An Efficient Approach for Increasing Security to Symmetric Data Encryption

[1]Muhammad Nazrul Islam, [1]Md. Monir Hossain Mia, [1]Md. Foizul Islam, [2]M.A. Matin [1]
Dept. of Computer Science & Engineering
[1]Khulna University of Engineering & Technology (KUET)
[2]BRAC University

Summary

The selective application of technological and related procedural safeguards is an important responsibility of every organization in providing adequate security to its electronic data systems. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data. The algorithm uniquely defines the mathematical steps required to transform data into a cryptographic cipher and also to transform the cipher back to the original form. Data encryptions standard (DES) use64 bits block size as well as 64 bits key size that are vulnerable to brute-force, attack. But for both efficiency and security a larger block size is desirable. The Advanced Encryption Standard (AES), which use 128 block size as well as 128 bits key size, is using as an encryption standard now. In this paper, we propose an algorithm which is higher secure than Rijndael algorithm but less efficient than that. The difference of efficiency between Rijndael and our propose algorithm is very negligible. We explain all this term in this paper.

KEY WORDS: Computer Security, Block cipher, plain text, cipher text, Differential cryptanalysis, Linear cryptanalysis, Symmetric Encryption.

1. Introduction

In this document we describe about symmetric cipher. Our proposed algorithm is much more similar to that of Rijndael. The difference is that, Rijndael algorithm start with 128 bits block size, and then increase the block size by appending columns[Rijn99], whereas our algorithm start with 200 bits. Our paper is organized as follows.

Section 2 describes the algorithm properly and section 3 gives a way of thinking in security measure. We gave the time comparison between the original Rijndael implementation on 128 bits block size (with 128 bit key) and our proposed algorithm which operate on 200 bits in section 4. Some advantages and disadvantages are given in section 5 and we conclude in section 6. We did not gave the mathematical preliminaries as it is same as the mathematical computation of Rijndael.

2. Proposed Algorithm

For simplicity, we refer the different transformation; operate on the intermediate result as State.

2.1 The General Definitions

The intermediate cipher result is called the State. The State can be pictured as a rectangular array of bytes. This array has five rows; the number of columns is denoted by **Nb** and is equal to the block length divided by 40.

As the security is the function of block length and the size of key length we increase the block length as well as the key length. Our basic block length is 200 bits which can be shown as a 5 by 5 matrix of byte. This is illustrated in figure 1.We can increase our block by appending a column at a time. But we like to emphasize on 200 bit and then compare the security & efficiency between our 200 bits block cipher and Rijndael 128 bits cipher. The input and output used by our proposed algorithm at its external interface are considered to be one dimensional arrays of 8-bit bytes numbered upwards from 0 to the 5***Nb**-1.The Cipher Key is considered to be a one-dimensional arrays of 8-bit bytes numbered upwards from 0 to the 5***Nk**-1.The cipher input bytes (the "plaintext" if the mode of use is ECB encryption) are mapped onto the state bytes in order

$a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{4,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$, $a_{4,1}$ ... , and the bytes of the Cipher Key are mapped onto the array in the order $k_{0,0}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{4,0}$, $k_{0,1}$, $k_{1,1}$, $k_{2,1}$, $k_{3,1}$, $k_{4,1}$ ... At the end of the cipher operation, the cipher output is extracted from the state by taking the state bytes in the same order.

Hence if the one-dimensional index of a byte within a block is n and the two dimensional index is (i ,j), we have:

$$i = n \bmod 5 ; \qquad j = \lfloor n/5 \rfloor ; \qquad n = i + 5 * j$$

2.2 The Round Transformation

The round transformation is composed of four different transformations. It is similar to that of Rijndael. In pseudo C notation we can represent this as below-

```
Round(State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);}
```

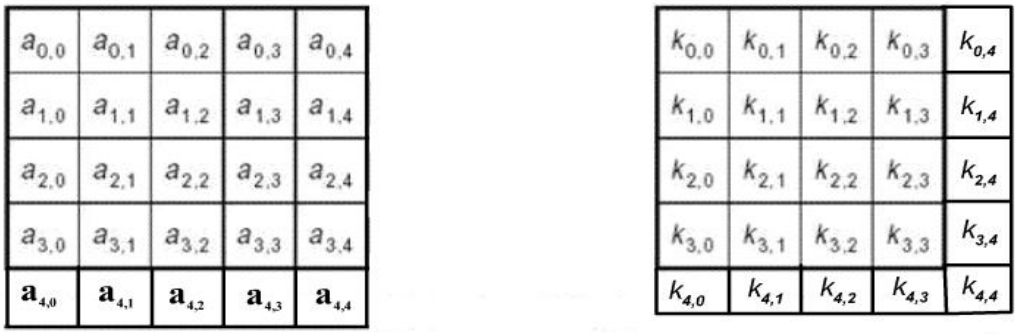

Fig 1. Matrix representation of data block and key

The final round of the cipher is slightly different. It is defined by-

```
FinalRound(State,RoundKey)
{
ByteSub(State) ;
ShiftRow(State) ;
AddRoundKey(State,RoundKey);
}
```

In this notation, the "functions" (Round, ByteSub, ShiftRow, …) operate on arrays to which pointers (State, RoundKey) are provided. It can be seen that the final round is equal to the round with the MixColumn step removed. The component transformations are specified in the following subsections.

2.2.1 The ByteSub Transformations

The bytesub transformation is similar as that of Rijndael bytesub transformation.The details is given at [Rijn99]. For increasing the efficiency we used Rijndael S-box.

2.2.2 The ShiftRow Transformations

For encryption, the 1st row remain unchanged, 2nd row is shifted 1 byte to the left, 3rd is 2 byte to the left, 4th is 3 byte to the left and 5th row is shifted 4 byte to the left. For decryption the operation is similar to that for encryption but in reverse direction.

2.2.2 The MixColumn Transformations

In MixColumn, the columns of the State are considered as polynomials over GF(2^8) and multiplied modulo $x^5 + 1$ with a fixed polynomial c(x), given by

$c(x) = '04'\ x^4 + '03'\ x^3 + '01'\ x^2 + '01'\ x + '02'$

This polynomial is co-prime to $x^5 + 1$ and therefore invertible. This can be written as a matrix multiplication. Let $b(x) = c(x) \otimes a(x)$,

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} 02 & 04 & 03 & 01 & 01 \\ 01 & 02 & 04 & 03 & 01 \\ 01 & 01 & 02 & 04 & 03 \\ 03 & 01 & 01 & 02 & 04 \\ 04 & 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

The application of this operation on all columns of the State is denoted by MixColumn(State).

The inverse of MixColumn is similar to MixColumn. Every column is transformed by multiplying it with a specific multiplication polynomial d(x), defined by

(‘04’ x^4 + ‘03’ x^3 + ‘01’ x^2 + ‘01’ x +’02’) ⊗d(x) = ‘01’.

It is given by:

d(x) = ‘7D’ x^4 + ‘09’ x^3 + ‘8A’ x^2 + ‘4C’ x +’E0’.

2.2.4 The Round Key Addition

In this operation, a Round Key is applied to the State by a simple bitwise EXOR. The Round Key is derived from the Cipher Key by means of the key schedule. The transformation that consists of EXORing a Round Key to the State is denoted by: AddRoundKey(State,RoundKey).

2.3 Key Schedule

The Round Keys are derived from the Cipher Key by means of the key schedule. This consists of two components: the Key Expansion and the Round Key Selection. The basic principle is the following:

- The total number of Round Key bits is equal to the block length multiplied by the number of rounds plus 1.
- The Cipher Key is expanded into an Expanded Key.
- Round Keys are taken from this Expanded Key in the following way: the first Round

2.3.1 Key Expansion

The Expanded Key is a linear array of 5-byte. In c code this can be written as

```
keyexpansion(unsigned short int *key,unsigned short int *expandkey)
{
    unsigned short int temp[5],*temp1;
    int i,j;
    for(i=0;i<5;i++){
        for(j=0;j<5;j++){
            expandkey [i*5+j]=key[i*5+j];
        }
    }
    for(i=5;i<55;i++){
        for(j=0;j<5;j++)
            temp[j]= expandkey [(i-1)*5+j];
        if(i%5==0){

temp1=subword(rotbyte(temp));
            for(j=0;j<5;j++)
                temp[j]=temp1[j];
            temp[0]=temp[0] ^ Rcon[i/5-1];
        }
        for(j=0;j<5;j++)
            expandkey [i*5+j]= expandkey [(i-5)*5+j]^temp[j];
    }
}
```

One important this is here we expand key for 10 round. And the rotbyte and subbyte is stand for rotation of byte in a single vector and substitute a byte using S-box. The detail about Rcon is given in [Rijn99].

3. Security

As we increase the key size as well as the block size the security has enhanced. And the linear cryptanalysis and differential cryptanalysis require more time then Rijndael to break our proposed cipher.

4. Comparison

The amount of time required to encrypt a packet is proportional to the number of bytes (as well as the number of bits) in the packet. If the packet size is 200bits long, then our proposed algorithm has to execute once to encrypt the whole data but Rijndael algorithm has to run 2 times to encrypt the whole data. In a common sense, it seems that our proposed algorithm is more efficient. But actually it is less efficient. The difference of the efficiency is very negligible. But our proposed cipher is much efficient than saffer+, RC5. A comparison of various types of Encryption and Decryption algorithm is given in [PCAS99]. The simulated result is given below through the figure 3 and 4 .

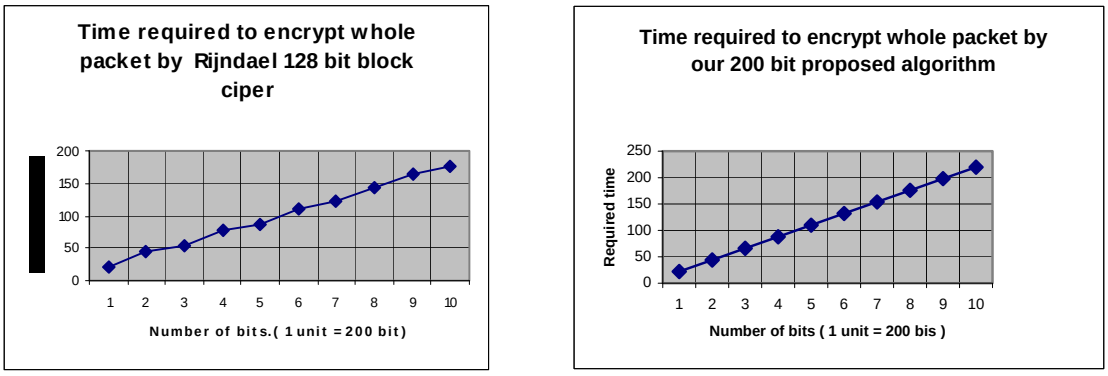

Fig. 3 Timing comparison to encrypt between Rijindael and our proposed algorithm

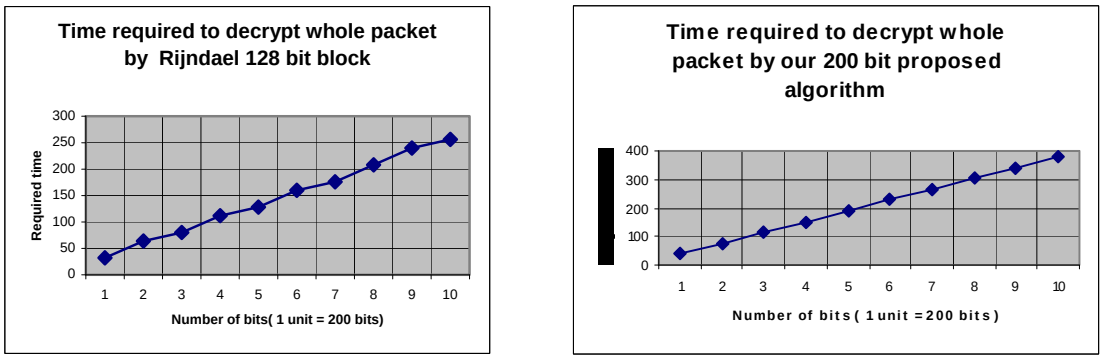

Fig. 4 Timing comparison to decrypt between Rijindael and our proposed algorithm

5. Advantages and Limitations

Implementation aspects:

- Our proposed algorithm can be implemented to run at speeds unusually fast for a block cipher on a Pentium (Pro). There is a trade-off between table size / performance.
- Our proposed algorithm can be implemented on a Smart Card in a small amount of code, using a small amount of RAM and taking a small number of cycles. There is some ROM/ performance trade-off.
- Our proposed algorithm can be used in Geographic Information System (GIS) and Satellite Communication when huge data need to be transferred securely.
- The round transformation is parallel by design, an important advantage in future processors and dedicated hardware.
- As the cipher does not make use of arithmetic operations, it has no bias towards big or little ending processor architectures.

Simplicity of Design:

- The cipher is fully "self-supporting". It does not make use of another cryptographic component, S-boxes "lent" from well-reputed ciphers, bits obtained from Rand tables, digits of p or any other such jokes.
- The cipher does not base its security or part of it on obscure and not well understood interactions between arithmetic operations.
- The tight cipher design does not leave enough room to hide a trapdoor.

Variable block length:

- Although the number of rounds of Rijndael is fixed in the specification, it can be modified as a parameter in case of security problems.

Limitations:

The limitations of the cipher have to do with its inverse:

- The inverse cipher is less suited to be implemented on a smart card than the cipher itself: it takes more code and cycles. (Still,

compared with other ciphers, even the inverse is very fast).
- In software, the cipher and its inverse make use of different code and/or tables.
- In hardware, the inverse cipher can only partially re-use the circuitry that implements the cipher.

- This algorithm is less efficient then Rijndael algorithm.

6. Conclusions

Though there is some difference in efficiency between our proposed algorithm and Rijndael, it is negligible as it varies in microsecond. If any user emphasis on security then he can use our proposed algorithm. But if the efficiency comes first then the user must use the Rijndael algorithm. And we can claim that if the block size is increase by increasing the matrix order then there is degradation in efficiency.

References

[1] Joan Daemen and Vincent Rijmen,AES submission document on Rijndael,June 1998.

[2] [Rijn99]Joan Daemen and Vincent Rijmen, AES submission document on Rijndael,Version 2, September 1999.

[3] Joan Daemen and Vincent Rijmen, The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag 2002 (238 pp.)

[4] [DaKnRi97] J. Daemen, L.R. Knudsen and V. Rijmen, "The block cipher Square," Fast Software Encryption, LNCS 1267, E. Biham, Ed., Springer-Verlag, 1997, pp. 149-165.

[5] [KeScWa96] J. Kelsey, B. Schneier and D. Wagner, "Key-schedule cryptanalysis of IDEA, GDES, GOST, SAFER, and Triple-DES," Advances in Cryptology, Proceedings Crypto '96,LNCS 1109, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 237-252.

[6] [PCAS99]Bruce Schneier, John Kelsey, Doug Wagner, Chris hall, Niels Ferguson "Performance Comparison of the AES Submission"

[7] [CNSPP03]Willam Staling "Cryptography and network security-principle and practice"

Muhammad Nazrul Islam received the B.Sc. degree in Computer Science and Information technology (CIT) from Islamic University of Technology (IUT), Bangladesh and M.Sc. degrees in Computer Engineering from Politecnico di Milano, Italy in 2002 and 2007, respectively. He has been servicing as a faculty member of the department of Computer Science & Engineering (CSE) of Khulna University of Engineering & technology (KUET), Bangladesh since July, 2003. He is currently working as an Assistant Professor of CSE department of KUET, Khulna 9203, Bangladesh.

Muhammad Foizul Islam received the B.Sc. degree in Computer Science and Engineering (CSE) from Khulna University of Engineering and Technology (KUE), Bangladesh. in 2004. He has been servicing as a faculty member to the department of Computer Science & Engineering (CSE) of Leading University, Sylhet, Bangladesh since January, 2005. He is currently working as a Research Assistant in Ontario Research Center for Computer Algebra (ORCCA) lab, Canada. Besides, he is doing his M.Sc. in Computer science at department of Computer Science, University of Western Ontario, Canada.

Md Monir Hossain Mia received the B.Sc. Engr. degree in Computer Science and Engineering (CSE) from Khulna University of Engineering and Technology (KUE), Bangladesh. in 2004. He has been servicing as a faculty member to the department of Computer Science & Engineering (CSE) of University of Information Technology & Sciences (UITS), Dhaka, Bangladesh since June, 2005 (Study Leave). He is currently working as an ICT Technician in St Charles College, UK. Besides, he has recently finished his M.Sc. in Mobile Computing & Communications from University of Greenwich London, UK.