brute-force or dictionary-based attacks. The program lacks authentication and integrity checks, meaning it cannot verify the sender's authenticity or the message's integrity. Furthermore, the limited character set support may restrict the encryption of messages containing non-standard characters.
  - Provide recommendation on improving the limitations of the program.
- Firstly, implementing a stronger encryption algorithm, such as modern symmetric key ciphers like AES (Advanced Encryption Standard), would enhance the overall security and resistance to known-plaintext attacks. Secondly, encouraging the use of longer and randomly generated keywords, preferably with a mix of uppercase and lowercase letters, numbers, and special characters, would strengthen the encryption and mitigate brute-force attacks. Adding additional layers of security, such as digital signatures or message authentication codes, could ensure the integrity and authenticity of the messages. Moreover, expanding the character set support to include a wider range of characters would enable the encryption and decryption of messages with diverse content. Finally, adopting a key management system that promotes the use of unique keys for each message, combined with secure key exchange protocols, would provide forward secrecy and protect the confidentiality of future messages.

3. Flowchart OR Pseudocode. (30 points)
   - Write the pseudocode for the program, from start to finish. Be sure to include decision-making branching.
   - If you choose to do flowchart, use standard shapes for flowchart, be sure to include decision-making branching. You can use web-based tool such as Draw.io to build your flowchart.

- 1. Prompt the user to input the plaintext message.
- 2. Prompt the user to input the keyword.
- 3. Prompt the user to choose between encryption and decryption.

```
-   4. Convert the plaintext and keyword to uppercase (to ensure consistency).
-   5. Initialize variables: ciphertext = "", keyIndex = 0.
-   6. Iterate through each character in the plaintext:
-     7. If the character is a letter:
-       8. Calculate the shifting value by subtracting 'A' from the uppercase character.
-       9. If the operation is encryption:
-         10. Shift the character by adding the shifting value of the corresponding keyword letter.
-       11. If the operation is decryption:
-         12. Shift the character by subtracting the shifting value of the corresponding keyword letter.
-       13. Handle wraparound by taking modulo 26 and adding 'A' back.
-       14. Append the shifted character to the ciphertext.
-       15. Increment the keyIndex by 1.
-       16. If keyIndex reaches the end of the keyword, reset it to 0.
-     17. If the character is not a letter, append it to the ciphertext as is.
-   18. Output the resulting ciphertext.
```