Lab Report Template

Title:Project 2 CS430

Date:4-26-23

Group Members:Braeden, Islam

Course:CS430

Instructor:Revees

**Introduction**

This project involves knowledge on the a5/1 and Tiny Encryption Algorithm (TEA). A5/1 is a stream cipher that uses 3 registers and based off rules with the registers the bits are shifted and added to leaving a keystream bit after each round of the stream. TEA is a block cipher that have simple rounds but to make it secure need lots of rounds. The purpose of this project is to gain a better understanding of TEA and how it can be applied in cryptographic scenarios. By completing this project, we'll improve our practical programming skills in computer security and encryption, which is a valuable area of study in today's digital age.

**Objectives**

Objectives of this Project are to implement the A5/1 algorithm and TEA to find a 32-bit keystream and encrypt the block text.

Implementing the Tiny Encryption Algorithm (TEA)

Understanding how the TEA algorithm works and its applications in cryptography

Learning how to use cipher block chaining mode in encryption

Developing practical programming skills in computer security and encryption

Encrypting a 64-bit plaintext block using a 128-bit key in cipher block chaining mode

**Methods**

To get the 32-bit keystream and register after for the A5/1 algorithm I used java to do string and array manipulations. For the A5/1 algorithm first the majority bit is calculated, then the xor of each register is calculated, then if the register steps then shift the bits and add the majority bit to the front of the stepped register. Then the xor of the last bit in each register will be the keystream bit.

For this project, I utilized Java programming language to implement the TEA encryption algorithm in cipher block chaining mode. I chose Java because of its strong support for cryptography through its Cryptography Architecture (JCA) and Cryptography Extension (JCE). I also used Eclipse as my development environment for its ease of use and built-in support for Java development. One of the challenges I encountered was understanding the TEA algorithm and how it could be implemented correctly in Java. However, I overcame this challenge through research and consulting relevant resources.

**Results**

A5/1

No know bugs

Output:

Ending registers

X= [0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Y= [1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]

Z= [0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1]

32 bit keystream

10000011010111111000011111100001

A screenshot of a computer

Description automatically generated with medium confidence

The project was successful in implementing the TEA algorithm and using it to encrypt a 64-bit plaintext block with a 128-bit key in cipher block chaining mode. The program generated the expected ciphertext output and met all the objectives set by the instructor. The implementation of the TEA algorithm helped to gain a better understanding of cryptographic algorithms and their applications. However, one limitation of the program is that it only works with 64-bit plaintext blocks and 128-bit keys. There are known security issues with using such small key sizes, and future work could focus on implementing larger key sizes and more secure encryption algorithms.

**Analysis**

No real insight was found in the A5/1, all straight forward, did learn the syntax for xor in java though (^).

The program written in Java uses TEA encryption algorithm in CBC mode to encrypt and decrypt messages. It has functions for encryption and decryption and takes plaintext, a key, and an initialization vector (IV) as inputs for encryption. The encrypted message is then outputted and decrypted using the same key and IV. The code is well-organized with appropriate use of functions, constants, and comments. It successfully implements the TEA encryption algorithm in CBC mode and includes methods for both encryption and decryption. While the code does not explicitly handle errors, the quality of the code suggests that the program should run smoothly without major issues.

**Conclusion**

To summarize, the lab was fun and a great coding experience into more ciphers!

The code implements the TEA encryption algorithm in CBC mode using Java. The main function encrypts and decrypts a message with a randomly generated Initialization Vector. The results are printed to the console. The code could be improved by using a more secure random number generator and by implementing padding that conforms to the PKCS#7 standard. The potential applications of this code include secure communication and data storage.

**Appendix**

The A5/1 is just one class with most of the code in the main method and there is one helper function to get the majority for the registers. All the starting registers are hard coded and there a var that is how long the key stream should be and that’s near the top in main. Project has only one class which is TEA.java. Made it clear as possible, could run without any problem and encrypts the given the 64-bit plaintext block 0x0123456789ABCDEF. Even though instructions don't mention decrypting, Professor Reeves recommended to add that method. Code runs and encrypts randomly 64-bit plaintext, and also output will show the decrypting part too which checks that program is working as expected.

The A5/1