**CMPE 494 Project 1 Report**

**AES Algorithm:**

AES (Advanced Encryption Standards) is a 128-bit symmetric block cipher which means the plain-text is divided into 128 bit blocks and each block is encrypted by using the algorithm, on ECB (Electronic Code Book) mode. In the plain-text each line is 32 hex characters long which represent 16 2-digit hex characters which are then filled into a 4x4 state matrix. For the encryption process, the key is expanded using the key scheduling into 10, 12 or 14 round keys depending on the size of the key (128, 192 or 256 bits). Then the first round key is added to the state matrix. Then the iterative process is repeated for n = (number of roundkeys – 1) times. It consists of subBytes, shiftRows, mixColumns and addRoundKey methods. Then for the final step mixColumns method is skipped as permuting the output has no effect on the outcome (increases the running speed). Then the final state is converted to the string and it is the encrypted version of each block. Decryption process is the backwards of this process, which is almost the same.

**Running the Algorithm:**

In order to run the algorithm, be sure that all AES.java, Makefile, your keyFile and your inputFile are on the current directory. Then, compile the code using “make” command. After compilation, use the format below to encrypt or decrypt your inputFile based on your encryption string in the keyFile:

java AES e|d keyFile inputFile

The algorithm checks for all 3 arguments on the command line, so be sure that all arguments are passed into the code, i.e. keyFile and inputFile variables are taken from the command line, files like “keyFile” are not checked. This information is given according to the description on moodle: “... must be named as keyFile, no extension ...”. The code works fine with the files with extensions, for example input.txt is encrypted to input.txt.enc and it is decrypted to input.txt.enc.dec.

**Description of the methods:**

* subBytes  
    
  An S-box derived from multiplicative inverse over GF(2^8, Galois Field) is used to ensure non-linearity. It is a 16x16 matrix of byte values that contains a permutation of all possible 256 8-bit values. Each value in the initial state matrix is substituted with it’s value on the table. We used a lookup table in our implementation for a faster run time.
* shiftRows  
    
  Each line on the state is shifted to the left n times, where n is the index of that line.
* mixColumns  
    
  Each column of state is multiplied (matrix dot product) with Galois table and rewritten onto the same column of state.
* addRoundKey  
    
  At each step, sub key (calculated on keySchedule method) and state are combined (bitwise xor) together to get new state.
* keySchedule  
    
  Using the encryption key string, 11 different sub keys are produced whereas the first one is the original encryption key string that is converted onto a 4x4 matrix. Number of sub keys change if AES-128, AES-192 or AES-256 is used (11, 13 or 15 sub keys respectively).

**Encryption Speed:**

For each operation, the number of primary operations done for 128-bit key (assuming the value that is written onto the state matrix as the primary operation):

16 (subBytes)

16 (addRoundKey)

16 (subBytes)

64 (mixColumns)

So for the whole block:

16 (initial add roundKey)

9\*102 ( one round)

48 ( the final round without mixColumns)

= 998 operations in total

With the CPU speed of x operations/sec, a 128-bit line is encoded in 998/x seconds which gives us the x/998 for the theoretical speed of the algorithm. Substituting 3.00\*10^9 for x (or 3 GHz CPU), we get around 3.0 \* 10^6 blocks/sec.

Each block is 128-bit (32 hex characters), which is 16 Bytes. Then, after calculations, our encryption speed is **45.9 MB/sec**.

**Conclusion:**

AES-128 algorithm (and AES-192, AES-256) or Rijndael Algortihm is still being widely used in many fields, especially in information security. Governments, companies and military foundations frequently use this algorithm. There are several reasons behind that:

* Ease of implementation on both hardware and software
* Runing speed
* Being memory-efficient
* Flexibility (applicable on many systems)
* Strength of security

The most important reason amongst them is the security strength, because when Rijndael Algorithm is implemented for the first time, encrypted files that are larger than the ones on the past became harder to be decrypted without the encryption key string. The hardness of being decrypted depends on the substitution box (subTable or SBox), and there are many articles on how SBox affects the security strength of the algorithm.