**Advanced Encryption Standard**

Advanced Encryption Standard was created using Rijndaels’ method. Text and key are assumed 128bits or 16 bytes and are stored in an unsigned char array. For testing purposes two sets of Texts are being encrypted by two different keys. In the main method the commented out section can be used to take user Text input as well as user Key input. Everything is handled as unsigned char. and then the final product is converted to hex when printing out in order to properly display the encrypted message as the char forms can sometimes be unreadable by the terminal.

Once the text and key arrays are filled it calls the encrypt function. This function starts by expanding the key. We need to add a round number to the text before starting the 10 round of substituting, shifting rows, mixing columns, and adding round keys. For the round key we XOR they key with the initial data. We then enter a for loop to run the cycle 9 times.

**Byte Substitution**

For Byte Substitution, we use an array sbox of 256 hex values. We take the values from the text and pass them into a substitution function 4 bytes at a time. We take the 4 bytes and exchange them with the corresponding value in the sbox. Since the arguments are handled as pointers there is no need to return anything.

**Row Shifting**

Shifting rows is done in a circular shift as opposed to bit wise shifting. From the for loop, shift\_rows function is called and passes in 4 bytes of the text. The substitution is done in the form of a 2d matrix, the text is filled into it vertically but the substitution happens horizontally. Once in the function each byte is shifted one space over. The first 4 bytes are left alone, the second set is shifted once, the third set is shifted twice and the last set is shifted three times.

Text: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 will form

Before After

1 5 9 13 -> 1 5 9 13

2 6 10 14 -> 14 2 6 10

3 7 11 15 -> 11 15 3 7

4 8 12 116 -> 8 12 16 4

**Column Mixing**

Column mixing is achieved by taking a single column of 4 bytes and performing matrix multiplication on a Galois multiplication field. The mix columns function takes in 4 bytes of the text and each of these is saved onto an array to be used in the multiplication. Using the Galois multiplication and XOR mixing columns can be achieved using in such manner:

r[0] = gmul(a[0],2) ^ gmul(a[3],1) ^ gmul(a[2],1) ^ gmul(a[1],3);

r[1] = gmul(a[1],2) ^ gmul(a[0],1) ^ gmul(a[3],1) ^ gmul(a[2],3);

r[2] = gmul(a[2],2) ^ gmul(a[1],1) ^ gmul(a[0],1) ^ gmul(a[3],3);

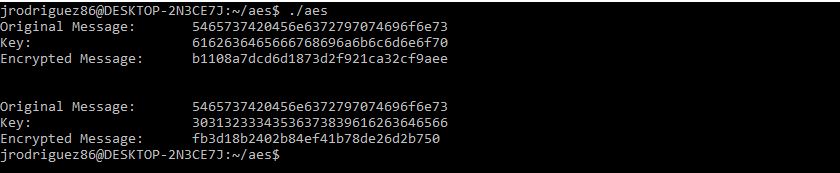
r[3] = gmul(a[3],2) ^ gmul(a[2],1) ^ gmul(a[1],1) ^ gmul(a[0],3);

The end result of the multiplications and XOR will be a hex number of no more than two digits.

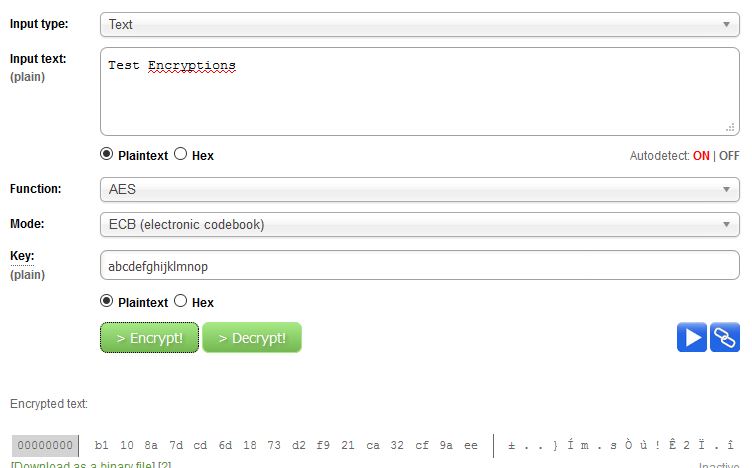
**Generate Round Key**

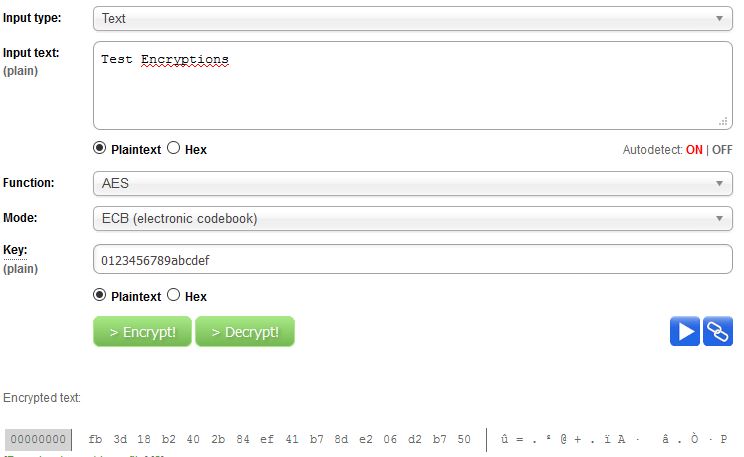
Each of the 16 bytes of the text is XORed against each of the 16 bytes of a portion of the expanded key. Every time the getroundkey function is called it switches the 16bytes it will be using. The first time it will use byte 1-16 of the expanded key, the next time it will use byte 17-32 to be XORed against. The purpose of memcpy(roundkey, expandedkey + which\*16, 16); is to move the bytes that are being used.

From aes.cpp



From aes.online-domain-tools.com





Sources Used

* <http://www.infosecwriters.com/Papers/ABerent_AESbyExample.pdf>
* <http://www.cs.utsa.edu/~wagner/laws/AESkeys.html>
* <http://www.samiam.org/key-schedule.html>
* <http://aes.online-domain-tools.com/>