EECE 455 Project Report

Dr. Ali L Hussein

Mohammad Abou Salem
Hiba Houhou
Reeda Al Saintbai
Maarouf Yassine

November 2021

1 Project Repository

`https://github.com/MoeAS/AES-Cryptography`

2 Technologies Used

The AES code was realized using JavaScript. For the interface, we developed a website using HTML and CSS, as well as JavaScript for dynamic output of the results.

3 User Interface

The interface is straightforward and user-friendly. It shows the encryption and the decryption processes in detail; the result of each round is shown to the user.

3.1 When the website is first loaded

Figure 1: The user input prompt

The user is prompted to fill in 3 fields of interest:

1. A 16-byte long hexadecimal message (plaintext or ciphertext)
2. The key size (128, 192 or 256 bits)
3. A key of length depending on the key size selected by the user

The user then has the choice to either Encrypt a plaintext or Decrypt a ciphertext using the key by clicking on the `Encrypt` or `Decrypt` buttons.

3.2 When the user Encrypts

In this section, we will test the encryption function using the plalintext `01010101010101010101010101010101010101010101` and the key `010101010101010101010...10101010101010101010101` of size 192 bits.
When the encrypt button is pressed, the page displays the final result of the encryption on the right, and the detailed encryption rounds on the left:

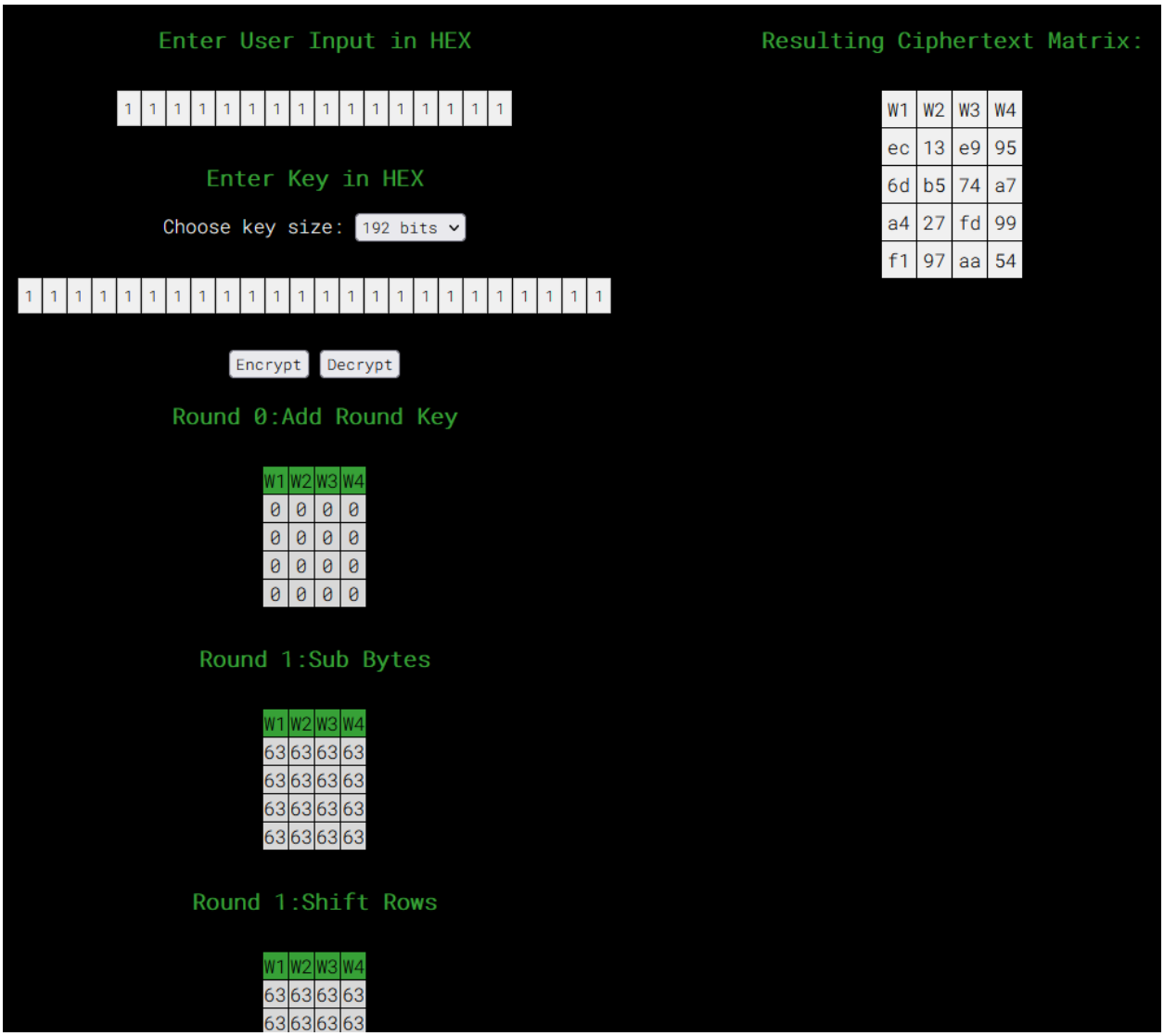

Figure 2: Page after clicking on `Encrypt`

Figure 3: Example of the encryption on the sixth round

3.3 When the user Decrypts

In this section, we will test the decryption function using the ciphertext `01010101010101010101010101010101` and the key `0101010101010101010...10101010101010101` of size 192 bits.

When the decrypt button is pressed, the page displays the final result of the decryption on the right, and the detailed decryption rounds on the left:

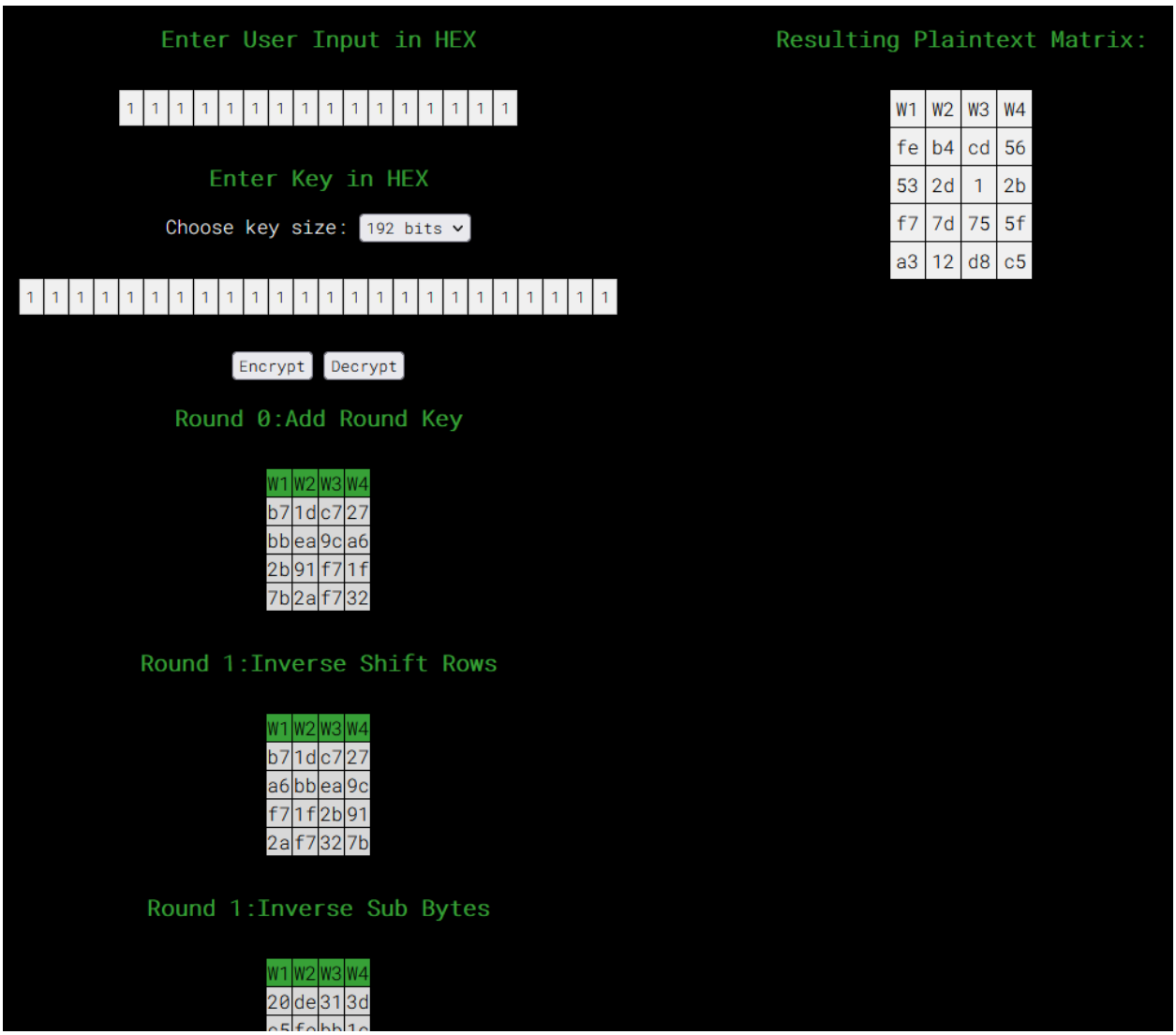

Figure 4: Page after clicking on `Decrypt`

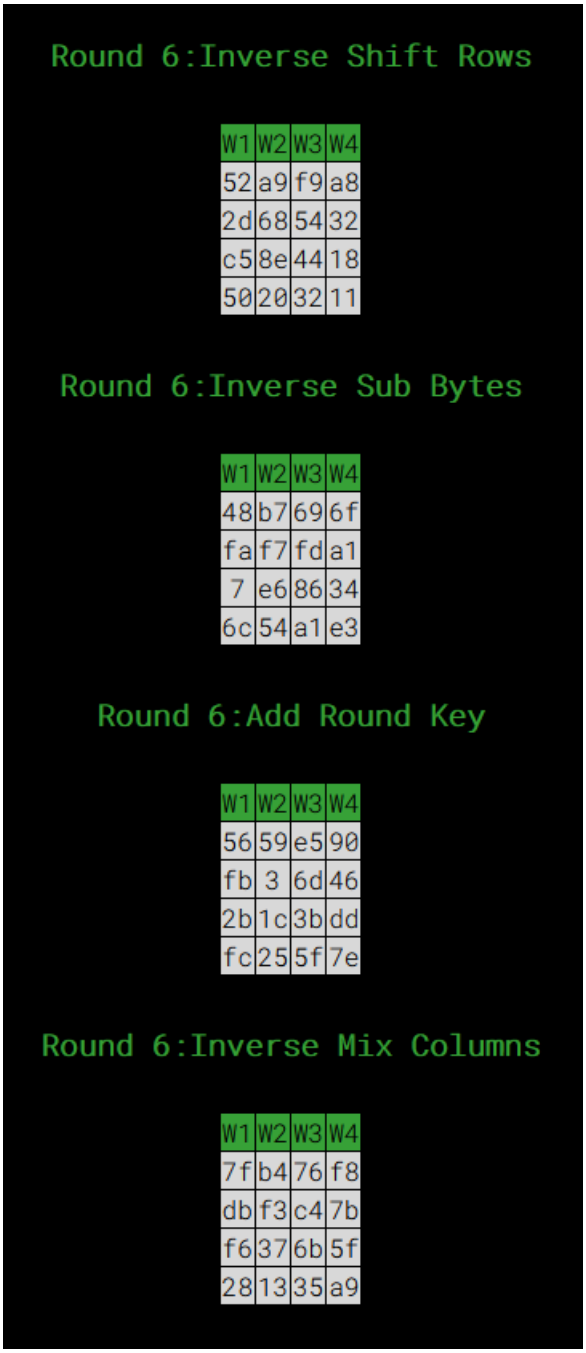

Figure 5: Example of the decryption on the sixth round

4 Code Explanation

The project folder is composed of 4 files. The file `main.js` holds the function used for the AES encryption and decryption.

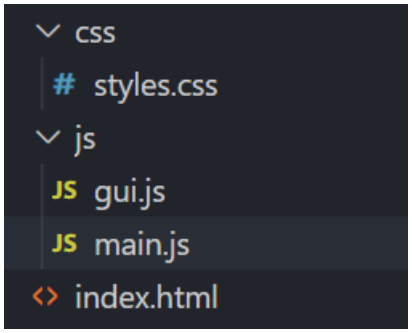

Figure 6: Project directory shot from Visual Studio

4.1 Functions in `main.js`

For each step of the encryption/decryption, we wrote a corresponding function.

4.1.1 Key expansion

```
function ExpandKey(key)
```

Figure 7: Key expansion function declaration

This functions takes 128/192/256 bit key and expands into an array of 44/52/60 32-bit words

- Starts by copying key into first 4 words
- Loops creating words that depend on values in previous and 4 places back
- Appends new key expansions of every round to the key array argument.

4.1.2 Adding the key

```
function AddRoundKey(state, rkey) {
    for (var i = 0; i < 16; i++){
        state[i] ^= rkey[i];
    }
}
```

Figure 8: Function for adding the key

This function XORs the each state with its corresponding round key.

```
function SubBytes(state, sbox) {
    for (var i = 0; i < 16; i++)
    state[i] = sbox[state[i]];
}
```

Figure 9: Function for Byte Substitution

4.1.3 Byte Substitution

This function uses a pre-defined sbox array variable (which represents the sbox table) in order to perform substitutions to the state passed to it.

4.1.4 Shift Rows

```
function ShiftRows(state, shifttab) {
    var h = new Array().concat(state);
    for (var i = 0; i < 16; i++)
    state[i] = h[shifttab[i]];
}
```

Figure 10: Function for Shift Rows

This function uses a pre-defined shifttab array variable (which represents the sbox table) in order to perform rotations on the rows of the passed state.

4.1.5 Mix Columns

This function performs mix columns operations by taking a copy of the column, ie. multiplying it by 01. Then, it performs multiplication by 02, which is stored in array b. Finally, it performs multiplication by 03 when needed and then updated the state accordingly, based on the mix columns matrix.

4.1.6 Mix Columns Inverse

This function performs the inverse of mix columns, where it multiplies a matrix (shown in fig) by the current state, outputting a new state. It was inspired by the inverse mix column function **here:** click.

```
function MixColumns(state){
    for (var c=0; c<16; c+=4) {
        var a = new Array(4);  // 'a' is a copy of the current column from 's'
        var b = new Array(4);  // 'b' is a•{02} in GF(2^8)
        for (var i=0; i<4; i++) {
            a[i] = state[c+i];
            b[i] = state[c+i]&0x80 ? state[c+i]<<1 ^ 0x011b : state[c+i]<<1;
        }
        // a[n] ^ b[n] is a•{03} in GF(2^8)
        state[c] = b[0] ^ a[1] ^ b[1] ^ a[2] ^ a[3]; // 2*a0 + 3*a1 + a2 + a3
        state[c+1] = a[0] ^ b[1] ^ a[2] ^ b[2] ^ a[3]; // a0 * 2*a1 + 3*a2 + a3
        state[c+2] = a[0] ^ a[1] ^ b[2] ^ a[3] ^ b[3]; // a0 + a1 + 2*a2 + 3*a3
        state[c+3] = a[0] ^ b[0] ^ a[1] ^ a[2] ^ b[3]; // 3*a0 + a1 + a2 + 2*a3
    }
}
```

Figure 11: Function for Mix Columns

```
function MixColumns_Inv(block) {

    for (let col=0; col<16; col+=4) {
        let v0 = block[col];
        let v1 = block[col+1];
        let v2 = block[col+2];
        let v3 = block[col+3];

        block[col] = mul_14[v0] ^ mul_9[v3] ^ mul_13[v2] ^ mul_11[v1];
        block[col+1] = mul_14[v1] ^ mul_9[v0] ^ mul_13[v3] ^ mul_11[v2];
        block[col+2] = mul_14[v2] ^ mul_9[v1] ^ mul_13[v0] ^ mul_11[v3];
        block[col+3] = mul_14[v3] ^ mul_9[v2] ^ mul_13[v1] ^ mul_11[v0];
    }
}
```

Figure 12: Function for Mix Columns Inverse

5 Running The Application

5.1 GitHub Pages

The application is deployed on GitHub pages, where you can try it on the following link: `https://maarouf-yassine.github.io/AES-Cryptography/`

0E	0B	0D	09
09	0E	0B	0D
0D	09	0E	0B
0B	0D	09	0E

Figure 13: Inverse Mix Columns Matrix

5.2 Running Locally

5.2.1 Open Index.html

Since the application is purely based on html, javascript, and css, you can open the source code and open index.html in your browser to launch the application.

5.2.2 Using VsCode Live Server

Finally, a third approach to run the application is as follows:

- Install Live Server Extension to VsCode.
- Open the project folder.
- Open the index.html with live server.
- The application will be deployed to the local host and will automatically open(usually on port 5501 if not occupied)

6 References

- `https://www.kavaliro.com/wp-content/uploads/2014/03/AES.pdf`
- `https://www.cryptool.org/en/cto/aes-step-by-step`
- `https://github.com/bozhu/AES-Python/blob/master/aes.py`