The Warden File System

Project Engineering

Year 4

Martin Boyce

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Galway-Mayo Institute of Technology

2021/2022

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

_____

Table of Contents

Contents

Summary

This project is a full stack application that uses several key components. The front end Nextjs web application, the backend java spring boot framework which connects to the tomcat server and the MySQL database. As well as the AES [1] algorithm which is the main focus of this project.

Poster

GMIT Department of Electronic and Electrical Engineering

The Warden File System

Author: Martin Boyce G-Num: G00368112
Program: Software and Electronic Engineering, 4th year

Summary

The Warden file system is an online server side database that will allow users to store files with an AES encryption. The user will access the database through a web application front end.

Software

The main part of this System is the AES encryption, I decided to do the algorithm myself while using as few libraries as possibly.

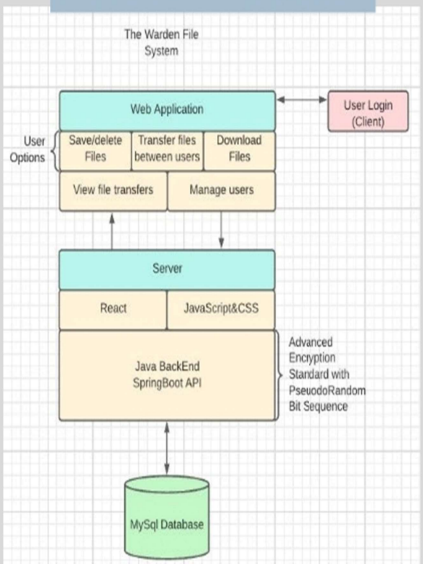
The AES encryption was written using Java, as i has a preference for it. The Web and server is programed using a react application that implements CSS, JavaScript and Nextui elements.

The order in which the System will run is as follows; The user will sign up, saving there details for login. Then they will select a file and save it on the database, the file will be encrypted before saving.

Technologies

Front End: React Web application
Server: Jave Using Spring boot to networking with the MySql Database and the frontend

Architecture Diagram



Results

GitHub:
https://github.com/Gabranth231/FinalProject

Through this project i have gained an understanding of AES cryptography and its uses in the world for data protection. The outcome of the project os a full stack web application for users to store data they wish to protect.

Applications

The Warden can be used as an online storage system for sensitive documents and details. The password hashing and file Encryption with AES will keep files safe

Made with VISME

Introduction

For my Final year Project, I went with a full stack encrypted file storage named the Warden File System. The reason behind this was based on something we learned in third year I believe, we were going over all the ways someone could use your data for malicious reasons after hacking into your account or gaining access to your sensitive data. There have been countless cases were someone's details were used against them or were robbed of their money. So having a way to save sensitive online data is something that has been a focus for a lot of people in recent times with the constant growth of data sharing and networking.

The idea for this project is so a user can save their sensitive data behind a login page and an encryption, having multiply layers of protection around their bank details and other documents. While having those options the user could also send important documents to another user for them to have access to.

The main features of this project are the server-side code that runs the AES [1] encryption and Decryption and the web page that will allow that user to interface with their files and make it so they can manage the files and how they are used. On more of the minor side is the Spring boot api that allows me to connect the front and back end using java code. For me that was something that I appreciate. We were taught how to use and run a full stack application using Node and Nextjs with a mongo database however I had some issues when I was working on that project but I'm much more comfortable with java.

I came at this project planning to do the web application first but I ran into an issue with OneDrive that set me back sometime. So instead, I redid the plan and went with the Encryption first and then the database and web page so I would have something for the Christmas demo. I used Jira and OneNote for project planning and timeline tracking. Every Week I would talk to my teammates about what we have done and the next thing to do.

The technologies used in this project are, the IntelliJ IDE for writing the java code backend, this used the Spring boot [2] api which is a java web application framework that I used to connect all the pieces of the project. I used an AES algorithm for the encryption standard, this was implemented in the backend. The Front end is coded using Visual Studios Code, I wrote it used a react framework which has Nextjs code that implements CSS and NextUi styling as well as JavaScript for the data handling.

What I have accomplished for my project is a web application that lets a user sign in, when they do that a sudo random key is created for the AES algorithm, from there they can view any text they might have and add to their storage with the help of the web application. Each entry they add is encrypted with AES 128 encryption before adding them to a table in the database.

Background of technologies implemented

AES Encryption

Advanced Encryption standard [1] is a cipher algorithm that was created in the US by the National Institute of Standards and Technology. The encryption is based around a Rijndael block cipher that uses a block of 128 bits (16 characters long) as the state and has three different key lengths to choose from; 128, 192 and 256 bits long. The key is chosen based on how in depth you want the encryption to be with a choice between 10 12 and 14 rounds of the cipher algorithm. The 128 bits long key Is used in the Warden File System.

I will go more in depth on how it works but for reference the 128 key cipher works by implementing 10 rounds on a state (128 bits) doing multiply operations to change the original data into the encrypted data.

Project Architecture

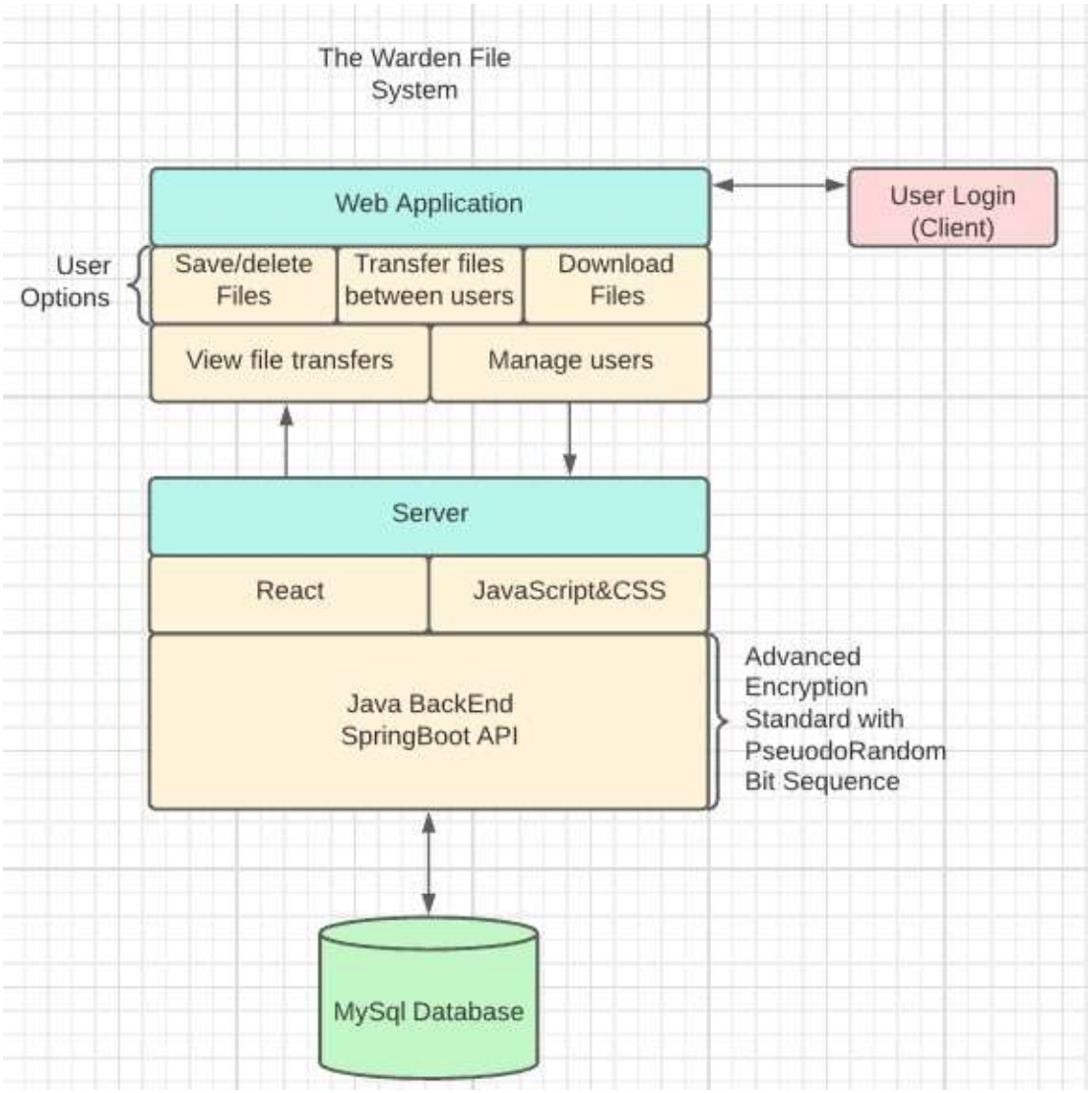


Figure 41 Architecture Diagram

Project Plan

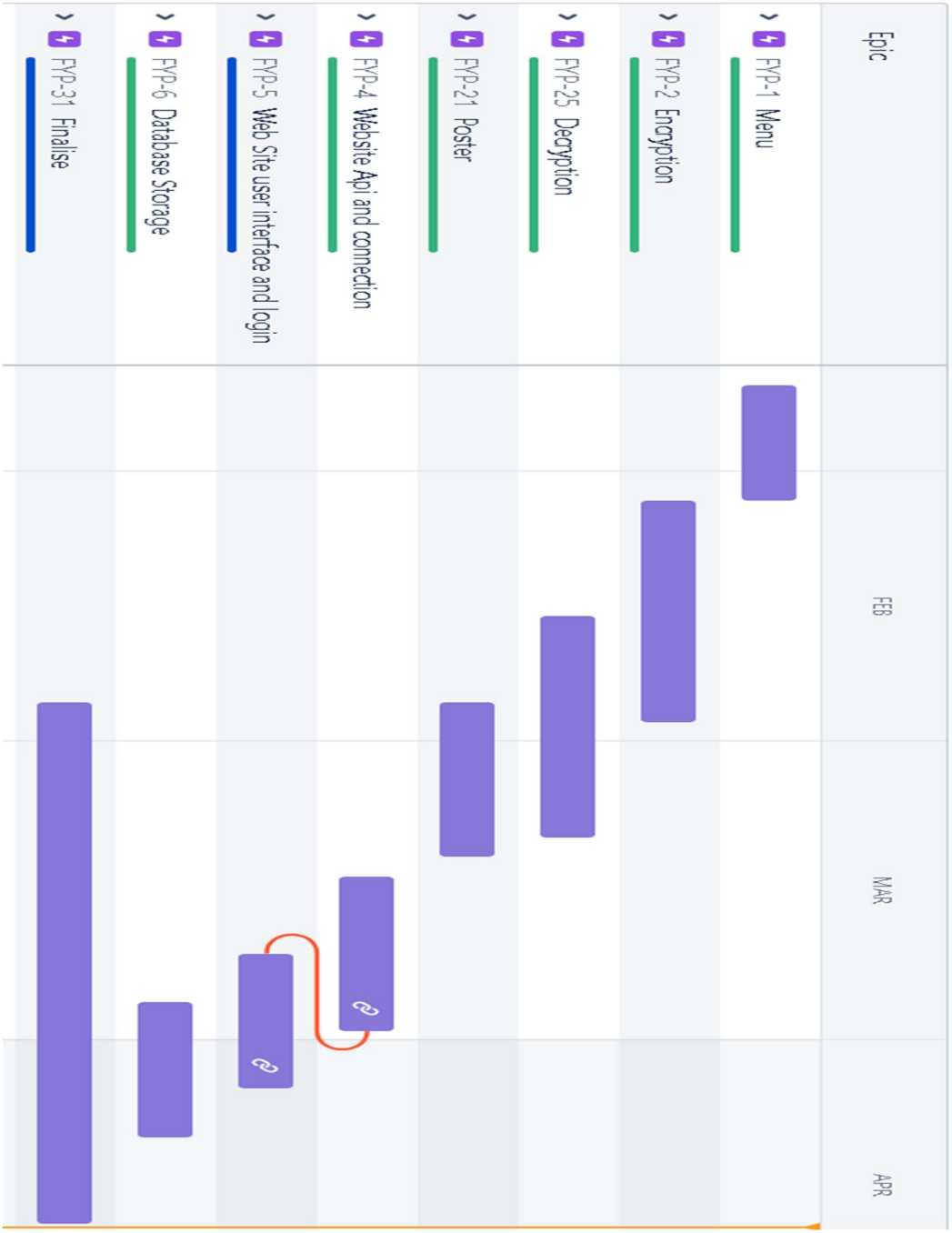


Figure 1: https://finalprojectmb.atlassian.net/jira/software/projects/FYP/boards/1/roadmap

AES Algorithm

Introduction to the cipher

The AES cipher is split into multiply into two sections, the key schedule, and the rounds. The Key schedule works by taking the 16 bytes key we generate and expanding it into a key that has 176 bytes. This is done so we have a different key for of the 10 rounds excluding the pre round which uses the original key.

There are 10 rounds in total for 128 AES encryption excluding the pre round, for the encryption the rounds go as follows. The Decryption is the below layout in reverse for the rounds.

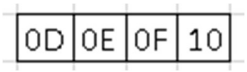
- PreRound
  - AddRoundKey
- Rounds 1-9
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- Round 10 / Final Round
  - SubBytes
  - ShiftRows
  - AddRoundKey

I had some problems getting the right layout for the cipher, when I researched for the right one there were three or four different versions.

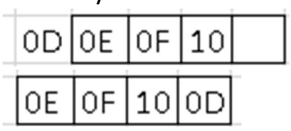
Key Schedule

The key schedule is used to expand the original key into 10 other keys, therefor the total amount of bytes for the expanded key is 176, split between keys of a size of 16 bytes. The first 16 are the original and the rest are generated as follows using 4 bytes at a time;

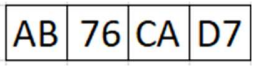
- The Expansion Core
  - Once every 16 completed bytes the core must operate to shuffle the next 4 bytes. To do this we need to take our 4 most recent bytes of the key (bytes number 13,14,15,16)

  - Above is our temporary 4 bytes that will be worked on
  - These bytes are first left shifted by one

  - After that we change out the values with their corresponding value in the Rijndael sBox value

- Then we take the first of these 4 and Xor it with its corresponding value in the RCON table

AA	76	CA	D7

- This step is done after the core and for the other bytes(core happens once every 16, this is common)
  - Next we XOR our 4 bytes with the first 4 of the most recent 16 bytes done and then add that to the end bringing the completed up to 20 bytes

01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
	XOR														
AA	76	CA	D7												
	=														
AB	74	C9	D1												

01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	AB	74	C9	D1

  - After that the next 4(highlighted blue) will only do the previous step mentioned, so they will be XORed with 05,06,07,08 and the result will be added
- This continues till we have 176 bytes.

AddRoundKey

For the Add Round Key Step we need to get our state, which is our current 16 bytes to be worked on and xor each value with its pair in a key. The key changes per round to one of the 11 keys which also has 16 bytes each. Below shows a state being XORed with a key, the pre-Round is the sample taken to show the original key and state of “Hello”.

State(PreRound)	48	65	6C	6C	6F	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Pre Round Key	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x10
After PreRound	49	67	6F	68	6A	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x10

ShiftRows/ InvShiftRows

The ShiftRows is used to mix the state up by shifting the values. This is best seen when we use a matrix to show it. The matrix is represented as a matrix column representation. This is a matrix that increments along the columns instead of the rows.

A1	A5	A9	A13
A2	A6	A10	A14
A3	A7	A11	A15
A4	A8	A12	A16

Below shows the operation for ShiftRows during the encryption. It works by left shifting by a number depending on the row number; 0,1,2,3 in that order.

			Before ShiftRows			
			0x3B	0x02	0x01	0xD7
			0x85	0x6F	0x67	0xAB
			0xAB	0xC5	0x2B	0x76
			0x45	0x30	0xFE	0xCA
			During			
			0x3B	0x02	0x01	0xD7
		0x85	0x6F	0x67	0xAB	
	0xAB	0xC5	0x2B	0x76		
0x45	0x30	0xFE	0xCA			
			After			
			0x3B	0x02	0x01	0xD7
			0x6F	0x67	0xAB	0x85
			0x2B	0x76	0xAB	0xC5
			0xCA	0x45	0x30	0xFE

To revert this in the Decryption version of ShiftRows we need to do the same operation however we right shift instead of left shift

Before ShiftRows Inv						
0x3B	0x02	0x01	0xD7			
0x6F	0x67	0xAB	0x85			
0x2B	0x76	0xAB	0xC5			
0xCA	0x45	0x30	0xFE			
During						
0x3B	0x02	0x01	0xD7			
	0x6F	0x67	0xAB	0x85		
		0x2B	0x76	0xAB	0xC5	
			0xCA	0x45	0x30	0xFE
After						
0x3B	0x02	0x01	0xD7			
0x85	0x6F	0x67	0xAB			
0xAB	0xC5	0x2B	0x76			
0x45	0x30	0xFE	0xCA			

SubBytes/ InvSubBytes

SubBytes Works by using our byte as a value to index a lookup in the Rijndael sBox. For example 0x45 would work by going to 4 in the rows and 5 in the columns giving us 0x6E

There is an inverse sBox as well, this is used during the decryption to reverse the effect. Same idea with the lookup of the byte.

At first, I had an issue with this because I didn't know that there was an Inverse sBox, for some time I was using the same one for both the decryption and encryption.

MixColumns/ InvMixColumns [3] [4]

This step was one of the more difficult problems in the algorithm. Mix Columns is done by doing Galois Field matrix multiplication on the state using a predefined table. For the encryption we multiply our state by this table. Which is predefined in the AES cipher

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Each number refers to the Galois field so 2 would be 0000 0010 and so one with the rest. Let's take for example our state after the ShiftRows step.

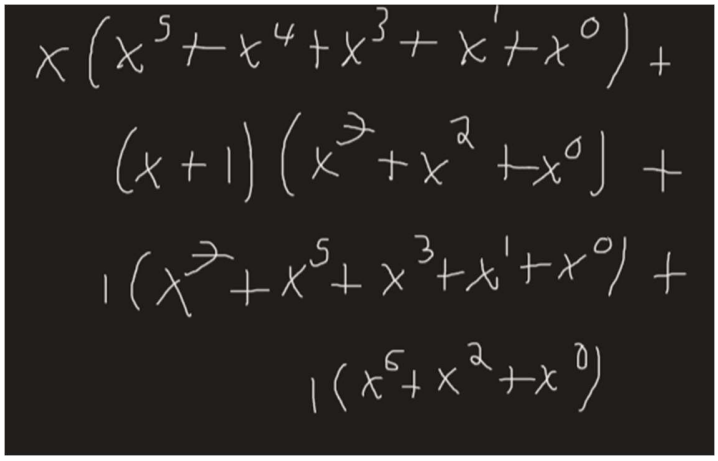
0x3B	0x02	0x01	0xD7
0x85	0x6F	0x67	0xAB
0xAB	0xC5	0x2B	0x76
0x45	0x30	0xFE	0xCA

- A1 of the new matrix is made by this formula; 2(0x3B) ^ 3(0x85) ^ 1(0xAB) ^ 1(0x45)
  - We extend this formula with Galois fields representation. To do that we convert to binary
  - (0000 0010) * (0011 1011) ^ (0000 0011) * (1000 0101) ^ (0000 0001) * (1010 1011) ^ (0000 0001) * (0100 0101)
  - Then to a polynomial, will continue with OneNote to write the formula. I apologise for the handwriting.

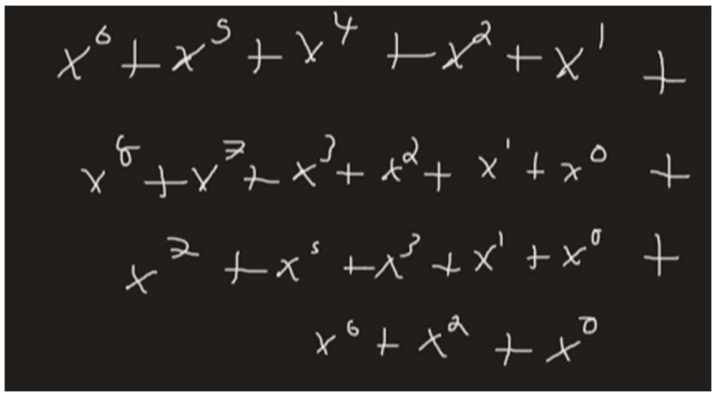
- I will only be doing one of the output fields (A1 in the matrix) as it has a lot to it.
- From here we expand the formula by changing the elements into binary numbers

(0000 0010) * (0011 1011) + (0000 0011) * (1000 0101) + (0000 0001) * (1010 1011) + (0000 0001) * (0100 0101)

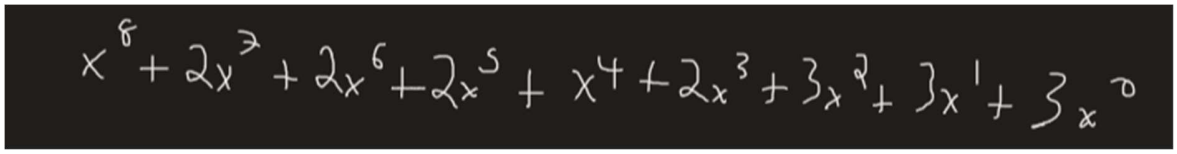
- These binary numbers are then put into a polynomial form



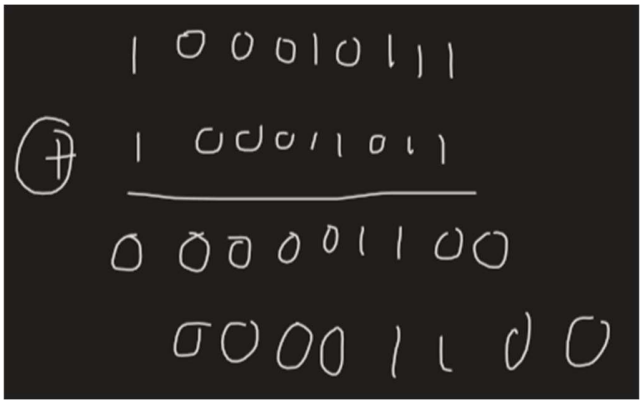
- We multiply out the equation and get



- Which we can then shorten to



- From here we need to convert to binary (evens become a 0 and odds become a 1) and mod reduce the answer to fit into a hexadecimal number using the reduction mod 100011011



- After all of that we are left with 0000 1100 which is 0x0C when converted

I will not go over the math again for the decryption, the only thing that changes is the predetermined table for the multiplication.

14	11	13	9
9	14	11	13
13	9	14	11
11	13	9	14

This is by far the step in the cipher I had the most issues with during my project. Initially I was trying to code in all the math you see above without realising that there was a shortcut by using set tables for each of the outcomes when multiplying the 1,2,3,9,11,13 or 14 with another value. I also had an issue arranging the formula correctly.

Instead of this;

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

0x3B	0x02	0x01	0xD7
0x85	0x6F	0x67	0xAB
0xAB	0xC5	0x2B	0x76
0x45	0x30	0xFE	0xCA

I had them the other way around which breaks the formula since all the pieces were multiplied by the wrong values.

Spring Boot And MySQL Database

Spring Boot is a java framework that I used to implement the backend of this project. It handles the connection between the MySQL database and the tomcat server, which handles the api calls I can use in my front end to send and receive data. It was easy to set up after some digging and I found the website which gives you the Maven project with the correct plugins prebuilt. As of yet I have not written test code using maven as the prebuilt structure given to me was not implemented before writing most of the code, so test driven development was not an option.

I choose MySQL because I have a better experience with it working for me compared to MongoDB, these two were the prevailing choices however since I wanted to use java for the backend, and we were already taught how to connect a java backend with a MySQL database the choice was obvious.

Nextjs Front end Web application

The application itself is nothing praiseworthy, I had a major setback in the first semester which cascaded into my later plans meaning I sent less time on it then I should have. With the setback, tests and other issues arising while working on the encryption I didn't get the user management done right and all the users can do is input text input a field and that text is then encrypted into the database.

Ethics

I researched on some of the ethics towards data security and collection, there is an interesting pdf created by the European Commission outlining their rules and regulations that need to be checked when creating something that stores or collects personal data. Mainly go the section 10 of the h2020 ethics and data protection pdf as well as the art.32 and art.25 of the GDPR

Conclusion

In this project I was able to implement a full stack application using a java backend and Nextjs frontend. With this I can get a user to sign into the web application and save data onto a MySQL database, the data that is being set through a JSON object is parsed and encrypted before saving it. The user details are saved on another table of the database, which stores their name and key. The key is a pseudo random string of characters 16 bytes long that is used to encrypt and decrypt the data.

References

[Advanced Encryption Standard, Online, 2001 Joan Daemen and Vincent Rijmen

[1]https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

]

[**AES MIX COLUMN MATRIX MULTIPLICATION, ONLINE LEARNING HUB,** 22 Jul 2020

[4]**https://www.youtube.com/watch?v=szwFNdf_pPo**

]

[Spring Boot Initializer

[2]https://start.spring.io/

]

[Online Domain Tools, useful web application for testing

[6]http://aes.online-domain-tools.com/

]

[Spring Boot JPA: One to many example

[7]https://www.youtube.com/watch?v=8qhaDBCJh6I

]

[Introduction to Cryptography by Christof Paar, Youtube

[3]https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg/videos

Lecture 7 and 8

]