Bitwise Operations Based Encryption and Decryption

K.Naveen Kumar
Dept of IT
GITAM UNIVERSITY
Visakhapatnam

G.V.S.Raj Kumar
Dept of IT
GITAM UNIVERSITY
Visakhapatnam

K.T.Praveen kumar
Visakhapatnam

P.Chandra Sekhar
Dept of IT
GITAM UNIVERSITY
Visakhapatnam

Abstract

In this paper, variable block length based character/bit level transformation has been proposed for encryption and decryption, where a block of characters /few bits has taken into account. The originated text is transformed into decimal values and passed to encoder module. In this module values are cluttered using binary operations present in most systems with keys. This routine is based on feistel iteration using large number of rounds for security justification to certain intensity and passed to the security module. This module performs conversion of values into octal base values and then padded with a vector. When compared with other encryption schemes it provides more security even though keys are revealed. Since, one parameter namely vector a_i alone been kept secret. The receiver passes the received cipher text to security module and yields the cipher decimal values. These values are transformed into a form of originated text.

Keywords: feistal, XOR, octal, block length,

I. INTRODUCTION

The need for security in systems is increasing tremendously during last few decades. Securing electronic data is gradually becoming important with the increasing dependency on the data interchange by the internet, because of general human tendency to access something even by violating limits and laws.

In order to fulfill those requirements in security aspects a few encryption/ decryption algorithms are available in [1,2,3,4,5],but none of the algorithm can be the ultimate solution and every algorithm have its own merits and demerits. The proposed method provides a secured algorithm thereby discouraging a potential cryptanalyst from attempting a brute force attack. The main design goal was to produce a cipher that is simple, short and does not rely on large tables or pre-computations. The main routine requires two fewer addition operations which results in a faster algorithm process even for devices with low speed of operations and lower bit operation registers [6].

This uses only simple addition, XOR and shifts, and has a very small code size that makes the proposed model an ideal model to provide data security services. Due its simple design makes it also very suitable for hardware implementations. The implementation designers can take advantage of the inherent parallelism to boost performance or on the other hand, reduce its area and power consumption. This makes it possible to implement for severe power constraint applications such as passive Radio Frequency Identification (RFID) tags and the next generation WSN nodes[7]. Customized hardware is currently the only option for inexpensive passive RFID tags as they do not have a processor. The same might become true for next generation sensor nodes which are predicted to be powered by energy scavenged from the environment. For environments in which the power consumption is not the most important design criteria, Field Programmable Gate Arrays (FPGAs) are an interesting option. FPGAs can be re-programmed in the field which makes it possible to update the cryptographic algorithm after deployment. This might be necessary if new crypt-analytic results lead to changes in the algorithm as we have seen in the past [8].

This method can be used in most of systems where small power and energy implementations are present in computing devices with power constraints and high speed implementation. The ambiguity of using method in devices like mobile terminals, wireless sensor devices in network, application specific integrated circuits (ASIC), FPGAs, PDA's, etc., can be reduced. Even though the encryption module provides the security by uses of large number of rounds it is been using more number of iterations thereby, decreasing the

computational speed of the system. In order to maintain that we developed a security model that performs certain operation which provides security. This makes the speed of operation of the system to maintain with security and making the attacker to discourage from attacks like brute force attack.

II. PROBLEM FORMULATION

The cryptographic text encryption algorithm uses a cryptographic key to encrypt or decrypt data. Each input data block is split into two halves y and z which are then applied to a routine similar to a Feistel network for N rounds, where N is typically 32. Most Feistel networks apply the result of a mixing function to one half of the data using XOR as a reversible function. This uses integer addition during encryption and subtraction during decryption [9].

It performs logical bitwise shifting, bitwise XOR, bitwise complement, basic addition and subtraction operations in the process on encryption and decryption. Additional parentheses were added to clarify the precedence of the operators. The main variables y, z, and sum, which assist with the sub key generation, have a length of 32 bits. All additions and subtractions within are modulo 32. Logical left shifts of z by 4 bits is denoted as z << 4 and logical right shift by 5 bits is denoted as z >> 5. The bitwise XOR function is denoted as "^" in this paper.

The formulae that compute the new values for y and z can be split into a permutation function

$$f(z) = (z << 4 + z >> 5) + z$$

and a subkey generation function sum + k(sum).

The function k(sum) selects one block out of the four 32-bit blocks that comprise the key, depending on either hits 1 and 0 or bits 12 and 11 of sum. The results of the permutation function and the subkey generation function are XORed and then applied to y and z respectively, by addition in the case of encryption or subtraction in the case of decryption. For encryption, z is applied to the left side, y is applied to the right side, and all adder/subtracters are in addition mode. For decryption the opposite is applied. Sum is incremented by a constant delta during encryption and decremented during decryption.

1. Proposed Scheme

Step 1: Read the Input
Step 2: Pass these values to decimal conversion method
Step 3: Read the key
Step 4: **Encryption**

Step a: Initialize sum, n and delta
Step b:
Begin n→o
Step c: sum+=delta
Step d: y+=((z<<4)+k[0])^(z+sum)^((z>>5)+k[1])
Step e: z+=((y<<4)+k[2])^(y+sum)^((y>>5)+k[3])
End
Step f: Pass these values to decimal to octal conversion method
Step g: Multiply each digit by vector a_i
Step h: Calculate cumulative sum
Step i: Send the result along with key, and vector series

Step 5: **Decryption**

Step m: Read received value
Step n:
Begin a_i series → 0
Step o: Calculate the cumulative difference
Step p: if diff > 0 increment k and append difference to current value
Step q: if diff < 0 multiply k by 10
End
Step r: Pass these values to octal to decimal conversion method.
Step s: Initialize sum, n, delta
Step t:
Begin n→0

Step u:
z- =((y<<4)+k[2])^(y+sum)^((y>>5)+k[3])
Step v:
y- =((z<<4)+k[0])^(z+sum)^((z>>5)+k[1])
Step w: sum- =delta
End
Step 6: Pass these values to text conversion method

End of Algorithm

2. Scheme Description

The Encryption method requires that we select a key which is been splitted into four parts and used for the encryption process. The given inputs are passed to decimal conversion method. In this method if alphabets are the inputs they are taken sequentially and converted them into decimal values. These are passed to the encryption module.

The encryption module performs addition, subtraction, exclusive-or (XOR), logical sift bitwise operations to yield the encrypted values. It performs logical shift operations for top 5 and bottom 4 bits as they are probably slightly weaker than the middle bit. So, the coverage rate to even diffusion is slower. These operations are more flexible to most of the computing machine with less number of cycles for operations and high speed of operation can be performed.

The main functional block in the encryption is operation of the addition, logical bitwise shift and XORed operation in the values of y and z

$$y+=((z<<4)+k[0])^(z+sum)^((z>>5)+k[1]),$$

$$z+=((y<<4)+k[2])^(y+sum)^((y>>5)+k[3])$$

This combines the key values into the given values and performs logical bitwise shifting of right for bottom 4 bits and also by shifting of left for top 5 bits as they are the probably weaker. These values are XORed to the values to yield the resulted values this is repeated for n no of rounds for higher security like feistel routine. The values that are encrypted with the key changes their original values to large values thereby confusing the attacker or cryptanalyst though they resemble decimal numbers. The values that are encrypted are then passed to security module.

The security module performs the operations of converting the values with base 10 to the base of n and thereby multiplied by with a vector a_i. The cumulative sum of these is performed. Even though the security of the encryption module can be increased by increasing the number of rounds but the problem is that to perform these many rounds the time taken for the computation increases tremendously. So not satisfied with this, to make the encryption more secure we proposed a new way of applying these values to a vector a_i. This introduces through diffusion and confusion to shatter any attempts by brute force attacks.

These values are first converted to the base of octal value so that they resemble similar to decimal but vary in representation in number systems only thereby in the view it makes the attacker looks like decimal values after some encryption routine. It provides some more security to the present systems.

The octal values are then taken digit by digit and multiplied by the values in the vector a_i where we generate a series of values for vectors a_i. The generation of the vector values can be performed in many ways for sake of illustration we shall consider the vector a_i as

$$a_i = 1, n^2, n^3, n^4 \ldots\ldots n^m \quad 1 \le i \le m$$

here m may be assumed as the no digits present in the octal number.

Now we shall explore how values are subjected to change when represented in its octal form as

$$o_i = d_1, d_2, d_3, d_4, \ldots\ldots d_m \quad 1 \le I \le m$$

We shall compute the cumulative sum $S[o_1]$ by

$$S[o_1] = \sum_{i=1}^{m} a_i \, o_i$$

In the final encrypted version the o_1 is replaced by its $S[o_1]$. Similarly all of these values are transformed to cipher values are transmitted to the receiver as $Cm=S[o_i]$.

The recipient B has all the relevant information for reversing the values using the keys, vector a_i and to recover Octal digits from these values but it is done only when the B knows the series of the vector a_i B receives the encrypted message $Cm= S[o_i]$

Now let us discuss how the reverse operation process is performed. Consider

$$S[o_1] = \sum_{i=1}^{m} a_i o_i \quad \text{for representation of } o_1$$

The o_1 value is recovered in an iterative fashion as shown

$S[o_1] - n^m$

If this value is positive i.e., $S[o_1]-n^m>0$, then k is incremented by 1 and current value is $S[o_1]=S[o_1] - n^m$. If however the value is negative then k is multiplied by 10 and $S[o_1]$ is remained unchanged this continued until a_1 series is exhausted.

Now the values are in the form of the octal number system with the base as 8. These are converted to decimal values and passed to the decryption module. Where the y and z values are represented by

z-=((y<<4)+k[2])^(y+sum)^((y>>5)+k[3]), y-=((z<<4)+k[0])^(z+sum)^((z>>5)+k[1]).

Thereby we are resulted with the originated values and these are passed to decimal to text conversion method where we get the original plain text we intended to transmit through the channel with security by encrypting.

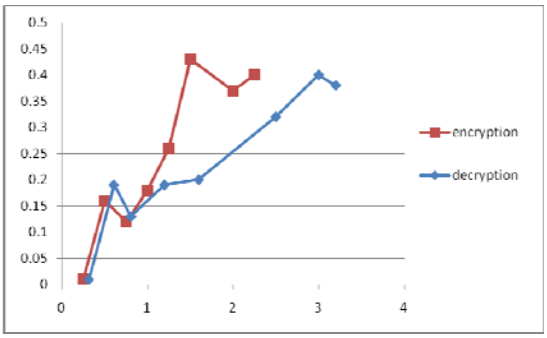
III. Results Analysis

The encryption/decryption time vs. file size is presented below.

Source File Size (KB)	Encryption Time (sec.)	Encrypted File Size(KB)	Decryption Time(sec.)
0.25	0.01	0.32	0.01
0.5	0.16	0.61	0.19
0.75	0.12	0.81	0.13
1.0	0.18	1.2	0.19
1.25	0.26	1.6	0.20
1.50	0.43	2.5	0.32
2.0	0.37	3.0	0.40
2.25	0.4	3.2	0.38

In above table alternation of size file after encryption and decryption can be easily observed.

The graph is plotted with File size vs Encryption and decryption time.



Analysing the results presented above, the following points are obtained by the proposed technique.
1. Encryption time varies non linearly with respect to source file size.
2. Decryption time varies almost linearly with respect to encrypted block size.

IV. Conclusion

We presented a simple algorithm which can be translated into a number of different languages and assembly languages very easily. It is shorter enough to be programmed from memory. It is hoped it is safe because of the number of cycles in the encoding length of key, vector series selection and number system selection. It uses a sequence of word operations rather than wasting the power of a computing system by doing byte or 4 bit operations. By applying the values with the vector a_i the attempts made by brute force attack can be discouraged for cryptanalyst. The strength of algorithm lies in the selection of the a_i vectors. As said earlier, there are infinite ways of selecting these vectors. All the series available in the world such as Tayolor's series, Maclaurin's series or fanciful generations like

$$a_i \rightarrow 1, n, (n+1)^1, (n+2)^2, (n+3)^3 \ldots (n+p)^p$$

can be used. The limit to this selection is left to the imagination of the researcher. Even assuming all the values are know to the cryptanalyst he cold not be able to figure out the a_i vector.

V. References

[1] J. K. Mandal, et al, "A 256-bit Recursive Pair Parity Encoder for Encryption", Advances in Modeling, *Computer Science & Statistics (AMSE)*, vol. 9, No. 1, pp. 1-14, France, 2004.

[2] J. K. Mandal and M. Paul, "A Permutative Cipher Technique(PCT) to Enhance the Security of Network Based Transmission", in *Proceedingsof 2nd National Conference on Computing for Nation Development*,Bharati Vidyapeeth's Institute of Computer Applications andManagement, New Delhi, pp. 197-202,08th -09th February 2008

[3] S. Dutta and J. K. Mandal, "A Universal Bit-Level Encryption Technique", *Proceedings of the 7th State Science and TechnologyCongress*, Jadavpur University, West Bengal, India, February 28-March 1, 2000.

[4] J. K. Mandal and S. Som, "A Session Key Based Secure-Bit Encryption Technique(SBET) to Enhance the Security of Network Based Transmission", *Proceedings of 2nd National Conference on Computing for Natin Development*, Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi, pp. 197-202, 08th -09th February 2008.

[5] J. K. Mandal and P. K. Jha, "Cascaded Recursive Key Rotation and Key Arithmetic of a Session Key (CRKRKA)", *International Journal of Intelligent Information Processing, India, vol. 2, No. 1, pp. 41-52, January-June 2008.*

[6] Jens-peter kaps, "Chai-Tea, Cryptographic Hardware Implementation of XTEA", in INDOCRYPT 2008 9th International Conference on Cryptology in India 09, LNCS 5365, pp 363-375,2008.

[7] National Institute of Standards, Data Encryption Standard, Federal Information Processing Standards Publication 46, January 1977.

[8] David J. Wheeler "Tiny Encryption Algorithm"

[9] Wheeler.D,"TinyEncryptionAlgorithm",Technical report, Cambridge University, England, November 1994.