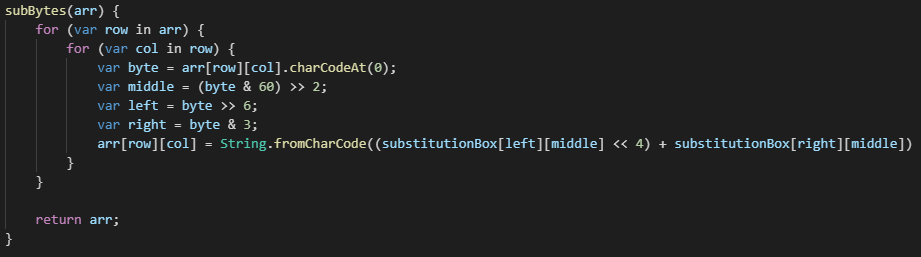
Video:

<https://drive.google.com/file/d/1LT_7uOSsGJIMesvFf-JQqacJQnyL2tRD/view?usp=sharing>

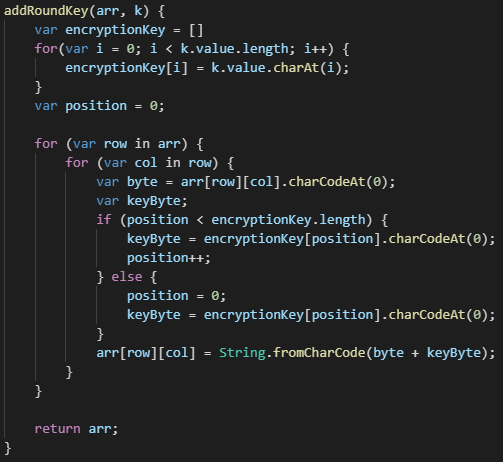
2a. My program is an encryption algorithm that encrypts a piece of text using a symmetric key. I wrote this program in Javascript and HTML using Microsoft Visual Studio Code development environment to write the code. The purpose of my program is to provide an example of data encryption using data and a key. With a symmetric key and a piece of text, data can be encrypted into unintelligible text. The video illustrates first generating a key, which is just 24 random numbers and characters. Then a message and key is input and an encrypted piece of text is output.

2b. I encountered several difficulties while writing the code for this program. One of the first problems I encountered was coding the subbytes step. I was not very skilled at bitwise functions and changing data through binary. Javascript included many bitwise functions that I did not know about. Individually, I researched these functions and learned to use them in substitution boxes for the subbytes step. I first created a 2d static array of the substitution box, where the first two and last two bits represented the row value and the middle 4 bits represented the column value. I went through every value in the data array, converted it to binary, translated the bytes to the left 2, middle 4, and then the right 2. I then matched the data values to the substitution box and converted it back to the corresponding ASCII character. Another problem I had was implementing my javascript file into the HTML webpage file. I did not have experience with scripts in HTML, so I had to individually look up how to implement javascript. Within the script tag, I created an instance of the Encryptor.js file, then matched it’s functions to the buttons on the HTML page with id tags and text boxes.

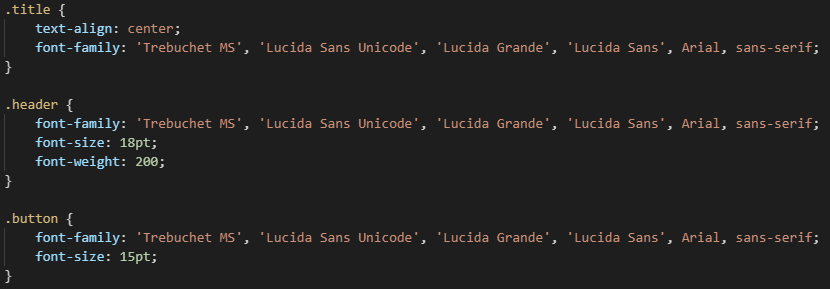
2c. I chose this algorithm below because it is one of the most important features of my program. This segment is responsible for changing the bytes of the data and replacing the old bytes with new bytes from a static table. This is achieved by first looping through every value in the 2d array of the input data that is being encrypted. Then, the character is converted to a byte. The middle 4 bits are matched to 60 in binary, which is 00111100. The left 2 bits are then chosen by shifting the 8 bit system 6 to the right, isolating the first 2 bits. The right 2 bits are then chosen by matching the data to 3, which is 00000011. This isolates the last 2 bits. Then, the 8 bit result is matched from the substitutionBox table and then converted back into the corresponding ASCII character and replaces the old value. This then connects to the other methods, as they all mutate the data from the subbytes step.



This other algorithm is essential because it is responsible for adding the values of the key to the values of the data to create a new value that is unique. Each letter in the key represents a different number to shift the original data by. After the data is changed bits, shifted rows, and then shifted columns, it is finally changed by the value of the encryption key. The code loops through every value within the 2d array, then it converts each letter of the data into binary. The next character in the key is then converted to binary, and the two binary values are added together and then converted to the corresponding ASCII character.



2d. This code implements abstraction because it condenses the amount of code needed to format the elements on the web page to allow the user to easily use the website and encrypt data. In my code, all the text uses the same font and size depending on whether it is a title, a header, or a button. Normally, I would have to add these formatting tags to every piece of text, button, and header individually. In total, each tag would triple in size. I simplified this process down by implementing classes that format based on whether the tag is in the title class, header class, or button class. This way, I just need to add the class title to the tag and it will automatically format it based on the font and size of the class.



3.

Algorithm highlighted in orange

Abstraction highlighted in yellow

**GUI.html**

<!DOCTYPE html>

<html>

<head>

<title>Data Encryptor</title>

<style>

.title {

text-align: center;

font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

}

.header {

font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

font-size: 18pt;

font-weight: 200;

}

.button {

font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

font-size: 15pt;

}

</style>

<script type='text/javascript' src="Encryptor.js"></script>

<script>

var encryptor = new Encryptor();

function generateKey() {

var txtOutput = document.getElementById("txtOutput");

txtOutput.value = encryptor.randomKey();

}

function encryptData() {

var txtInput = document.getElementById("txtInput");

var keyInput = document.getElementById("keyInput");

var txtOutputEncrypt = document.getElementById("txtOutputEncrypt");

txtOutputEncrypt.value = encryptor.encryptData(txtInput, keyInput);

}

</script>

</head>

<body style="background-color: lightgray">

<h1 class="title">Data Encryptor!</h1>

<form>

<fieldset>

<h2 class = "header">Key Generator: </h2>

<input type = "button" class = "button" value = "Generate!" onclick = "generateKey()">

<input type = "text" id = "txtOutput">

</fieldset>

</form> <hr>

<form>

<fieldset>

<h2 class = "header">Text you want to encrypt: </h2>

<input type = "text" id = "txtInput">

<h2 class = "header">Key: </h2>

<input type = "text" id = "keyInput"> <br><br>

<input type = "button" class = "button" value = "Encrypt!" onclick = "encryptData()"> <br><br>

<input type = "text" id = "txtOutputEncrypt">

</fieldset>

</form> <hr>

</body>

</html>

**Encryptor.js**

var data

var key

var substitutionBox = [

[0b0010, 0b1100, 0b0100, 0b0001, 0b0111, 0b1010, 0b1011, 0b0110, 0b1000, 0b0101, 0b0011, 0b1111, 0b1101, 0b0000, 0b1110, 0b1001],

[0b1110, 0b1011, 0b0010, 0b1100, 0b0100, 0b0111, 0b1101, 0b0001, 0b0101, 0b0000, 0b1111, 0b1010, 0b0011, 0b1001, 0b1000, 0b0110],

[0b0100, 0b0010, 0b0001, 0b1011, 0b1010, 0b1101, 0b0111, 0b1000, 0b1111, 0b1001, 0b1100, 0b0101, 0b0110, 0b0011, 0b0000, 0b1110],

[0b1011, 0b1000, 0b1100, 0b0111, 0b0001, 0b1110, 0b0010, 0b1101, 0b0110, 0b1111, 0b0000, 0b1001, 0b1010, 0b0100, 0b0101, 0b0011]

];

class Encryptor {

constructor() {

data = "";

key = "";

}

randomKey() {

var key = "";

for (var i = 0; i < 24; i++) {

var rand = Math.floor(Math.random() \* (126 - 33 + 1)) + 33;

key += String.fromCharCode(rand);

}

return key;

}

encryptData(string, k) {

data = string;

key = k;

var size = Math.ceil(data.value.length / 4);

var pos = 0;

var array = [ ];

for(var i = 0; i < 4; i++) {

array[i] = [ ];

for(var j = 0; j < size; j++) {

if (pos < data.value.length) {

array[i][j] = data.value.charAt(pos);

pos++;

} else {

array[i][j] = "5";

}

}

}

array = this.subBytes(array);

array = this.shiftRows(array);

array = this.shiftColumns(array);

array = this.addRoundKey(array, key);

var result = "";

for (var i = 0; i < array.length; i++) {

for (var j = 0; j < array[i].length; j++) {

result += array[i][j];

}

}

return result;

}

subBytes(arr) {

for (var row in arr) {

for (var col in row) {

var byte = arr[row][col].charCodeAt(0);

var middle = (byte & 60) >> 2;

var left = byte >> 6;

var right = byte & 3;

arr[row][col] = String.fromCharCode((substitutionBox[left][middle] << 4) + substitutionBox[right][middle])

}

}

return arr;

}

shiftRows(arr) {

for (var i = 0; i < arr.length; i++) {

for (var g = 1; g <= i; g++) {

arr[i].push(arr[i].shift());

}

}

return arr;

}

shiftColumns(arr) {

for (var i = 0; i < arr.length; i++) {

for (var g = 0; g <= i; g++) {

arr.push(arr.shift());

}

}

return arr;

}

addRoundKey(arr, k) {

var encryptionKey = []

for(var i = 0; i < k.value.length; i++) {

encryptionKey[i] = k.value.charAt(i);

}

var position = 0;

for (var row in arr) {

for (var col in row) {

var byte = arr[row][col].charCodeAt(0);

var keyByte;

if (position < encryptionKey.length) {

keyByte = encryptionKey[position].charCodeAt(0);

position++;

} else {

position = 0;

keyByte = encryptionKey[position].charCodeAt(0);

}

arr[row][col] = String.fromCharCode(byte + keyByte);

}

}

return arr;

}

}