International Journal of Computer Theory and Engineering, Vol. 2, No. 4, August, 2010
1793-8201

A Novel Approach for Enciphering Data of Smaller Bytes

R.Satheesh Kumar, E.Pradeep, K.Naveen and R.Gunasekaran

***Abstract*—Security being the major concern during data transfers, we have various challenges to face. The requirements of the security model vary depending on the type of data to be encrypted. Considering small size of data the requirements change from the existing security encryption techniques. One of the major issues to be considered is the computational speed of the encryption model. Though there are many challenging encryption algorithms, which provide high level of security, they increase the level of execution time for encryption. Cost effectiveness of the encryption algorithm is also another main concern for determining the efficiency of the algorithm. The computational time of the algorithms have a major impact on the determining the cost effectiveness. In this paper, we propose an enhanced cost effective symmetric key algorithm for small amount of data.**

***Index Terms*—Cryptography; Decryption; Encryption; Secret key; Security; Shared key; Small data; Symmetric key**

I. Introduction

Security is one of the major issues in Information storage and transfer. Any small hole in the encryption mechanism will lead to the benefit of the intruders and the hackers. Several algorithms have been evolving for the encryption of data since the time internet has started growing [12].

The symmetric key algorithm is a simple algorithm for encrypting and decrypting data. Even advanced algorithms like asymmetric key algorithm, DES, AES [14] gives more security than the trivial symmetric key encryption algorithm. When the amount of data is very small, then even some efficient algorithms prove to be obsolete. They take more amount of time than usual to compute this small amount of data. Hence a special algorithm, which is a modification to the trivial symmetric key algorithm, is to be used.

The bit size of the secret key influences the computational speed and the level of security provided [16]. Though higher bit size of the key increases the complexity, it shouldn't be too high to increase the computational time of the encryption model. Hence a 4-bit key is used as the secret key for encryption. This 4-bit key varies for encrypting each character, thereby increasing the challenge of the algorithm for the intruders to find the key for small amount of data.

Sharing of key is another issue to be dealt with in symmetric key encryption algorithms. There are several key management algorithms available to support this issue. It sometimes becomes an additional overhead by involving new protocols in our algorithm. Hence the key management strategy has also been handled more efficiently in this paper.

R.Satheesh Kumar, E.Pradeep, K.Naveen and R.Gunasekaran are with the Department of Information Technology, Anna University, Chennai, India(email: satheesh.ravindranath@gmail.com, pradeepfree4u@gmail.com, naveen19892000@gmail.com, gunamit@annauniv.edu).

The remaining part of this paper includes related work, proposed algorithm for symmetric key encryption and decryption, performance analysis comparison with other existing algorithms, conclusion and future work.

II. Related Work

Between the two types of cryptography techniques, the symmetric key encryption is the fast and most commonly used, compared to the asymmetric key encryption. In case of symmetric key encryption only one key is used on both sides of encryption and decryption. Few commonly used symmetric key algorithms are DES, RC2, RC4 etc. [1]

In case of Data Encryption Standard (DES) encryption technique 64 bit blocks of data are considered and an encryption key of 56 bit is used for encrypting the plain text which results in 64 bit cipher text. The key used for the encryption is shared between the encryption and decryption sides. The key management for this encryption technique doesn't follow a fixed key management scheme. Encryption techniques such as double DES, triple DES are an extension of DES technique by repeated encryption and decryption with different keys.

Advanced Encryption Standard (AES) is an advanced version of DES technique. Here the key used for encryption is usually 128,192,256 bits in size. This key size is directly proportional to the complexity of the encryption technique. 128 bits of plain data are considered for encryption and results in a 128 bit cipher text. The method used for sharing the key between the sender and the receiver side doesn't follow a standard key management technique.

According to the new encryption algorithms mentioned in [1], the ASCII value of the letters are considered and it is divided by the secret key and the cipher text is obtained from the quotient and remainder of the division. The 4-bit secret key needs to be greater than 1000.Even the public key cryptography [3] doesn't work out well for small amount of data. The authentication system has been well explained in [2]. In [1], some special cases have been missed out which have been considered in this paper.

Strong authentication scheme can also be maintained with a small size key [10]. In our paper, we use a short key of fixed size for authentication purposes. The performance evaluations of various symmetric key algorithms have been clearly stated in [11]. Further the requirements for information security are stated in [16].

III. Proposed Cryptography Technique

Our proposed encryption and decryption technique provides a cost effective method for encrypting/decrypting small data. It provides high level of security by having dynamic secret key. The main feature of our proposed algorithm is the use of dynamic 4-bit secret key and quick encryption and decryption technique.

A. *Dynamic 4-BIT secret key*

In order to provide quick and simple encryption/decryption, the bits size of the secret key has to be chosen effectively. When a large bit secret key is considered, though it adds more complexity to the algorithm and makes it more secure, it creates overhead to the encrypting/decrypting system. For encrypting small amount of data, there should not be any overhead to the encrypting system as well as there should not be any compromise on the security level. Thus an optimized size of 4 bits is chosen.

According to the encryption algorithm in [1], the 4-bit key is considered static throughout the encryption/decryption process. But in our encrypting/decrypting technique, the 4-bit secret key keeps on varying for each character of the plain text, thus increasing the level of security. This 4-bit secret key can be any binary value from 1000 to 1111.

B. *Dynamic 4-BIT shared key*

In order to enhance the security of the algorithm and provide safe transmission of the secret key, a 4-bit shared key is used. This 4-bit shared key is used to encrypt the 4-bit secret key using the same encryption algorithm. This 4-bit shared key can be any binary value from 1000 to 1111. Any one is chosen as the initial value for the shared key. The dynamic nature of the key involves incrementing the key by 1 for each bit of the secret key. This key has to be shared between the two parties very securely via an Authentication center. Each bit of the secret key is now encrypted to 9 bits. The security of our algorithm for small amount of data is increased by the introduction of this shared key over the secret key.

In [1], though authors have specified that the key size can be 4bit greater than 1000, it doesn't work out well for certain special cases of the key chosen and the plain text. In [1], the authors have allotted only 3 bits for storing the reminder. But for certain special cases, the remainder bits become 4 bits. Hence our algorithm overcomes that drawback by representing the final cipher text in 9-bit, allocating 4 bits for the reminder.

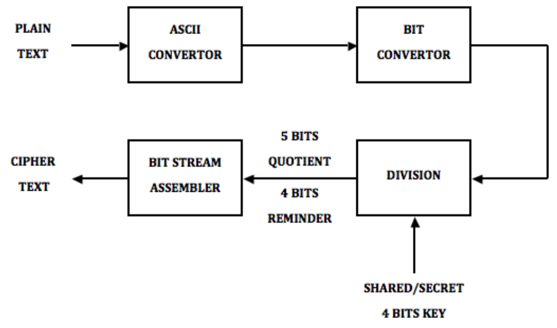
C. *Key Management Strategy*

The existing encryption algorithms such as DES, double DES, triple DES, and AES follow a separate key management and authentication scheme. This separate key management scheme is an extra overhead. In case of small amount to be encrypted, reducing this type of overhead will improve the efficiency. Thus in case of small amount of data to be encrypted, the same encryption algorithm can be used for encrypting the key too.

We have used the same encryption algorithm for sharing the secret key between the two parties. Key management being a significant issue with symmetric key cryptographic algorithms, we have overcome this issue by using a two level encryption technique for key sharing. The first level deals with the encryption of the plain text with the help of the secret key. The second level corresponds to the encryption of the secret key using the shared.

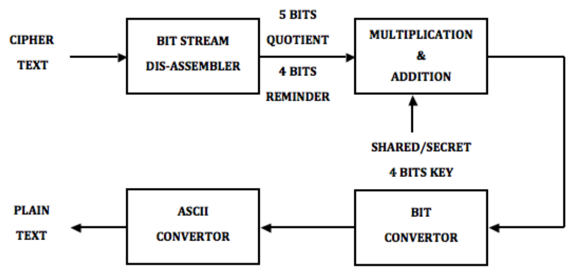
IV. ARCHITECTURE MODEL

Figure 1 and Figure 2 shows the block diagram for the encryption model and the decryption model respectively.



Encryption Block Model

The common blocks available in both the models are complementary in nature. The "Shared/Secret key" indicates that either the shared key or the secret key is used in the block with respect to the current context. When the secret key is encrypted, then the shared key is used or when the plain text is encrypted, then the secret key is used.



Decryption Block Model

ASCII Converter

This block converts the plain text to a set of ASCII values. Each character in the string is converted to its 8-bit ASCII value at the encryption side. The same block is used at the decryption block, which converts the 8-bit ASCII value to the characters of the plain text.

Bit Converter

This block converts the ASCII value to stream of bits and vice-versa at the encryption and the decryption side respectively. Each ASCII value is converted to its corresponding 8-bit stream. At the decryption side, from the stream of bits, each 8-bit frame is taken and converted to its ASCII value.

Division

The actual encryption technique is done here. The 8-bit stream is reversed and it is divided by the secret key or the shared key. The output of this block is the 5-bit quotient and the 4-bit reminder.

Multiplier

The input to this block is the 9 bit stream containing the quotient and the reminder found at the sender side. The output from this block is the 8-bit stream corresponding to the plain text. The secret key or the shared key has to be given as an additional input.

Bit Stream Assembler

This block places the quotient and reminder obtained from the previous block in its right place.

Bit Stream Dis-assembler

This block identifies the position of the quotient and the reminder from the bit stream and gives the result as an input to the *Multiplier* block.

V. Encryption Technique

The following is the algorithm for encrypting a small amount of data

Step 1: Convert each plain character into its corresponding ASCII equivalent.
Step 2: Convert the ASCII values to its binary representation in 8 bits.
Step 3: Reverse the bits.
Step 4: Divide these bits with the secret key.
Step 5: Find the quotient and the reminder.
Step 6: Represent the reminder in the first four bits and quotient in the last five bits of the 9-bit cipher text.
Step 7: Apply the same algorithm and encrypt the secret key with the help of the shared key.
Step 8: The encrypted secret key is sent before the cipher text of the actual plain text.

A. *Case Study*

Let us consider the sample text "Hello". The encryption process is done for each character individually in a sequence. Let's consider the first character "H".

Step 1: ASCII value of "H" is 72.

Step 2: The ASCII value 72 is converted into its binary value 01001000.

0	1	0	0	1	0	0	0

Step 3: The 8-bit value obtained in step 2 is reversed resulting in 00010010.

0	0	0	1	0	0	1	0

Step 4: Divide the 8-bit value got in step 3 with the secret key 1001.

Step 5: Quotient of the division is 00010 and the remainder is 0000.

Step 6: According to the algorithms the 9-bit cipher text is formed from the quotient (5-bit) and remainder (4-bit) of the division performed.

0	0	0	0	0	0	0	1	0

The above steps are repeated for the remaining characters of the plain text. The key secret key for the next character "e" will be 1010. The secret key keeps on varying (1001, 1010, 1011 etc.) for the individual characters of the plain text.

The cipher text for the plain text "Hello" is 000000010011010000101000100011000100110010010. It also transmits the encrypted secret key 100001011110000000011000000000010101001. The secret key is encrypted with the shared key 1100.

VI. Decryption Technique

Extract the secret key with the help of the shared key by using the following procedure.

Step 1: Extract the quotient and reminder from the 9-bit cipher text.
Step 2: Multiply the shared key with the quotient and add it with the reminder.
Step 3: Reverse and find out the actual character with the help of its ASCII equivalent.
Step 4: The above procedure is followed until the secret is found out.
Step 5: With the help of this secret key, the actual plain text is found out by following the same procedure.
The keys used in every point of time will be dynamic in its nature (i.e.) the value of the key will be shuttling between 1000 and 1111.

A. *Case Study*

The encrypted secret key 100001011110000000011000000000010101001 is first decrypted using the shared key (1100). Using the secret key 1001 the remaining cipher text is decrypted to get back the plain text. The decryption method followed for decrypting the secret key is similar to the decryption of the remaining cipher text.

Let's consider the decryption of the cipher text corresponding to the plain text "H".

Step 1: Extract the quotient (00010) and reminder (0000) from the 9-bit cipher text.

0	0	0	0	0	0	0	1	0

Step 2: Secret key 1001 is multiplied with the quotient 00010 and added with the reminder 0000.

0	0	0	1	0	0	1	0

Step 3: Reverse of the above value being 01001000 is obtained and the corresponding character based on ASCII equivalent "H" is found.

0	1	0	0	1	0	0	0

The above steps are repeated for the remaining cipher text with different secret keys (1001, 1010, 1011, etc) to get back the plain text "Hello". The decryption of the encrypted secret key is also done using the above mention decryption technique.

B. *Special Case*

Step 1: ASCII value of "U" is 85.

Step 2: The ASCII value 85 is converted into its binary value 01010101.

0	1	0	1	0	1	0	1

Step 3: The 8-bit value obtained in step 2 is reversed resulting in 00010010.

1	0	1	0	1	0	1	0

Step 4: Divide the 8-bit value got in step 3 with the secret key 1010.

Step 5: Quotient of the division is 10010 and the remainder is 1000.

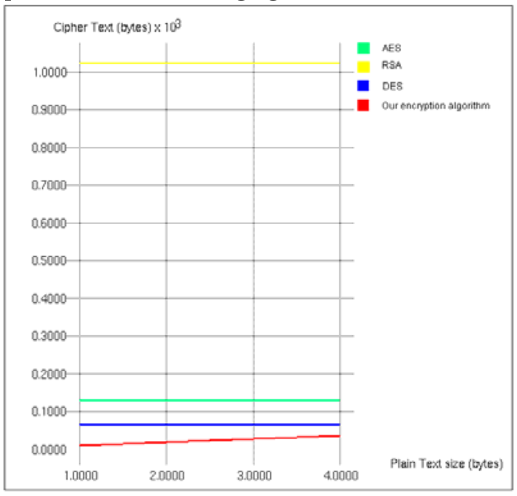
Step 6: According to the algorithms the 9-bit cipher text is formed from the quotient (5-bit) and remainder (4-bit) of the division performed.

1	0	0	0	1	0	0	1	0

It has to be noted the number of bits for the reminder that has been generated in this special case is 4. Hence the old algorithm would not work fruitfully.

VII. Performance Analysis

Performance analysis has been made with all other existing encryption algorithms. The graphs have been developed with the knowledge gained from [5] [6] [7] [15]



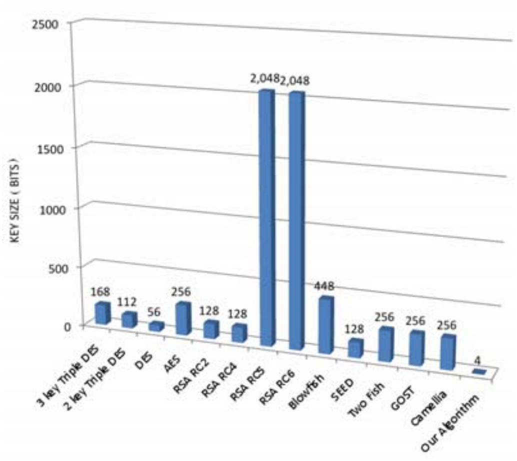
Plain Text Vs Cipher Text

Figure (3) shows the variation of the number of bits in the cipher text with respect to the plain text's bits. This graph shows that our encryption algorithm results in lesser number of cipher text bits for small amount of data comparing with other existing encryption algorithms namely AES, DES, and RSA.

Table 1 shows the comparison of key sizes among different existing encryptions algorithms [6] [7] [8] [9] [13]. Though the security is provided in various algorithms by having large key size, equivalent security is provided in our encryption algorithm by having a small key size with reversing techniques.

TABLE I. KEY SIZE COMPARISON OF VARIOUS ENCRYPTION ALGORITHMS

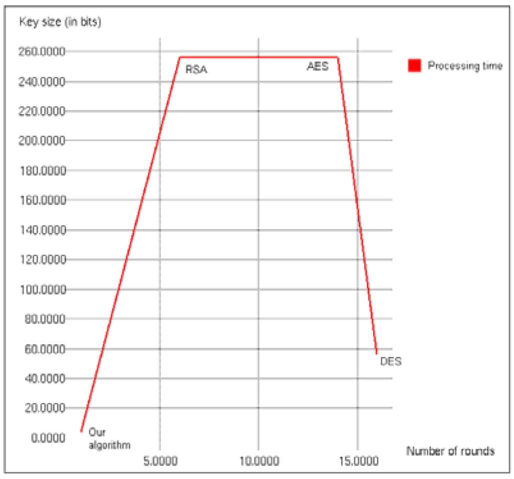
ALGORITHM	TYPE	KEY SIZE
3 key Triple DES	Block cipher	168
2 key Triple DES	Block cipher	112
DES	Block cipher	40/56
AES	Block cipher	128/192/256
RSA RC2	Block cipher	128 or any size
RSA RC4	Stream cipher	40/56/128/25
RSA RC5	Block cipher	0 to 2,048 bits
RSA RC6	Block cipher	0 to 2,048 bits
Blowfish	Block cipher	32 up to 448 bits
SEED	Block cipher	128
Two Fish	Block cipher	128 to 256 bits
GOST	Symmetric cipher	256 bits
Camellia	Block cipher	128 bits, 192 bits, and 256 bits



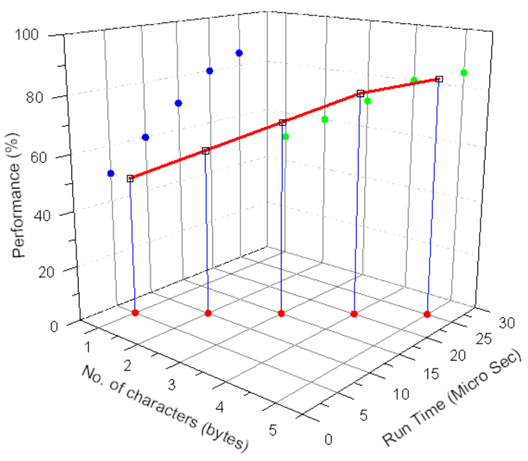
Histogram for different key sizes

Figure 4 shows the visual representation of different encryption algorithms' key sizes. It has to be noted that our algorithm occupies the lower position in the key size. Though the key size is less, it provides an equivalent security to the information.

Figure 5 shows the plot of key size and number of rounds for different encryption algorithms namely AES, DES, RSA [5] [6]. The graph depicts the processing time of each algorithm to compute the cipher text for small amount of data. An increase in the processing time [4] for small amount of data shows an unnecessary degradation in the system, which our encryption algorithm over comes.



Number of Rounds Vs Key size



Performance of our Cipher model

Figure 6 shows the run-time of our algorithm with the increase in the bytes of the plain text. It also shows the efficiency of our novel approach in enciphering the plain text. Since our algorithm is meant only for small amount of data, plain text's size greater than 4 bytes may lead to degradation in the efficiency. So, we always recommend using our algorithm for only small amount of data. Since the key exchange is made only once during the start of the session, the overhead of the amount bits used for the encryption of the secret key is reduced in further transformation of the data.

VIII. Conclusion

Our encryption algorithm for small amount of data provides high level of security with short duration and smooth encryption of data. It provides equal challenge as other encryption algorithms with less number of bits used for the secret key. Hence small amount of data can be encrypted in a faster and secure manner. Division and reversing operations increase the challenge of the algorithm for small amount of data by introducing toughness for the intruders to decipher the plain text. Our enciphering approach also provides high level of authentication between the end users with only a small key size, which is fixed. The key management for the sharing of the secret key between the two parties is also efficiently handled using the same algorithm. Enormous overhead due to the large key size has been effectively ruled out in this paper. Various projections on the performance of our algorithm have also been discussed in this paper.

IX. Future Scope

Repeating our encryption algorithm for small amount of data in a cyclic manner can increase the complexity of our system. Various permutation techniques can be used in place of reversing the bits to increase the complexity of the algorithm. Our system can be extended to be used along with different key management and sharing strategies. Optimizing the bit size of the secret key can also encrypt the medium size data. This system can be further optimized to combine with other existing encryption algorithms for exchanging their keys. The position of the secret key transmission can also be changed dynamically for each character.

One small modification can be done to our encryption algorithm to increase the challenge for the hackers. The resultant cipher text which is obtained can be grouped in the form of 8 bits. Sufficient number of zeros can be padded at the end, if the number of bits is not a multiple of 8. Each group of 8 bits can be again converted to their respective ASCII values before transmitting to the receiver. Because of this enhancement, there wouldn't be any significant change in the performance.

References

[1] Sarker, M.Z.H.; Parvez, M.S., "A Cost Effective Symmetric Key Cryptographic Algorithm for Small Amount of Data", 9th International Multitopic Conference, IEEE INMIC2005, pages: 1-6.

[2] D. Jablon, "Strong Password Only Authenticated Key Exchange" Computer Communication Review, ACM SIGCOMM, Vol 26 No 5, pp 5-26, 1997

[3] Cilardo, A.; Mazzeo, A.; Mazzocca, N.; Romano, L., "A novel unified architecture for public-key cryptography", Proceedings of the conference on Design, Automation and Test in Europe, 2005, Volume 3, Pages: 52 - 57.

[4] M. McLoone, "Hardware performance analysis of the SHACAL-2 encryption algorithm", IEEE Proceedings on Circuit Devices and Systems, 2005, VOL 152; pages 478-484.

[5] Different bit sizes of various encrypting algorithms http://download.oracle.com/docs/cd/B12037_01/network. 101/b10772/asoappa.htm.

[6] Different encryption/decryption algorithms http://www.networksorcery.com/enp/data/encryption. htm.

[7] Camellia web page, http://www.faqs.org/rfcs/rfc4312.html.

[8] RC algorithms http://www.smartcomputing.com/ editorial/ dictionary/detail.asp? guid=&searchtype=& DicID=18720&RefType=Encyclopedia.

[9] Rounds of encryption for RSA. http://www.rsa.com/ rsalabs/node.asp?id=2254.

[10] David P. Jablon, "Strong Password-Only Authenticated Key Exchange", Computer Communication Review (ACM SIGCOMM) 26 (5): 5–26, September 25, 1996.

[11] Diaa Salama Abdul. Elminaam1, Hatem Mohamed Abdul Kader2 and Mohie Mohamed Hadhoud3, "Performance Evaluation of Symmetric Encryption Algorithms", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[12] W.Stallings, "Cryptography and Network Security 4th Ed," Prentice Hall, 2005, PP. 58-309.

[13] Bruce Schneier. The Blowfish Encryption Algorithm, October 25, 2008, http://www.schneier.com/blowfish.html.

[14] Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard."D r. Dobb's Journal, March 2001, PP. 137-139.

[15] "A Performance Comparison of Data Encryption Algorithms," IEEE [Information and Communication Technologies, 2005. ICICT 2005. First International Conference, 2006-02-27, PP. 84- 89.

[16] Hardjono, "Security In Wireless LANS And MANS," Artech House Publishers 2005.

R.Gunasekaran received the BE degree in computer science and engineering from the University of Madras, Chennai, India, and the ME degree in computer science and engineering from the Bharathiyar University, Coimbatore, India.
Since 2003 he has been with Anna University Chennai, India where he is currently a senior lecturer in the Department of Information Technology. He is in the verge of completing his Ph.D degree in Computer Science and Engineering in Anna University Chennai, India. His research interests include Mobile Ad Hoc Networks, Mobile Communications and WiMAX.
Mr. Gunasekaran is associated with Journal of Network and Computer Applications, Elsevier Publications, Journal of the Network and Systems Management, Springer Publications as a reviewer. He is a member of IEEE, ISTE and CSI.

R.Satheesh Kumar is pursuing his final year Bachelor of Technology in Information technology at Anna University, Chennai, India.
He was an intern at Microsoft, IDC, Hyderabad, India as a PROGRAM MANAGER during May – July 2009. His research interests include networks, information security, cryptography and grid computing.
Mr. Satheesh is a Student member of IEEE.

Pradeep.E is pursuing his final year Bachelor of Technology in Information technology at Anna University, Chennai, India.
He has completed CCNA training at Cisco Labs. He is an alumnus of Cisco Networking Academy. His research interests include cryptography, network security, computer networks and grid computing.
Mr. Pradeep is a Student member of IEEE and a member of IEEE Communications Society.

Naveen.K is pursuing his final year Bachelor of Technology in Information technology at Anna University, Chennai, India.
He has completed certified network courses conducted by Microsoft and Cisco. He holds MCP, MCSE, MCTS and CCNA. He is also pursuing CCNP and CWNA. His reach interests include networks, information security and grid computing.
Mr. Naveen is a Student member of ACM.