2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

Lightweight Data Security Model for IoT Applications: A Dynamic Key Approach

Manish Kumar, Sunil Kumar, M.K. Das and Sanjeev Singh
Institute of Informatics and communication,
Cluster Innovation Center,
University of Delhi South Campus, New Delhi, India
Email: sanjeev@south.du.ac.in.

Rajat Budhiraja
Aricent Technologies,
India

***Abstract*—A new symmetric encryption model for enhanced security is designed for the data of any size and type. It is envisaged that the proposed encryption model would enable a range of IoT devices to transfer data or to communicate more securely over open/insecure communication channel and also to provide enhanced data integrity. The model keeps the size of output cipher text unchanged by XORing the parity bits to avoid the overhead and making it lightweight. The chaotic dynamical system was used for key scheming to produce highly sensitive cipher text. Further, the key is dynamically updated using the same parity to increase the reliance of output cipher text on the input plain text. The key with 128-bit length is used for data complexity which is widely known and proven to withstand any kind of brute force attack. Finally, diffusion process was implemented to ensure the effect of any change is retained and carried throughout the process. The simulation results yield average NPCR value to be 99.7113, which demonstrates the potential of the model to be used for IoT based devices and systems where high complexity for data security is required. Different simulation exercises were carried out for change in key, plain text and cipher text and the results are discussed in the paper.**

Keywords: Lightweight Encryption; IoT, Data Security; Data Integrity; Dynamic Key.

I. Introduction

Data security in the digital world has been an area of major concern globally due to the risk and stakes involved in it. It is becoming highly sensitive with emerging heterogeneous technology framework such as IoT with device to device communication. It is intended to make human lives more simpler but at the same time introducing new challenges to make information more secured. Noticeable work has been done to protect the data or information with various encryption techniques which can be found in the literature. Many data encryption methods based upon symmetric key encryption have been reported in recent years, which uses confusion or/and diffusion processes for data encryption [1]. Some of the methods use substitution techniques for achieving better encryption quality. Owing to their high degree of randomness and sensitivity to initial conditions, chaos based data encryption methods have emerged rapidly [2-5]. The control parameters and initial conditions of these dynamical systems are generated from the key [6], hence making these systems more key dependent. Many crypto systems use the binary key to generate sub keys and these in turn are used for different parameter generation [7-9].

The present work focusses on chaos based data encryption to provide data security in IoT devices. The method uses multiple chaotic dynamical systems for various processes. A 128-bit key is used for initial and control parameter generation which is sufficient enough to resist brute force attack. The encryption quality is strengthened by confusion and diffusion properties of the cipher text. Encryption key is continuously updated with the parity for each input data block, thereby increasing the cipher text dependency on the dynamically updated key. XORing randomly generated key dependent data further subjected to shuffling process derived from logistic map enables the model to resist chosen and known plain text attacks. It is envisioned that this model shows its suitability for systems/devices having smaller computational footprint.

In this paper, section II describes the key generation process and section III gives an overview of shuffling process. Complete diffusion procedure is explained in section IV. Sections V and VI briefly describe encryption and decryption processes respectively. Results and discussion are summarised in section VII and conclusions made in section VIII.

II. Key Generation

As the chaotic systems are totally dependent on their initial conditions and control parameters, the cipher text becomes highly key dependent. A random data of 128-bit length was chosen and divided into sixteen 8-bit sub keys as: -

$$k_i = key(8 \times i + 1) \quad \text{to} \quad key(8 \times (i + 1)) \tag{1}$$

where, i ranges from 0 to 15.

The initial parameters for the different chaotic systems are obtained in [10]:

$$x_1 = ((k_1 \times k_5)/(k_4 + k_{16})) \quad Mod \quad 1 \tag{2}$$
$$x_2 = ((k_3 \times k_9)/(k_{10} + k_{15})) \quad Mod \quad 1 \tag{3}$$
$$x_3 = ((k_7 \times k_{11})/(k_6 + k_{12})) \quad Mod \quad 1 \tag{4}$$
$$y_1 = ((k_{16} \times k_{13})/(k_5 + k_9)) \quad Mod \quad 1 \tag{5}$$
$$z_1 = ((k_2 \times k_{10})/(k_8 + k_{14})) \quad Mod \quad 1 \tag{6}$$

978-1-5090-5880-8/16 $31.00 © 2016 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.100

Scaling/control factors are computed as: -

$$s_1 = (k_{13}/k_9) \times (k_{16} \times k_{14}) + 113 \quad (7)$$
$$s_2 = (k_{10}/k_7) \times (k_{11} \times k_{15}) + 117 \quad (8)$$
$$s_3 = (k_{14}/k_4) \times (k_2 \times k_{16}) + 131 \quad (9)$$

here, $0 < x_1$,x_2, x_3, y_1 and $z_1 < 1$.

The above factors are used to calibrate the output parameters for controlling their values within desired limits, which are to be used in subsequent processes.

To begin with, the first 8 bytes of data (data block) is XORed and shuffled (as described in section III) for encryption. The parities of these shuffled bytes are calculated and XORed with the initially chosen 128-bit data. The resulted output was then used in generation of initial and scaling/control parameters for the subsequent data block. This process is applied till the last data block is XORed and shuffled.

III. Shuffling Procedure

The shuffling process shuffles the input data based on the key used hence, one can easily reverse this shuffling at the receiving end for decryption process. In shuffling process, two random arrays (array1, array2) are generated using intertwining logistic map. The input data in positions indicated by the elements in array2 are relocated to positions indicated by the elements in array1. The maximum value of an element in both these arrays can equal the maximum number of elements in input data.

Intertwining logistic map used for generating array1 and array2 is:-

$$x_{n+1} = [\mu \times y_n \times (1 - xn) \times k_1 + z_n] \quad mod \quad 1 \quad (10)$$
$$y_{n+1} = \left[\mu \times y_n + z_n \times k_2 \times 1/(1 - x_{n+1}^2)\right] \quad mod \quad (11)$$
$$z_{n+1} = [\mu \times (x_{n+1} + y_{n+1} + k_3) \times \sin(z_n)] \quad mod \quad (12)$$

where, $0 \leq \mu \leq 4$, $|k_1| > 33.5$, $|k_2| > 37.9$, $|k_3| > 35.7$, and x_1, x_2, x_3 are the initial conditions generated using the key.

Figure 1, shows the effect of shuffling 1 time, 10 times, 100 times and 1000 times on same data. One can notice from the Figure 1(a to e) that each instence of shuffling of data, irrespective of the number of shuffle, the randomness remains consistence.

Figure 2 shows the Bifurcation plot for intertwining logistic map. It is clearly evident that the distribution of values for the above parameters are uniform thereby, reducing the probability of frequency attack.

IV. Diffusion Process

A Lozi map [9] is used for diffusion process due to its high degree of randomness. For each symbol x and y, values are first generated and then used for diffusion. Lozi map can be explained by following equations: -

$$x_{n+1} = 1 - \alpha|x_n| + y_n \quad (13)$$
$$y_{n+1} = \beta x_n \quad (14)$$

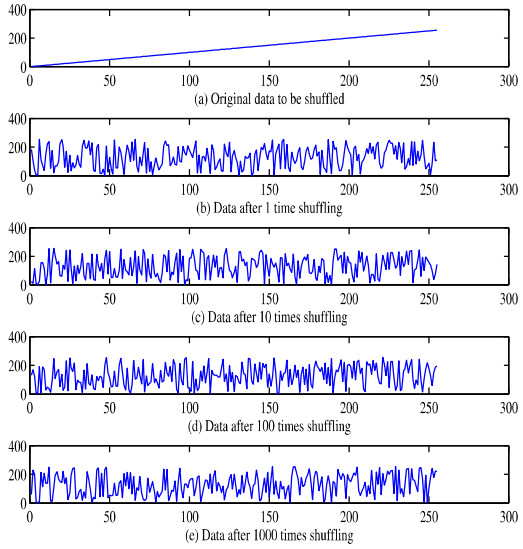

Fig. 1. (a) Original data to be shuffled. (b to e) Effect of shuffling process on data while shuffling multiple times.

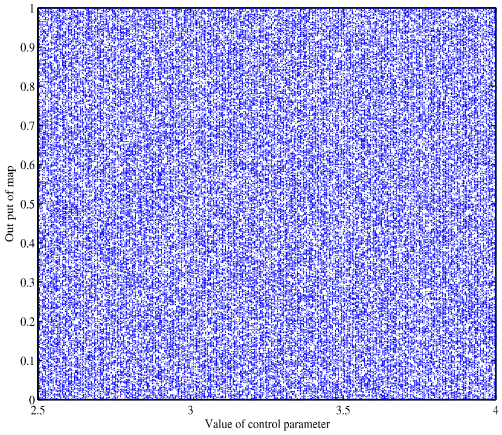

Fig. 2. Bifurcation of intertwining logistic map.

where α = 1.4 and β = 0.3 is used. The values of x are first scaled up using scaling factor s_2 and then mod 256 is taken, whereas, the values of y are scaled up using scaling factor s_3 and then mod 256 is taken.

Figure 3(a) and (b) shows the lyapunov exponent and bifurcation plot for Lozi map. It is clear from both the plots that generation of input parameters leads to random output; hence, diffusion process takes its input from a random process.

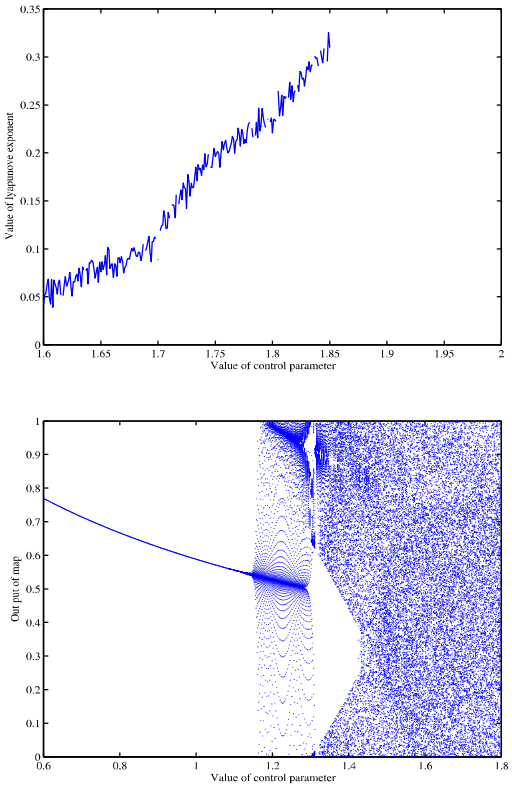

Fig. 3. Plot of lyapunov exponent(above) and bifurcation diagram of Lozi map(below).

$$d\_data_i = (data_i + |x_{scaled}| \times |y_{scaled}|) \quad mod \quad 256 \quad (15)$$

where, d_data_i is the generated diffused data after the diffusion of $data_i$. The original data (O_data) can be recovered at the recieving end using the following equation.

$$d\_data_i + 256 - (d\_data_i + |x_{scaled}| \times |y_{scaled}|) \, mod 256 \quad (16)$$

V. Encryption Process

Encryption involves shuffling and XORing a randomly generated data with the original plain text data blocks. This is followed by diffusion process. Following are the sequential steps performed during the encryption process:-

1. Input data is divided into 8 byte data blocks and necessary padding of 0s is added if required.

2. A random sequence of length 8 byte is generated using logistic map and is scaled up by multiplying with s_1 and taking mod 256.

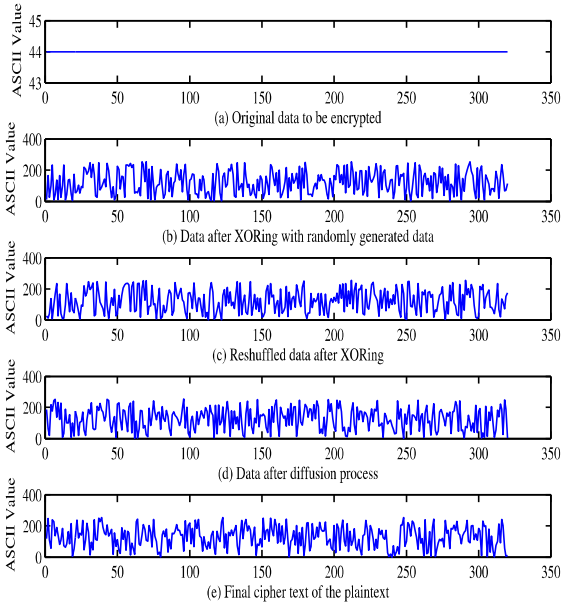

Fig. 4. Plot of data during different steps of encryption process.

3. The first input data block is shuffled and bitwise XOR operation is performed with sequence generated as output from step 2.

4. The XORed data is then shuffled back to get its arrangement as that of the input block in step 3.

5. The parity for each block element is calculated and arranged as byte, which is then XORed with each sub key.

6. Steps 2 to 5 are repeated for all the input data blocks.

7. Diffusion parameters are generated using the initial 128 bit data. Diffused data is then obtained by the application of diffusion process defined in section IV.

8. Final cipher text is obtained by concatenating the diffused data blocks into a data stream and then shuffling using scaling factors s_2 and s_3.

Figure 4 shows the input plain text and output cipher text plots during different steps of encryption.

VI. Decryption Process

Decryption involves the reverse procedure to that of encryption. It primarily requires the initial key used for encryption of first data block of plain text. Following are the sequential steps performed for successful decryption of cipher text:

1. Shuffling of cipher text is first done using array 1 and array 2 (described earlier in section III). Since this follows the reverse path of encryption, these arrays are interchanged before shuffling the cipher text data.

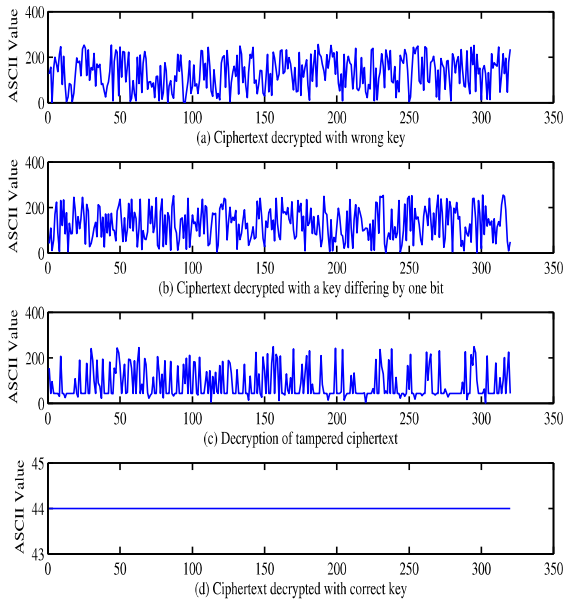

Fig. 5. Plot for different test cases while decrypting the cipher text.

2. The resulting data is then divided into 8 byte data blocks.

3. Reverse of diffusion process is applied to each data block as stated in eqn.(16).

4. XORing process was then carried out on parity of each data block and initial key for subsequent data block processing.

5. The shuffling process of step 1 is carried out on the data block resulting from step 3.

6. A random 8 byte data value is obtained using Logistic map, which is scaled up as in step 2 [cf. section V].

7. The shuffled data block of step 5 has been XORed with the scaled up value, and further shuffled to get the original plain text data block.

8. Step 3 to 7 are repeated untill all the data blocks are processed.

VII. Results and Discussion

The proposed algorithm was examined for various practical negative use cases such as wrong key, wrong cipher text, minor change in key for encryption and decryption and minor change in input plain text. It was observed that, in all such cases the model was able to detect the changes and corresponding ciphertext could not be correctly decrypted. Figure 5(a-d) shows the decrypted ciphertext corresponding to the use cases mentioned above.

As the input plain text and cipher text size is equal, NPCR score can be calculated and is found to be in the range of 99.568 to 99.7447. As the encryption process involves diffusion of data, a small modification in the input plain text is reflected to a large extent in the final cipher text. An incorrect key or tampered cipher text leads to the false decryption of data. Similar input texts also lead to different cipher texts because of the shuffling and XORing followed by parity calculation.

Data Size (kb)	Time in s. [4]	Time in s. [7]	Time in s. [10]	Time in s. [PM*]
10	**	**	0.071465	0.0329
30	0.27	0.27	0.185454	0.0887
90	1.03	0.79	0.568465	0.1932
240	2.75	2.10	1.835408	0.5226

TABLE I
**: Data not available, PM*: present Model.

Table 1: The comparison of encryption times of the present model with other models, configuration used in PM* is a system running on 700MHz and 512MB RAM.

VIII. Conclusion

A lightweight encryption model based on dynamic key update and multiple chaotic dynamical systems demonstrated the strong possibility for IoT domain where a smaller process footprint is required to protect the valuable user information. The model uses Logistic map, Lozi map and intertwining logistic map at different stages to introduce higher level of randomness in the system. The control parameters for each of these maps are derived to achieve highest chaotic behaviour. The results in section VI and VII shows the process is highly sensitive to even minor changes in key. Key length is self sufficient to address security issues related to brute force attack. The output cipher size is same as that of the input plain text size which ensures savings in the network bandwidth for transmitting the cipher text when compared to plain text. From Table 1, it is evident that the time taken in encryption is substantially smaller when compared with other models. Therefore, this lightweight model shows a prominent applicability in the IoT based applications/systems requiring enhanced security with smaller footprint.

Acknowledgment

Authors acknowledge the support provided by University of Delhi for this research work under DU DST Purse Phase-II (381/382/22-09-2015). Manish Kumar acknowledges with thanks to UGC for financial support under RGNF-2014-15-DEL-SC-59666/SA (III).

References

[1] X. Wnag, X. Wang, J. Zhao, Z. Zhang, Chaotic encryption algorithm based on alterant of stream and block cipher, Nonlinear Dynamics 63(4), 2011, pp. 587-597.

[2] E. Alwarez, A. Fernandez, P. Gracia, J. Jimenez, A. Marcano, New approach to chaotic encryption, Physics Letters A 263(4), 1999, pp. 373-375.

[3] K.W. Wong, A fast chaotic cryptographic scheme with dynamic look-up table, Physics Letters A 298(4), 2002, pp. 238-242.

[4] N.K. Pareek, V.Patidar, K.Sud, Discrete chaotic cryptography using external key, Physics Letters A 309(1), 2003, pp. 75-82.

[5] M. Baptista, Cryptography with chaos, Physics Letters A 240(1), 1998, pp. 50-54.

[6] M. Kumar, S. Kumar, R. Budhiraja, M. K. Das and S. Singh, Intertwining logistic map and Cellular Automata based color image encryption model, 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, 2016, pp. 618-623.

[7] N. Pareek, V. Patidar, K. Sud, Cryptography using multiple one-dimentional chaotic maps, Communications in Nonlinear Science and Numerical Simulation 10 (7), 2005, pp. 715-723.

[8] N. K. Preek, V. Patidar, K. K. Sud, Block cipher using 1D and 2D chaotic maps, International Journal of Information and Communication Technology 2 (3), 2010, pp. 244-259.

[9] R. Lozi, UN ATTRACTEUR TRANGE (?) DU TYPE ATTRACTEUR DE HENON, J. Phys. Colloques 39 (C5), 1978, C5-9-C5-10.

[10] Manish Kumar, Sunil Kumar, Rajat Budhiraja, M.K. Das, Sanjeev Singh, A cryptographic model based on logistic map and a 3-D matrix, journal of information security and applications (2016), doi: 10.1016/j.jisa.2016.09.002