Cognitive Symmetric Key Cryptographic Algorithm

Y.R.A. Kannan, S. Aravind Prasad, and P. Varalakshmi

Department of Information Technology,
Madras Institute of Technology, Chrompet,
Chennai, 600044
{yra.kannan,raja.avi,varanip}@gmail.com

Abstract. Today, Cryptographic schemes play a major role in storage, retrieval and transfer of data and code in a secured manner. The major factors which determine the efficiency of a cryptographic system are computational speed, level of security provided, cost effectiveness and key size of the algorithm. Choosing the optimal bit size for keys in encryption and decryption algorithm is necessary for the efficient computation of the algorithm and bit size of the key is directly proportional to the complexity of the algorithm and in turn cost of the encryption and decryption algorithm. In this paper, we propose a novel cost-effective small key size symmetric key algorithm which is suitable to all sizes of data and all kinds of data such as audio and video because of its low key size and high computational speed.

Keywords: Key size, Computational speed, Encryption algorithm, Decryption algorithm, Symmetric key algorithm.

1 Introduction

In today's world, data security plays a prominent role in all areas and with latest advancements in the fields of data storage and processing speeds, the focus towards security is even more increased. Basically cryptography is the way of transforming the ordinary data or information into meaningless information in order to maintain the secrecy of its contents. Usually the type of data being dealt with, determines the type of security model to be used for it. Conversion of plain text to cipher text is referred to as Encryption and vice versa is termed as Decryption. Usually the major security goals to be satisfied are *Availability, Confidentiality* and *Integrity.*

The most prominent factor in any cryptographic algorithm is the key size which in turn influences computational speed and level of security provided. As the key size is directly proportional to the complexity of the algorithm, we make use of 4-bit size key which can be used to all sizes of data and all kinds of data such as audio and video because of its low key size and high computational speed.

This paper is organized as follows. The proposed algorithm is given in section 2 and the performance analysis of our algorithm compared with the existing algorithms is done at section 3. Related work is described in section 4. Finally conclusions are drawn in section 5.

2 Proposed Mechanism

In our proposed algorithm, dynamic 4-bit key size and different 4-bit keys are used for different characters in the plain text. Cognitive Symmetric key Cryptographic algorithm (CSCA) can encrypt and decrypt the large data since there will be no increase in key size. Even though any one key is identified in CSCA, other keys cannot be identified because of using different 4-bit keys for different characters rather than choosing only one 4-bit key and adding one to that initial key. Security is also enhanced in CSCA by encrypting secret key by using 4-bit shared key which should be shared securely via Authentication center between the two parties. In order to extract the secret plain text from the cipher text, first the secret key has to be decrypted by using shared key and then the character should be decrypted by using that secret key. In CSCA after decrypting all the characters, plain text will be found.

2.1 Dynamic 4-Bit Secret Key

Choosing the optimal bit size for keys in encryption and decryption algorithm is necessary for the efficient computation of the algorithm. If the bit size of the key is very large, the complexity of the algorithm is increased. This also increases the cost of the encryption and decryption algorithm. In many encryption and decryption algorithms the key size is very large. In [1], the dynamic 4-bit key is used for encryption and decryption for small amount of data. In our proposed algorithm, 4-bit key size is used for encryption as well as decryption. 4-bit Randomizer is used in our algorithm which returns the random value from 10 to 15 i.e. from 1000 to 1111 for encrypting the characters and same key has to be used for decryption since it is a symmetric algorithm. So only 4-bit key size is used throughout the algorithm but different keys are used for characters in the plain text by using 4-bit Randomizer. Hence, our proposed algorithm is used to encrypt and decrypt the large amount of data with small key size in efficient and secured manner.

2.2 4-Bit Shared Key

In order to increase the security 4-bit shared key can be introduced. This key has to be shared between sender and receiver of the data in a secured manner via an Authentication center. The shared key is also 4-bit key. This key is any value between 1000 to 1111 which is chosen by 4-bit Randomizer in Authentication center. This shared key is used to encrypt and decrypt the secret key by using the proposed encryption and decryption algorithm for encrypting and decrypting the keys. Introduction of the 4-bit shared key increases the security and efficiency of our algorithm.

2.3 Encryption Technique

In this technique first the key generated from the 4-bit Randomizer is used to encrypt the first character in the Plain text. Since the 4-bit shared key is constant throughout the encryption and decryption technique consider it as n_1. Then the shared key is used to encrypt the secret key.

CSCA is used to encrypt the data with the 4-bit secret key which is given in Fig 1.

Step1 : 4-bit Randomizer generates a key value that varies between 10 to 15 and considers it as n.

Step2 : Convert the generated value into binary value in 4 bits which is the secret key for encrypting a character.

Step3 : Compute the ASCII value for the character in the plain text and convert ASCII value into binary representation in 8 bits.

Step4 : Perform left circular shift operation for n (the value generated in Step1) times on the binary representation of ASCII value.

Step5 : Divide the result from step 4 by the secret key.

Step6 : Represent the remainder in first four bits and quotient in the last five bits which gives 9-Bit cipher text for the character.

Step7 : Represent the 4-bit key in 8 bits.

Step8 : Perform left circular shift operation for n_1 times on these bits.

Step9 : Divide these bits with the 4-bit shared key.

Step10: Represent the remainder in first four bits and quotient in the last five bits which gives 9-Bit cipher text for the secret key.

Step11: Form 18-Bit cipher text by using 9-Bit cipher text for secret key and 9-Bit cipher text for the character.

Repeat the above steps for encrypting each character in the plain text. If P is the plain text, Ch_1 is the first character, Ch_2 is the second character then C_1 is 9-bit cipher text for Ch_1 encrypted with key K_1 and C_2 is 9-bit cipher text for Ch_2 encrypted with key K_2. EK_1 is 9-bit cipher text for K_1 encrypted with Shared key and EK_2 is 9-bit cipher text for K_2 encrypted with Shared key. The pattern of the resultant Cipher text from this Encryption Technique for Plain Text P will be $EK_1C_1EK_2C_2EK_3C_3EK_4C_4\ldots$

Fig. 1. CSCA for Encryption

2.3.1 Flowchart

The flowchart for Cognitive Symmetric Key Encryption Algorithm for Encryption is given in Fig 2.

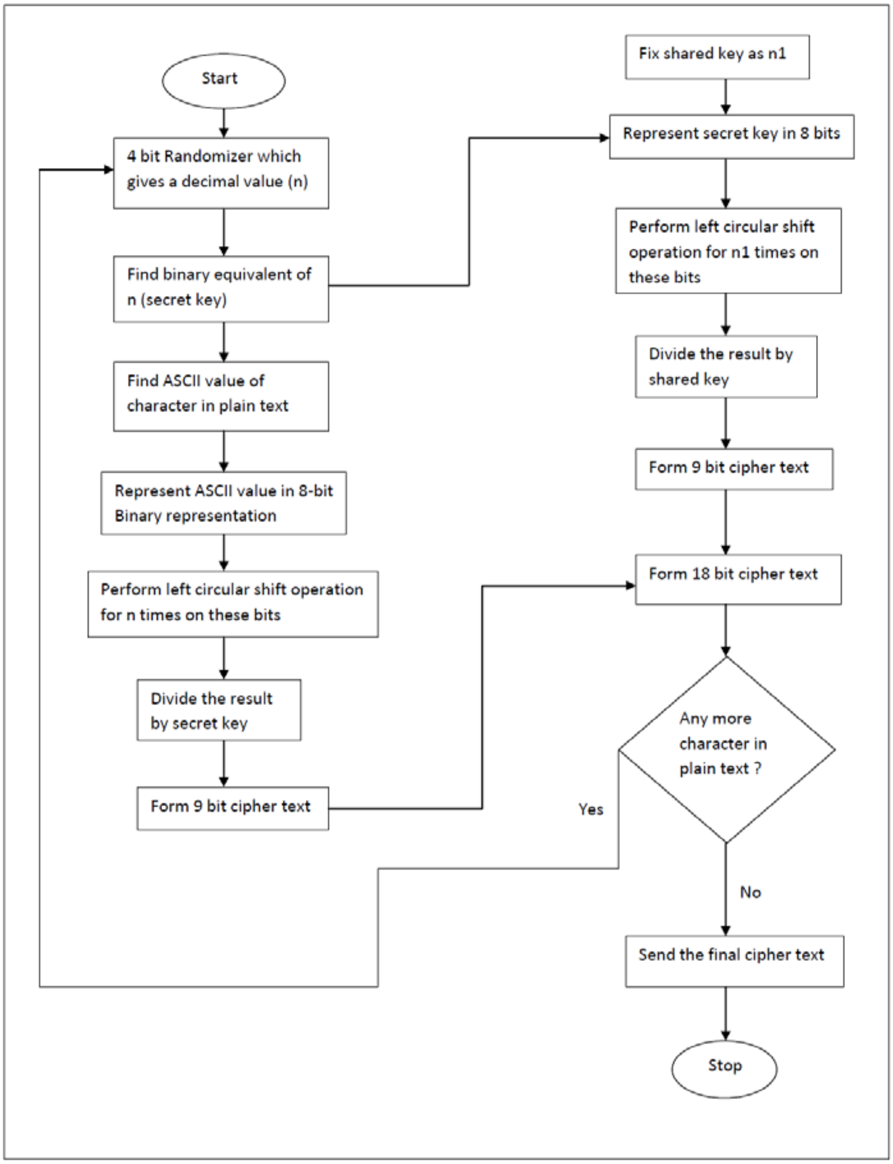
Start
4 bit Randomizer which gives a decimal value (n)
Find binary equivalent of n (secret key)
Find ASCII value of character in plain text
Represent ASCII value in 8-bit Binary representation
Perform left circular shift operation for n times on these bits
Divide the result by secret key
Form 9 bit cipher text
Fix shared key as n1
Represent secret key in 8 bits
Perform left circular shift operation for n1 times on these bits
Divide the result by shared key
Form 9 bit cipher text
Form 18 bit cipher text
Any more character in plain text ?
Yes
No
Send the final cipher text
Stop

Fig. 2. Encryption Technique in CSCA

2.3.1 Case Study

Let us consider the plain text as "HI". The encryption technique has to be done for each and every character in the plain text individually. First we will consider the character H in plain text HI. Let us consider shared key as 1010 i.e. 10 (n_1).

Step1 : Let say 4-bit Randomizer generates the value 1000.
Step2 : Decimal value of 1000 is 8 and considers it as n.
Step3 : ASCII value for H is 72 and 8-bit binary representation for 72 is 01001000.
Step4 : Perform left circular shift on 01001000 for n times i.e. 8 times which gives the result as 01001000.
Step5 : Divide 01001000 by secret key 1000. Quotient is 01001 and the remainder is 0.
Step6 : The 9-Bit cipher text is 000001001.
Step7 : 8-bit representation of secret key is 00001000.
Step8 : Perform the left circular operation on 00001000 for n_1 times i.e. 10 times which gives the result as 00100000.
Step9 : Divide 00100000 by shared key 1010. Quotient is 00011 and remainder is 0010.
Step10: The 9-Bit cipher text is 001000011.
18-Bit Cipher Text for the character H: 001000011000001001.

Let us consider the secret key for I is 1011. Then Final Cipher text for the text "HI" is 001000011000001001010000100100000110.

2.4 Decryption Technique

In this technique first the secret key has to be decrypted from the 9-bit cipher text by using the shared key then the character has to be decrypted by using the respective secret key.

Reverse of CSCA is used to decrypt the characters in the plain text from cipher text.

Step1: Extract 18 bit cipher text from the received bits and extract the quotient and remainder from the first 9-bit cipher text for finding the secret key.
Step2: Multiply the quotient with the shared key and add it with the remainder.
Step3: Perform the right circular shift operation for n_1 times on these bits and this computes the secret key.
Step4: Compute the decimal value for the secret key and consider it as n_2.
Step5: Extract the quotient and remainder from the next 9-bit cipher text for finding the character.
Step6: Multiply the quotient with the secret key and add it with the remainder.
Step7: Perform the right circular shift operation for n_2 times on these bits and compute the decimal value.
Step8: Find the secret character based on its ASCII equivalent.
The above steps have to be repeated until all the secret characters in the plain text are found.

Fig. 3. CSCA for Decryption

2.4.1 Flowchart

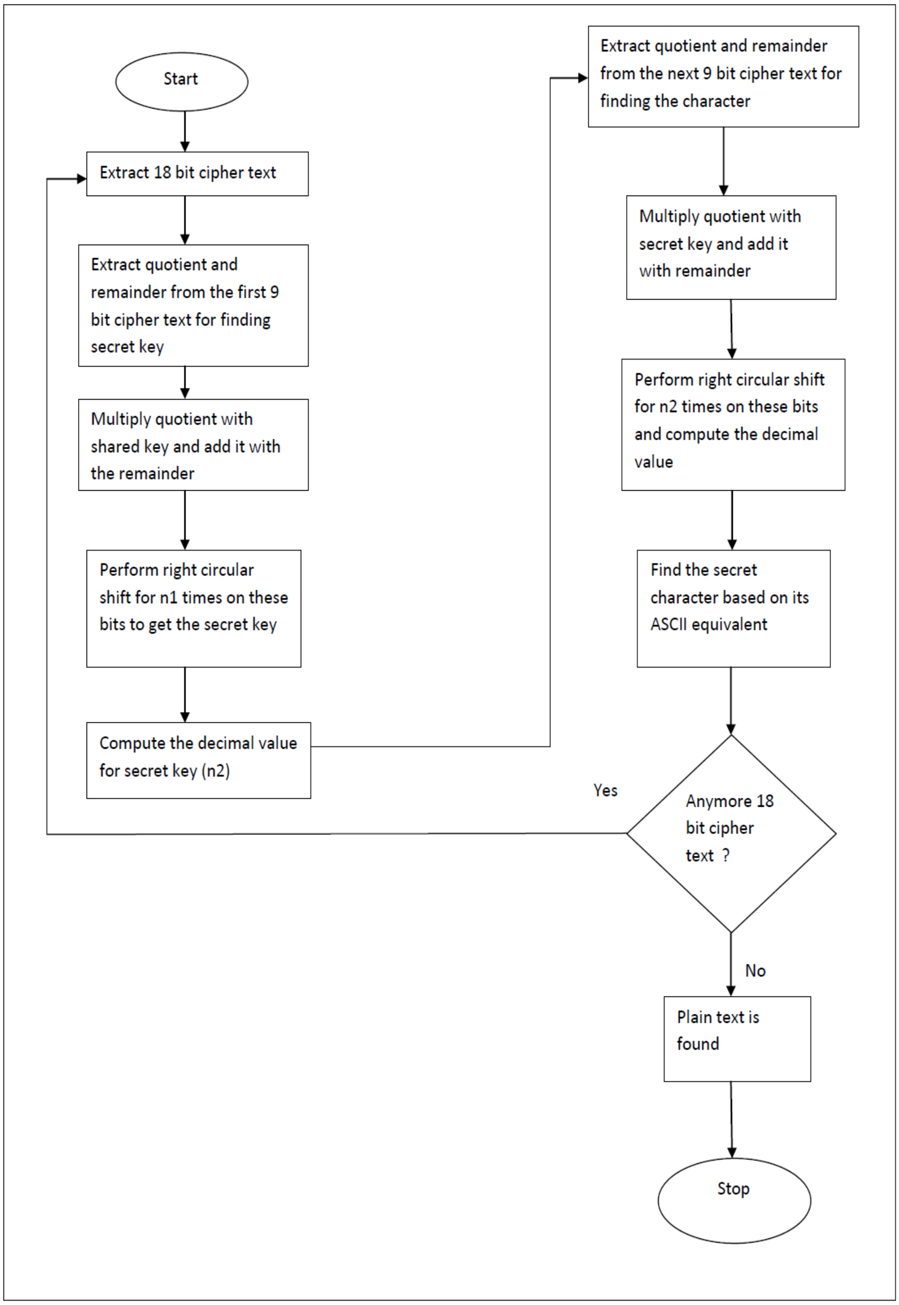
Start
Extract 18 bit cipher text
Extract quotient and remainder from the first 9 bit cipher text for finding secret key
Multiply quotient with shared key and add it with the remainder
Perform right circular shift for n1 times on these bits to get the secret key
Compute the decimal value for secret key (n2)
Extract quotient and remainder from the next 9 bit cipher text for finding the character
Multiply quotient with secret key and add it with remainder
Perform right circular shift for n2 times on these bits and compute the decimal value
Find the secret character based on its ASCII equivalent
Anymore 18 bit cipher text ?
Yes
No
Plain text is found
Stop

Fig. 4. Decryption Technique in CSCA

2.4.2 Case Study

Let us consider for the decryption of character H. The Cipher text for H is 001000011011100001.

Step1: Quotient and remainder are extracted from first 9 bits as 00011 and 0010.
Step2: Multiply 00011 with shared key 1010 and add it with remainder 0010 and this gives the result as 100000.
Step3: Perform right circular operation on 00100000 for n_1 times i.e. 10 times which gives the secret key as 1000.
Step4: Decimal value for 1000 is 8 and considers it as n_2.
Step5: Quotient and remainder are extracted from next 9 bits as 01001 and 0.
Step6: Multiply 01001 with 1000 and add it with 0 which gives the result as 01001000.
Step7: Perform right circular operation on 00001111 for n_2 times i.e. 8 times which gives the result as 01001000.
Step8: Decimal value for 01001000 is 72 and the corresponding character based on ASCII equivalent "H" is found.

Similarly the character "I" is found by using this decryption technique to get the plain text "HI".

3 Performance Evaluation

In this section, we show performance evaluation of the proposed Cognitive Symmetric Key Cryptographic Algorithm (CSCA) compared to the various existing Algorithms. For our simulations, we made use of Crypto++ tool.

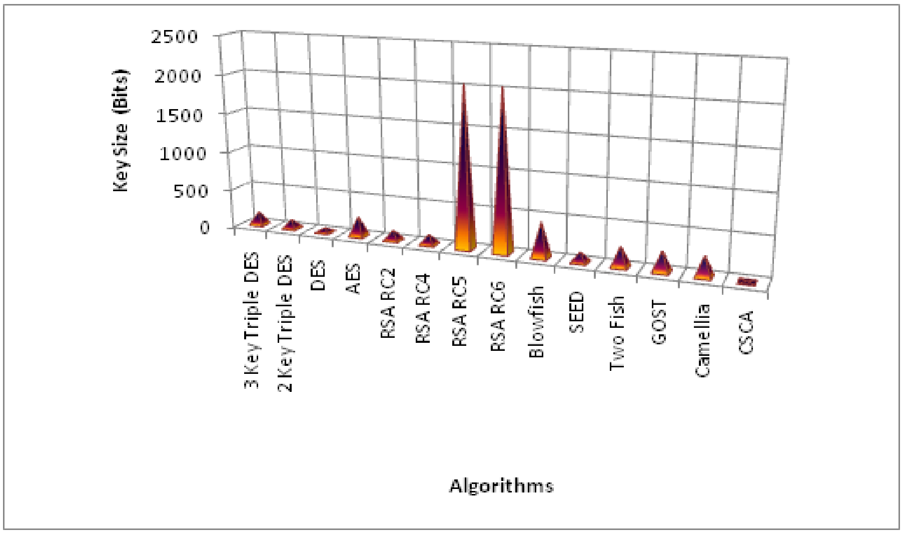

Fig. 5. Key sizes of Algorithms

Fig 5 provides the visual representation of various cryptographic algorithms with different key sizes and clearly depicts the low and effective key size of our proposed algorithm. With larger key sizes, the complexity increases and leads to decrease in cost effectiveness and computational speed of the encryption and decryption processes of the entire message.

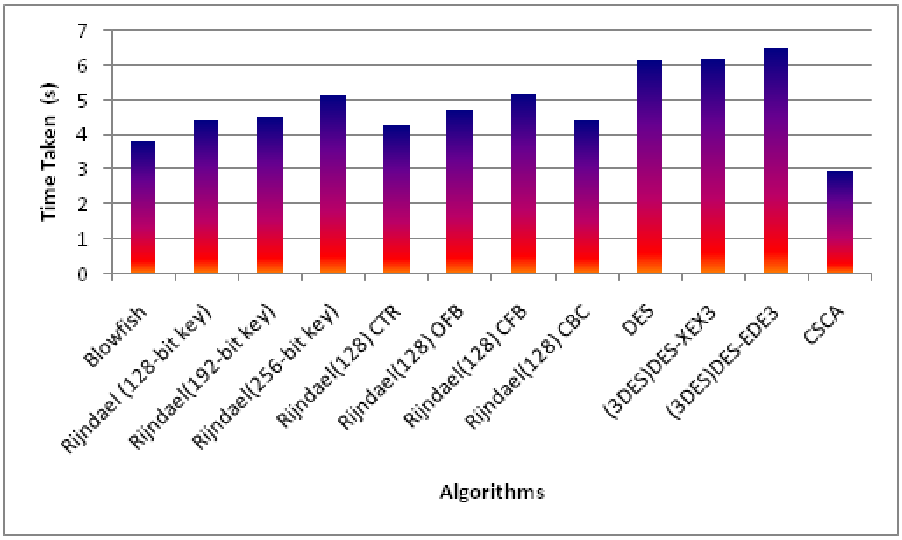

Fig. 6. Time consumption of various Algorithms

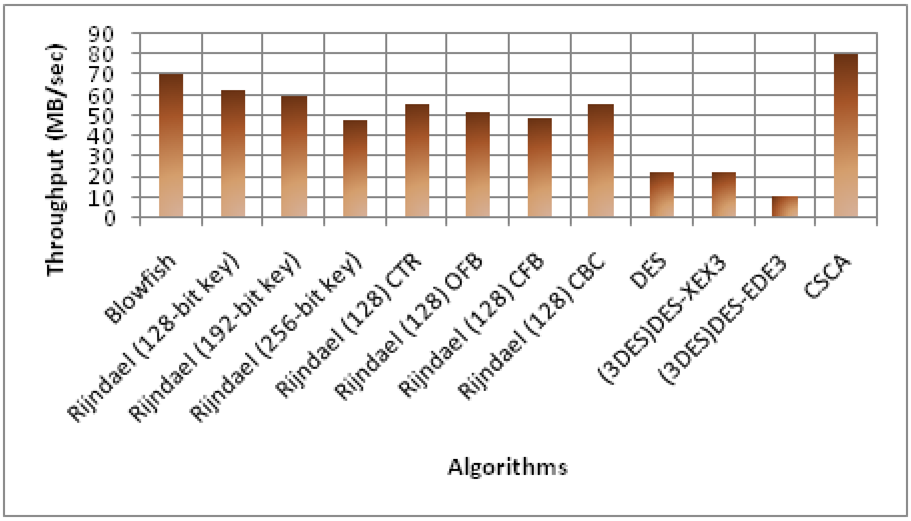

Fig. 7. Throughput for different Algorithms

Fig 6 shows the time consumed by various algorithms. It has to be noted that our algorithm occupies the lowest position among all existing algorithms. Again with the low time consumption, our proposed algorithm performs better than almost all existing algorithms. In the above figure, CTR, OFB, CFB and CBC represent Counter mode, Output Feedback Mode, Cipher Feedback Mode and Chain Block Chaining Mode respectively. Time consumption of operations performed in CSCA is low when compared to other algorithms as shown in Fig 6.

Fig 7 provides the comparison of throughputs of various existing algorithms and with the higher throughput and clearly our algorithm proves to be highly cost-effective and faster when compared to other algorithms.

Table 1. Megabytes processed for various algorithms using Crypto++ tool for Performance evaluation

Algorithm	Megabytes processed	Time consumed	Key size
Blowfish	256	3.79	32 up to 448 bits
Rijndael (128-bit key)	256	4.41	128
Rijndael (192-bit key)	256	4.52	192
Rijndael (256-bit key)	256	5.1	256
Rijndael (128) CTR	256	4.26	128
Rijndael (128) OFB	256	4.71	128
Rijndael (128) CFB	256	5.2	128
Rijndael (128) CBC	256	4.4	128
DES	128	6.123	40/56
CSCA	256	2.97	4

4 Related Work

The two techniques used for cryptography are symmetric and asymmetric cryptographic techniques. Symmetric key encryption is simple, fast and most commonly used. In this technique, same key has to be used for encryption and decryption and so this key has to be shared between the two parties in secured manner. In asymmetric cryptographic technique, two different keys are used for encryption and decryption. Some examples of symmetric key algorithms are DES, RC4, AES, Blowfish, IDEA. According to the symmetric algorithm proposed in [1], 4-bit key size is used for encryption and decryption. In [1], the initial key is chosen as any value from 1000 to 1111 for encrypting and decrypting the first character. For remaining characters in the plain text, the secret key is increased by 1 from the initial value. I.e. If the initial key is 1000 then the keys for the remaining characters will be 1001, 1010, 1011, 1101, 1110, 1111, 10000… Because of increasing the keys, the key size also increases and so error will occur if we encrypt and decrypt the larger amount of data according to the encryption and decryption algorithm proposed in [1]. Also in [1], if any one key is determined then other keys can be easily determined either by adding or subtracting 1 to key which is found.

In [1], ASCII values of the letters in plain text are divided by the secret key and cipher text is generated by using the quotient and remainder of the division. Computational speed is high because of using small key size. But using the algorithm in [1], some letters cannot be encrypted because of using 3 bits in remainder. This problem is solved by using 4 bits in the remainder according to the algorithm proposed in [2]. In these two algorithms, only small data can be encrypted and decrypted because of incrementing keys, key size will be increased and hence error will be occurred while encrypting and decrypting the large data. Also in [1] and [2], if any one key is found then other keys can be easily generated by either adding or subtracting one to the key which is found.

5 Conclusion

Hence our proposed cost-effective small key size Cognitive Symmetric key Cryptographic Algorithm (CSCA) is suitable to all sizes of data and all kinds of data such as audio and video because of its low key size and high computational speed. The bit shifting operations and the independency of keys used from the previous set of keys makes it really challenging and the performance analysis clearly shows the superiority of our algorithm compared to all the existing algorithms in terms of time consumed, throughput and key size which in turn reflects the cost effectiveness and decreased overhead.

References

1. Sarker, M.Z.H., Parvez, M.S.: A Cost Effective Symmetric Key Cryptographic Algorithm for Small Amount of Data. In: 9th International Multitopic Conference, IEEE INMIC (2005)
2. Kumar, R.S., Pradeep, E., Naveen, K., Gunasekaran, R.: A Novel Approach for Enciphering Data of Smaller Bytes. International Journal of Computer Theory and Engineering 2(4) (August 2010)
3. Performance Analysis of Data Encryption Algorithms, `http://www1.cse.wustl.edu/~jain/cse567-6/ftp/encryption_perf/index.html`
4. Elminaam, D.S.A., Kader, A.M.A., Hadhoud, M.M.: Performance Evaluation of Symmetric Encryption Algorithms. IJCSNS International Journal of Computer Science and Network Security 8(12) (December 2008)
5. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Research in Security and Privacy. IEEE Press (2003)
6. Liu, J.K., Au, M.H., Susilo, W.: Self-generated certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In: ACM Symposium on Information, Computer and Communications Security. ACM Press (2007)

7. Malan, D.J., Welsh, M., Smith, M.D.: A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In: IEEE International Conference on Sensor and Ad Hoc Communications and Networks. IEEE Press (2004)
8. Deng, J., Han, Y.S.: Multipath Key Establishment for Wireless Sensor Networks Using Just-enough Redundancy Transmission. IEEE Transactions on Dependable and Secure Computing (2008)
9. Boris, K., Markus, D.: A Provably Secure And Efficient Countermeasure Against Timing Attacks. In: IEEE Computer Security Foundations Symposium. IEEE Computer Society (2009)
10. Massey, J.L.: Guessing and Entropy. In: IEEE Symposium on Information Theory. IEEE Computer Society (1994)
11. Zhu, S., Xu, S., Setia, S., Jajodia, S.: Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach. In: IEEE International Conference on Network Protocols. IEEE Press (2003)
12. MacKinnon, S.J., Taylor, P.D., Meijer, H., Akl, S.J.: An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. IEEE Transactions on Computers (1985)
13. Kumar, K., Begum, J.N., Sumathy, V.: Efficient Region-Based Class Key Agreement Protocols for Ad Hoc Networks using Elliptic Curve Cryptography. In: IEEE International Advance Computing Conference. IEEE Press (2009)