A New Symmetric Cryptosystem using Randomized Parameters of SHA-512 and MD5 Hash Functions

T. Sivakumar
Assistant Professor (Sl.G), Department of Information Technology
PSG College of Technology, Coimbatore, Tamilnadu, India

T. Anusha
Assistant Professor (Sr.Gr), Department of Computer Science and Engineering
PSG College of Technology, Coimbatore, Tamilnadu, India

Abstract- The explosive growth of digital information places a high demand for information security. Information security deals with securing the information from unauthorized access or misusing information either intentionally or accidentally. Information may be represented in forms like text, audio, video, and image. The most widely used form of information exchange is text documents. This paper presents a novel symmetric key cryptosystem to protect the text messages communicated over the public network from unauthorized disclosure. To provide increased security with reduced computational complexity, the encryption method uses the bitwise XOR operation with a larger key space derived from the randomly generated additive constants used in Secure Hash Algorithm (SHA-512) and sine constants of Message Digest (MD-5) hash functions. The proposed method is safe against cryptanalysis attack and has good avalanche effect. The method is sensitive with respect to the encryption key and satisfies both confusion and diffusion properties significantly.

Keywords – Data Security, Symmetric Key Algorithm, Hash Functions, Cryptanalysis, Avalanche Effect

I. INTRODUCTION

Encryption is the process of converting a plaintext message into ciphertext which can be decoded back into the original message using the secret key. There are several types of data encryption schemes which form the basis of data confidentiality. Encryption schemes are generally based on either block or stream ciphers. Historically, the focus of encryption has been on the use of symmetric encryption to provide confidentiality service [3]. In the last several decades, other considerations such as authentication, integrity and digital signature have been included in the theory and practice of cryptology. The public key cryptosystems are mainly used for key exchange, authentication, and digital signature services in real time applications, because of high computational complexity. The symmetric key cryptosystems are more popular to provide the confidentiality service. A single round of encryption offers inadequate security while multiple rounds offer increased security with increased computational complexity. The SHA-512 additive constants represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers and the constants used in MD5 represent the binary integer part of the sine (radians) value of the integers 0 to 64. These constants provide a randomized set of 64-bit patterns which should eliminate any regularity in the input data [3]. In this paper, the random constants used in SHA-512 and MD5 hash functions are used as random key stream key to encrypt text data.

II. LITERATURE REVIEW

In Hill cipher algorithm, the inverse of the key metrics used for encrypting the plaintext does not always exist. If the key matrix is not invertible, then encrypted text cannot be decrypted. Bibhudendra *et al.* [1], proposed a novel Advanced Hill (AdvHill) cipher technique which uses an involutory matrix to encrypt data. Since the involutory matrix is invertible, the computational complexity can be reduced by avoiding the process of finding the inverse of the matrix at the time of decryption.

Jawahar Thakur *et al.* [2] provided a comparison between three symmetric key cryptographic algorithms such as Data Encryptions Standard (DES), Advanced Encrypted Standard (AES), and Blowfish. The comparison takes into consideration, the behavior and the performance of the algorithm when different data loads are used. The comparison is made on the basis of the parameters such as speed, block size, and key size. Blowfish has a better

performance than other encryption algorithms and AES showed poor performance results compared to other algorithms since it requires more processing power. The characteristics of AES are its security and resistance against attacks. The major characteristic of RC4 algorithm is its speed. Prabhudesai Keval Ketan *et al.* [4], developed a hybrid cipher by combining the characteristics of the algorithms AES and RC4. This hybrid method is seen to have a 20% improved speed compared to the original AES and a higher security compared to the original RC4.

Rizvi *et al.* [5] analyzed the security issues of two popular symmetric cryptographic algorithms Blowfish and CAST and then compared their efficiency for encrypting text, image, and audio with the AES algorithm across different widely used Operating Systems. For text data, all algorithms run faster on Windows XP but Blowfish is the most efficient and CAST run slower than AES. Blowfish encrypts images most efficiently on all three platforms. CAST runs faster on Windows XP than AES but on Windows Vista and Windows7, AES and CAST perform at the similar speed. For audio files, CAST performs better than Blowfish and AES on Windows XP but on Windows Vista and Windows 7, there is no significant difference in performance of CAST and AES; however Blowfish encrypts audio files at less speed.

Shashi Mehrotra Seth *et al.* [6] performed comparative analysis of the algorithms DES, AES, and RSA by considering the parameters such as computation time, memory usage, and output byte. Based on the text files used and the experimental result, the authors concluded that the DES algorithm consumes least encryption time and AES algorithm has least memory usage while encryption time difference is very minor in case of AES and DES algorithms. The RSA consumes longest encryption time and memory usage is also very high but output byte is least in the case of RSA algorithm. Vishwagupta *et al.* [7] developed a cryptography algorithm based on the block cipher concept using XOR and shifting operation. The algorithm takes less time when compared to the existing symmetric algorithms such as DJSA and AES. To encrypt a text file of size 560 KB, the proposed method takes 0.28ms but the DJSA and AES algorithms take 0.37ms and 0.35ms, respectively.

The hash algorithms such as MD5 (Message Digest 5) and SHA (Secure Hash Algorithm) are frequently used to convert the variable size input into fixed size output called digest. A hash function is defined as the mapping of bit strings of an arbitrary finite length to strings of fixed length. A hash function (H) converts a variable-length block of data M into an output of fixed-size hash value h = H(M). A change to any bit or bits in M results, with high probability, in a change to the hash code [3, 4, 9, 10]. Abbas Cheddad et. al. [9] described a new way of encrypting digital images with password protection using SHA-2 hash algorithm coupled with Fourier Transform and XOR operation. S.M Seyedzade et. al. [10] introduced an image encryption algorithm based on SHA-512 hash function. The entropy of the encrypted image is increased and both security and performance aspects are satisfactory with two rounds. Other existing text encryption methods proposed by various others are found in [11, 12, 13].

The Message Digest (MD5) algorithm has four rounds and each round is repeated for 16 times. The MD5 algorithm makes use of 64 constants each of size 32 bits constructed from the mathematical sine function. These constants provide a randomized set of 32-bit patterns which should eliminate any regularity in the input data [3]. This technique is adopted by the proposed method as random key to encrypt text messages.

The SHA-512 hash function has 80 rounds and each round makes uses a 64-bit additive constant K_t, where $0 \leq t \leq 79$, which is derived from the fractional parts of the cube roots of the first 80 prime numbers. These constants are used to provide a randomized set of 64-bit patterns which should eliminate any regularity in the input data [3, 4]. This technique is adopted by the proposed method as random key to encrypt text messages.

III. THE PROPOSED ENCRYPTION METHOD

The proposed symmetric key encryption method does block ciphering using XOR operation and extended key space to encrypt the characters which includes alphabets, numerals, and special characters. The typical working model of the proposed symmetric key encryption method is shown in Figure 1.

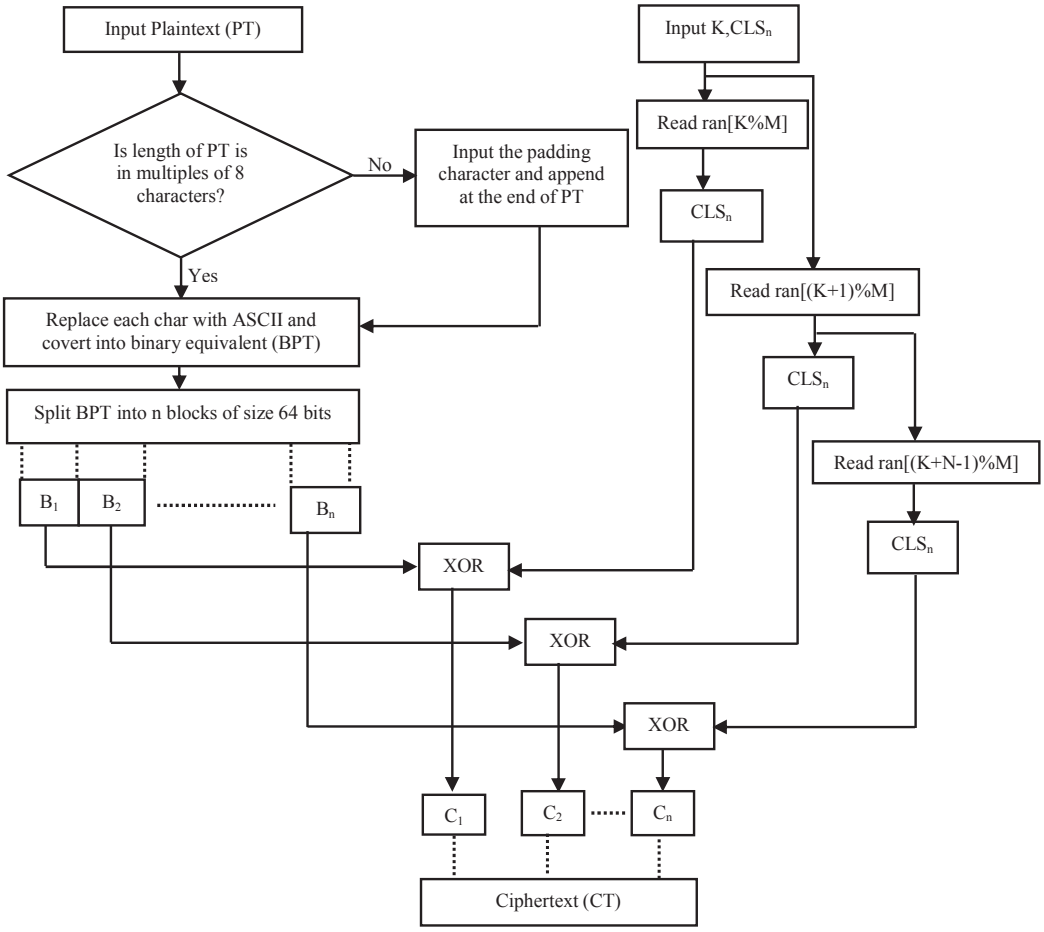

Figure 1. Block diagram of proposed encryption method

The plaintext message is divided into equal blocks of size 64 bits and block ciphering is applied to these blocks. If the length of original message is not in multiples of 8 characters, then padding characters are appended to make sure each block contains exactly 8 characters. The strength of any encryption algorithm is purely based on the key. Hence, the proposed method uses two keys to provide a stronger security, namely '**i'** which represents the i^{th} key of the key space and '**cls'** which represents the number of bits circularly left/right shifted on the i^{th} key. The circularly left/right shifted i^{th} key is used to encrypt the first block of plaintext and $(i+1)^{th}$ key to encrypt the second block of plaintext and $(i+2)^{th}$ key to encrypt the third block of plaintext and so on. Therefore, the repeated words of plaintext will not have the same ciphertext as the key differs for each block which provides a stronger security against cryptanalysis. Also, the proposed method uses characters from the infeasible range for mapping the plaintext into ciphertext which will confuse the information to intrude and remains as the strength of the suggested method.

Table – 1 Notations used in the Proposed Encryption Method

S.No.	Notation	Meaning
1	PT	Plaintext
2	BPT	Binary string representation of plaintext
3	B_i	i^{th} block of BPT, size of each B_i is 64 bits
4	K	K^{th} key to be used for encryption of the first block
5	CLS_n	Circular left shift of n bits
6	ran[]	random key stream generated from the constants of SHA-512 or MD5 hash function

7	**C_i**	**i^{th} block of binary representation of ciphertext**
8	**M**	**Number of constants (64 for MD5 and 80 for SHA-512**

A. Random Key Stream Generation using MD5

The Message Digest (MD5) algorithm is developed by Ron Rivest at MIT. There are four round and each round is repeated for 16 times. The MD5 algorithm makes use of 64 constants, each of size 32 bits constructed from the mathematical sine function. These constants provide a randomized set of 32 bit patterns which should eliminate any regularities in the input data and to offer collision resistance [3]. In the proposed method, 64 bits have been considered instead of 32 bits. The following is the algorithm to generate the constants used in MD5 algorithm which is used as random key stream in the proposed method.

Step 1: Let Random[] be the array to store the generated keys.
Step 2: Repeat for i::1 to 64
    2.1 Temp[i]←(floor(abs(sin(S_i)))*(2^64)
    2.2 Random[i]←toBinaryString(Temp[i])
    2.3 Increment i
Step 3: Return the array Random[].

B. Random Key Stream Generation using SHA-512

The most widely used hash function has been the Secure Hash Algorithm (SHA) and it is developed by the National Institute of Standards and Technology (NIST). There are 80 rounds and each round of SHA makes use of a 64-bit additive constant K_t, where $0 \leq t \leq 79$. These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers. These constants provide a randomized set of 64-bit patterns which should eliminate any regularities in the input data and to offer collision resistance [4]. The following is the algorithm to generate the additive constants of SHA-512 algorithm. These constants are used as random key stream in the proposed method.

Step 1: Let the first 80 prime numbers are stored in the array Prime[ ].
Step 2: Let Random[ ] be the array to store the random bit stream.
Step 3: Repeat for i varies from 1 to 80
    3.1 Temp1 ← cube_root(Prime[i])
    3.2 Temp2 ← Temp1%1; //To extract the fractional part from Temp1
    3.3 Key ← toBinaryString(Temp2)
    3.4 Random[i] ← MSB 64-bits of (Key)
Step 4: Return the array Random[].

C. The Encryption Algorithm

Input : Plaintext (PT), Key (K), and Circular Left Shift (CLS_n)
Output : Ciphertext (CT)

Step 1: Let PT be the input plaintext and CT be the array to store the ciphertext.
Step 2: if(length(PT)%8 != 0) then
    2.1 Input the padding character.
    2.2 Append the padding character at the end of plaintext (PT).
Step 3: Convert the PT into corresponding ASCII value and then binary format.
Step 4: Divide the binary string into blocks of size 64 bits.
Step 5: Repeat steps 6 and 7 until all blocks are processed.
Step 6: Repeat for i :: 0 to n-1
    6.1 key ←key[K+i] % tnk, where tnk represents the total number of keys (64 or 80).
    6.2 CLS_n (key)
Step 7: CT ← Concatenate (CT, BitXoR(B_i, key))

Step 8: Return the Ciphertext (CT)

The decryption process is just the inverse of the encryption function.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In the proposed method, different keys are used to encrypt different data blocks of the plaintext message so the system is less vulnerable to cryptanalysis attacks. The repeated characters of plaintext space are mapped to different characters in the ciphertext space even though they appear in the same block or different blocks. Table 2 gives the results of encrypting the messages "meet me afer the toga party" and "attack postponed to two am" using the random key streams generated from both the constants used in MD5 and SHA-512 hash functions. The result shows that the repeated characters are mapped to different cipher characters. Hence, the ciphertext is not easily amenable to cryptanalysis and less vulnerable to letter frequency attack.

Table – 2 Security Against Cryptanalysis

Key (K, CLS_n)	**45, 34**
Plaintext	**Meet me after the toga party**
Ciphertext (SHA constants)	**8óÁ77y÷)¡òß¸]¢z-63ÞøÀ**
Ciphertext (MD5 constants)	**-þöñì+àìbÒ`Å**
Decrypted Text	**Meetmeafterthetogaparty**
Plaintext	**Attack postponed to two am**
Ciphertext (SHA constants)	**âÐ"9hþ.°ðÄ¾|²a¥1ÇÀ**
Ciphertext (MD5 constants)	**ïøøý)ûìeÜsÜY**
Decrypted Text	**Attackpostponedtotwoam**

A. Testing of Avalanche Effect

The change of a single character in the plaintext message affects the subsequent characters to produce different ciphertext and this is given in Table 3 and Table 4. Hence, the proposed algorithm has achieved a good avalanche effect which is one of the desired qualities of a good encryption algorithm.

Table – 3 Testing of Avalanche Effect using SHA Constants

Plaintext	***meet* me after the toga party**
Key (K, CLS_n)	**25, 16**
Ciphertext	**rMÌ^êlDâ²eûfò-ô†˜**
Plaintext	***met* me after the toga party**
Key (K, CLS_n)	**25,16**
Ciphertext	**rMÝVø}Bþ¿tó`ã>ùÿ˜**

Table – 4 Testing of Avalanche Effect using MD5 Constants

Plaintext	***meet* me after the toga party**
Key (K, CLS_n)	**25, 16**
Ciphertext	**¤ômeâ¶A&´he÷²Èkártú¢**
Plain Text	***met* me after the toga party**
Key (K, CLS_n)	**25,16**
Cipher Text	**¤íeaå¤P1¨etiºÌzòty¢**

B. Testing of Key Sensitivity

A small change in the encryption key (k) and circular shift (CLS_n) should produce a completely different ciphertext. Table 5 and Table 6 give the ciphertext produced for the plaintext message "meet me afer the toga party" for different key pair with minor variation.
The hamming distance between the ciphertext produced by the key pairs (24, 15) and (25, 16) is 87, and similarly, between the ciphertext produced by the key pairs (25, 16) and (26, 17) is 78 for the random key stream generated by using the SHA-512 constants.

The hamming distance between the ciphertext produced by the key pairs (24, 15) and (25, 16) is 65, and similarly, between the ciphertext produced by the key pairs (25, 16) and (26, 17) is 87 for the random key stream derived by using the constants used in MD5 algorithm. Thus, the proposed method has good key sensitivity and also satisfies the avalanche property significantly.

Table – 5 Testing of Key Sensitivity using SHA Constants

Plaintext	**meet me after the toga party**
Key (K, CLS_n)	**24, 15**
Ciphertext (CT_1)	**¡5åÅ-Ý{ñ&ÏñÕ©kë*tÊ**
Key (K, CLS_n)	**25,16**
Ciphertext (CT_2)	**rMÌ^êlDâ²eûfò-ô†˜**
Key (K, CLS_n)	**26,17**
Ciphertext (CT_3)	**\£YÙdNwBºid_>-·ßu?**
Hamming distance between CT_1 and CT_2	**87 bits**
Hamming distance between CT_2 and CT_3	**78 bits**

Table – 6 Testing of Key Sensitivity using MD5 Constants

Plaintext	**meet me after the toga party**
Key (K, CLS_n)	**24,15**
Ciphertext (CT_1)	**tme¦´he5ýÀÀrt8î**
Key (K, CLS_n)	**25,16**
Ciphertext (CT_2)	**¤ômeâ¶A&´he÷²Êkártú¿**
Key (K, CLS_n)	**26,17**
Ciphertext (CT_3)	**ã¦ômdfÜ.qthds*òáru~"**
Hamming distance between CT_1 and CT_2	**65 bits**
Hamming distance between CT_2 and CT_3	**87 bits**

C. Testing of Active Attacks

Repeated words present in the same plaintext block or different blocks are mapped into different ciphertexts. Hence, active attacks such as chosen plaintext and chosen ciphertext attacks are quite difficult to succeed. Thus, the proposed method is less vulnerable to active attacks and the same is given in Table 7. Since the time required to encipher or decipher a data block is same for all data blocks, the proposed method is less prone to timing attacks.

Table – 7 Security Against Active Attacks

Plaintext	**Alice has the *key* for encryption.*Key* is of 24 bits**
Key (K, CLS_n)	**67, 36**
Ciphertext	**ñc{7»jÎe%a-Ú¥/aÇzÛFÍiÞ{h0´**
Decrypted Text	**Alicehasthekeyforencryption.Keyisof24bits**

D. Key Space Analysis

The size of a single key is 64 bits and the total numbers of keys generated from the MD5 and SHA-512 constants are 64 and 80, respectively. Hence, the total key space is $60\times2^{64}+80\times2^{64}$ which is approximately 2^{71} and this is enough to resist the exhaustive key search attack.

IV.CONCLUSION

A symmetric key cryptosystem to encrypt text data by using the random constants used in SHA-512 and MD5 hash functions is proposed in this paper. The method is very simple and efficient because of the XOR operation and extended key space. The observation illustrates the generated ciphertext is junk characters or special characters which does not provide any information directly to the intruders. The proposed method is safe against cryptanalysis attack and has good avalanche effect. It is found that the suggested method is very much sentitive with respect to the encryption key. The algorithm satisfies both confusion and diffusion properties significantly.

REFERENCES

[1] Bibhu Dendra, Soraj Kumar Panigrahy, Sarat Kumar Patra and Ganapati Panda, "Image Encryption using Advanced Hill Cipher Algorithm", International Journal of Recent Trends in Engineering, Vol.1, No.1, 2009.

[2] Jawahar Thakurand Nagesh Kumar, "DES, AES, and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis", International Journal of Emerging Technology and Advanced Engineering, Vol. 1, No. 2, 2011.

[3] William Stallings, "Cryptography and Network Security", Pearson Education, New Delhi, 2005.

[4] William Stallings, "Cryptography and Network Security – Principles and Practice", Pearson Education, New Delhi, 2013.

[5] Prabhudesai Keval Ketan and Vijayarajan V, "An Amalgam Approach using AES and RC4 Algorithms for Encryption and Decryption", International Journal of Computer Applications, Vol.54, No.12, 2012.

[6] S.A.M Rizvi, Syed Zeeshan Hussain and Neeta Wadhwa, "A Comparative Study of Two Symmetric Encryption Algorithms Across Different Platforms", International Conference on Security and Management (SAM'11), World Academy of Science, USA, 2011.

[7] Shashi Mehrotra Seth and Rajan Mishra, "Comparative Analysis of Encryption Algorithms for Data Communication", International Journal of Computer Science and Technology, Vol. 2, No.2, 2011.

[8] Vishwagupta, Gajendra Singh and Ravindra Gupta, "Advance Cryptography Algorithm for Improving Data Security", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 1, 2012.

[9] Abbas Cheddad, Joan Condell, Kevin Curran and Paul McKevitt, "A hash-based image encryption algorithm", Journal of Optics Communications, vol. 283, no. 6, pp. 879-893, 2010.

[10] S.M Seyedzade, R.E Atani and S Mirzakuchaki, "A novel image encryption algorithm based on hash function", Machine Vision and Image Processing (MVIP), Iranian - Isfahan, pp. 1-6, 27-28 Oct. 2010.

[11] Jeyamala Chandrasekaran, Subramanyan B, and Raman G.S, "Ensemble of Blowfish with Chaos Based S-Box Design for Text and Image Encryption", International Journal of Network Security & its Applications (IJNSA), Vol.3, No.4, pp. 165-173, 2011.

[12] Sumira Hameed1, Faisal Riaz2 ,Riaz Moghal3, Gulraiz Akhtar4, Anil Ahmed5, Abdul Ghafoor Dar6, "Modified Advanced Encryption Standard For Text And Images", Computer Science Journal Volume 1, Issue 3, December 2011.

[13] Rohan Rayarikar , Sanket Upadhyay , Priyanka Pimpale," SMS Encryption using AES Algorithm on Android", International Journal of Computer Applications (0975 – 8887) Volume 50– No.19, July 2012.