Final Report
Satvik Kulshreshtha

Course: CSE 178 – Computer Network Security
Instructor: Professor Mukesh Singhal
TA: Mina Naghshnejad

1. Description: Implementation of Advanced Encryption Algorithm(AES) for 128-bit key. AES is an iterated symmetric block cipher, which means that:
   i. AES works by repeating the same defined steps multiple times.
   ii. AES is a secret key encryption algorithm.
   iii. AES operates on a fixed number of bytes

   AES as well as most encryption algorithms is reversible. This means that almost the same steps are performed to complete both encryption and decryption in reverse order. The AES algorithm operates on bytes, which makes it simpler to implement and explain.

2. Language Used: C++

3. Implementation follows the steps of algorithm which includes:
   i. Taking input as plain text
   ii. Using 128-bit key
   iii. Padding if number of bits is not equal to 128
   iv. Key expansion:
   The key expansion takes 128-bit key and expands it such that we have a new key for each round. The core key expansion consists of 3 parts:
      a) Rotate (left)
      b) S-box substitution: we swap each byte with the corrosponding value from the sbox.
      c) R-con: We raise 2 to the power i-1 where "i" is the iterator number. We then add this to the value of the first byte in the 4, i.e. the first byte becomes itself+2^{i-1}.
   v. AES Encryption (which includes the following steps):
      a) **Byte Substitution**: During encryption, each value of the state is replaced with the corresponding SBOX value
      b) **Shift rows:** Arranges the state in a matrix and then performs a circular shift for each row. This is not a bit wise shift. The circular shift just moves each byte one space over. A byte that was in the second position may end up in the third position after the shift. The circular part of it specifies that the byte in the last position shifted one space will end up in the first position in the same row.
      c) **Mix Column**: In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher. During this operation, each column is multiplied by the known matrix that for the 128-

bit key. The multiplication operation is defined as: multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing xor with the initial unshifted value. After shifting, a conditional xor with 0x1B should be performed if the shifted value is larger than 0xFF. In more general sense, each column is treated as a polynomial over **GF**(2^8) and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from **GF**(2)[x]. The MixColumns step can also be viewed as a multiplication by a particular MDS matrix in a finite field. This process is described further in the article Rijndael mix columns.

d) **Add round keys:** Each of the 16 bytes of the state is XORed against each of the 16 bytes of a portion of the expanded key for the current round. The Expanded Key bytes are never reused. So once the first 16 bytes are XORed against the first 16 bytes of the expanded key then the expanded key bytes 1-16 are never used again. The next time the Add Round Key function is called bytes 17-32 are XORed against the state.

The first time Add Round Key gets executed

State	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
Exp Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

The second time Add Round Key is executed

State	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
Exp Key	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

And so on for each round of execution.

vi. Converting to hex and producing encrypted output

4. Results:
   i. Test Case 1**:** (Screenshots attached)
   Input text (Plain Text): this is the message to be encrypted
   Key (Plain Text): a b c d e f g h i j k l m n o p

   ii. Test Case 2: (Screenshots attached)
   Input text (Plain Text): CSE 178 lab
   Key (Plain Text):1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

   iii. Test Case 3: (Screenshots attached)
   Input text (Plain Text): thats my kung fu
   Key (Plain Text): i n p u t k e y f r o m u s e r

5. Issues:
   Using <conio.h> on mac OS (restricting in taking input from user on console)

6. References:
   https://kavaliro.com/wp-content/uploads/2014/03/AES.pdf
   http://www.infosecwriters.com/Papers/ABerent_AESbyExample.pdf
   https://en.wikipedia.org/wiki/Advanced_Encryption_Standard#Implementations
   https://www.youtube.com/
   Textbook: Cryptography and Network Security Principles and Practice(5thEdition)