International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Apr-2022)

https://dx.doi.org/10.12785/ijcds/1101106

Keyboard Encryption Algorithm Based on Software Engineering Security

Muayad Sadik Croock[1]

[1]*Department of Control and System Engineering, University of Technology-Iraq, Baghdad, Iraq*

Received 22 Jul. 2021, Revised 20 Jan. 2022, Accepted Mar. 20122, Published 15 Apr. 2022

Abstract: It is well known that nowadays the security issues in the information system are increased in noticed way. These issues include the hardware and software sides, in terms of data or information as well as the devices. One of the most important parts that can be used to tackle these issues is the keyboard. The attacks on the keyboard, such as key stroke and key logging, need to be tackled. In this paper, an encryption algorithm for keyboard is proposed to guarantee the security in the output keyboard from attackers. The proposed algorithm is built based on software engineering model in terms of structure and formulation. This is to increase the flexibility of development, scalability to involve more number of users (keyboards), and reliability of employing in real-time system due to its light and efficiency. The proposed algorithm uses the interleaving and XOR processes in encryption, where the interleaver uses random seeds in generating specific indexing orders. It is tested among numerous case studies including the randomness of the generated indexing orders that are passed through different tests. The efficiency of the proposed algorithm is tested over many experiments to obtain more than 98%. Moreover, the consumed time for encryption and decryption processes is checked as well to obtain that the decryption process takes less time.

Keywords: Software Engineering, Encryption, Interleaving, security, Keyboard structure

1. Introduction

The security of data is currently being the most important challenge in the information systems. This is due to the increase in the number of attacks and the greedy requests of getting data of other systems by attackers. The security can be performed in different parts of the information systems including the hardware and software. One of these parts is the keyboard that is based on converting each key to equivalent ASCII code. Therefore, the encryption can be used in securing the output of keyboard for specific systems. In addition, the designed algorithms of encryption can be structured based on software engineering. These algorithms can guarantee the reliability, flexibility, and scalability, in which the aim of them is satisfied in optimal way [1], [2], [3], [4], [5].

In this paper, an encryption algorithm for securing keyboard is proposed based on software engineering model. This algorithm is based on changing the ASCII code for the produced character from keyboard in proposed encryption steps. In this way, the randomize of producing different ASCII code for the keys in keyboard is increased to prevent the easy keystroke and password steeling. The software engineering model here is used for ensuring the right structure of the proposed algorithm, in which the flexibility, scalability and reliability are guaranteed.

The rest of the paper is organized as follows: Section 2 covers the related research works so far; Section 3 outlines the proposed algorithm that includes the software engineering design model and the work flow of the working steps; Section 4 presents the obtained results of the proposed algorithm; Section 5 describes conclusions of the presented work.

2. Review of Related Works

Different research works have considered the encryption of keyboard output in distinct fields. This is to cover the possible solutions for the issues that are occurred throughout the printing process. In [6], the robot machine was proposed, in which the text was read in a way similar to keyboard encryption. While in [7] and [8] a virtual keyboard authentication was proposed to prevent the key loggers and other attackers from steeling the user name and password. Although the proposed authentication algorithm offered a form of security, it still suffered from different disadvantages, such as click based screenshot capturing. These disadvantages can be reduced using the dynamic generation of keyboard structure each time there is an access. In [9], the Screen Pass technique was adopted in securing the touch screen based devices, such as mobile and tabs. This technique tackled different issues in security including password tagging, and optical character recognition. Authors produced a prototype that involved two

E-mail address: muayad.s.croock@uotechnology.edu.iq

user sites. Based on this prototype, the proposed technique was evaluated. The authors of [10] proposed QWERTY cipher that extended the well-known English 26 characters to 36 characters and produced the QWERTY keyboard that uses the n key in ciphering. In [11], the structure of the keyboard interaction was changed in a modern imagination view. The restructuring included producing a novel mapping of characters. The authors of [12] proposed a biometric based authentication system. It adopted the users keystroke pattern as important identifier to recognize the users. This was based on discovering that each user has its own way in using the keyboard. The obtained results proved the efficiency of the proposed method. In [13], static and dynamic malwares analysis were adopted to produce system analyzer. The information of the keystroke and key-logger used through the monitoring keyboard was sent using email for real-time analysis guarantee. Moreover, in [14], an overview of keylogging malware was produced. In this overview, the most of protocols, used in securing the system against the keylogging have been studied. This study provided the researchers with the idea of important of keyboard securing. In [15], a software and hardware development represented as a hidden key logger device was detected using the proposed algorithm. This algorithm analyzed the performance of a keyboard and detect the hidden devices related to keyboard based on this analysis. The analysis report included the expected structure and the block diagram of the detected hidden device. In [16], a monitoring system for models with keyboard was presented in addition to offering a security using key logger. Different companies were involved in presenting this system to supervise their products well. In [17], [18], [19], and [20], the review and study research works have been introduced with high covering to the subject within the subject of data encryption. It is important to note that these studies encouraged researchers to provide more efforts on enhancing the encryption methods to be strong and light. At the other hand, the authors of [21] introduced a hiding of images in excel data sheet as a method that supported the security in information transmission. While in [22], a modified butterfly architecture that was used in fast Fourier Transform (FFT). This butterfly was employed in cryptography to cipher the transmitted information in efficient way. In [23], a multi-level structure was proposed for registering the images and extracting the important parameters form the used keypads in secured approach. The authors [24] produced a key-logger method that prevented the keystroke and screenshots in different systems. All these events were performed throughout the email services. Moreover in [25], an easy in use and cost efficient broken-stroke method was proposed that could catch the pre-defined typed keywords from the transmission wireless channel between the keyboard and other devices. In addition, authors of [26] presented a keystroke approach that guaranteed the quality of insurance in information exchange. It worked in different phases to work on computers, and phones.

Authors of [27], [28], [29], [30], and [31] discussed the development of software engineering technologies and methods, in which the adopted algorithms were designed accordingly to be more efficient. These methods supported the performance enhancement in a way that the proposed system be save and performed.

3. Proposed Keyboard Encryption Algorithm

The proposed encryption algorithm is allocated inside the keyboard software part to change the traditional ASCII code of the printed characters. The software engineering is used for completing the design in efficient way. This section is divided into two sub-sections for easing the reading flow.

A. *Software Engineering Algorithm Structure*

As mentioned earlier, the presented software engineering security algorithm is structured based on software engineering modeling. This is to ensure the building of algorithm in a right way.

Figure 1 illustrates the structure of the proposed algorithm that is based on software engineering.

The structure includes the four phases of software life cycle model, which are requirements, design, implementation as well as testing and evaluation. In the requirement phase, the adopted ASCII code is collected to complete the imagination of the working flow of the proposed algorithm. In addition, the required information regarding the algorithm structure is prepared. The seeds for generating the interleave indexing orders are prepared. The interleaving is used as important step in the proposed algorithm that is explained in the next sub-section. It is well-known that the requirements phase plays as the core of the design and implementation of the underlined systems. In this work, the requirements are employed in pointing out the needed software components that can be used in implementing the proposed keyboard encryption method. In the same way, the testing of the designed and implemented work is highly recommended that employs the requirements in mapping out the case studies, environments and components. This is to ensure the right presentation and analysis of the proposed method. In terms of the design phase, the proposed algorithm is designed to include the working steps of encryption and decryption algorithms. The design process is established based on the requirements that are explained earlier. Each part of the design phase is mapped to the related requirements at the top and related implementation part as well to be in a connected chain. The designed algorithm is implemented in the implementation phase. The designed working steps are implemented in precise way to ensure the efficiency is satisfied. In the same way, the implementation parts are connected at the top with the related parts of design phase. While these parts are related to the testing components, used for evaluating the whole system. To check that the requirements are satisfied throughout the design and implementation phases, the algorithm is tested and evaluated. In case the evaluation results are not well, these results are reflected on the developing design

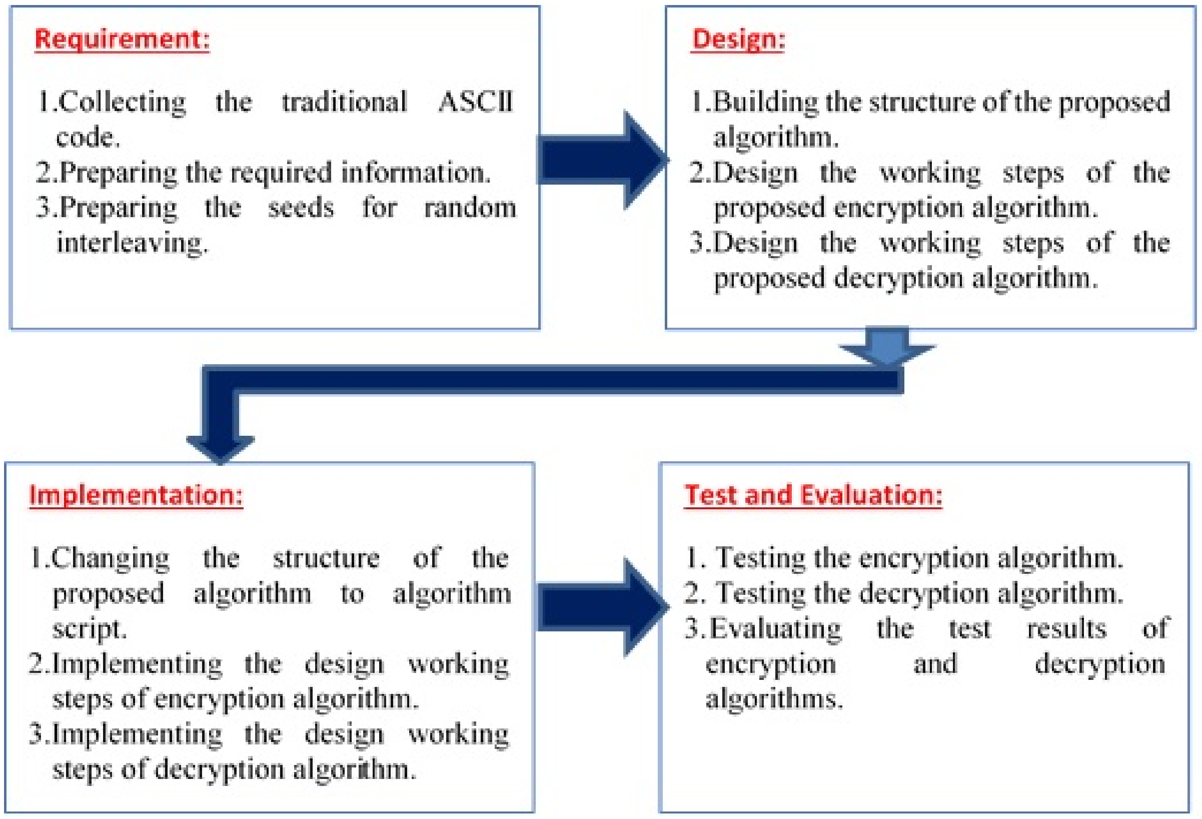
Requirement:
1.Collecting the traditional ASCII code.
2.Preparing the required information.
3.Preparing the seeds for random interleaving.
Design:
1.Building the structure of the proposed algorithm.
2.Design the working steps of the proposed encryption algorithm.
3.Design the working steps of the proposed decryption algorithm.
Implementation:
1.Changing the structure of the proposed algorithm to algorithm script.
2.Implementing the design working steps of encryption algorithm.
3.Implementing the design working steps of decryption algorithm.
Test and Evaluation:
1. Testing the encryption algorithm.
2. Testing the decryption algorithm.
3.Evaluating the test results of encryption and decryption algorithms.

Figure 1. Software engineering algorithm structure

and implementation phase to come out with the satisfied requirements.

B. *Proposed Algorithm*

In order to explain the proposed encryption algorithm in steps, Figure 2 shows the working flow block diagram.

From Figure 2 the following working steps of the proposed keyboard encryption algorithm can be explained:

- The user in real-time operation presses on a key in the keyboard.
- This key (character) is converted to 8-bit ASCII code.
- The obtained ASCII code enters to the interleaver that reordering the 8-bit according to the random operation.
- The indexing order form of the interleaver is generated using a set of random seeds. This set is available at both encrypter and decryptor. The random is adopted to increase the randomization of the interleaved ASCII, in which the guess of the character is being hard. The interleaving process is performed using the following equation:

$$S(t) = S(i) \tag{1}$$

Where, S is the bit stream of the real-time pressed characters. In addition, t= 1, …., N is the 8-bit interleaving indexing order. Moreover, i= 1, . . . , N is the real indexing of the characters ASCII code stream. N=8 is the number of considered codes.

- At the encrypter side, the selected seeds code is sent as a side information to the decryptor. The seed code is the sequence number of the selected one.
- After obtaining the interleaved ASCII code of a character, a XOR operation is performed on the interleaved code with itself with right circular shift by one as shown in Figure 3 The circular shift is performed for one input before the preceding of XOR operation.
- The result of the XOR is formatted with the side information to be ready for sending.
- At the decryptor side, the XOR operation is performed on the received 8-bit code as a step in decryption operation.
- The received 8-bit code from the XOR gate is entered to the de-interleaver.
- The de-interleaver uses the same seeds in performing

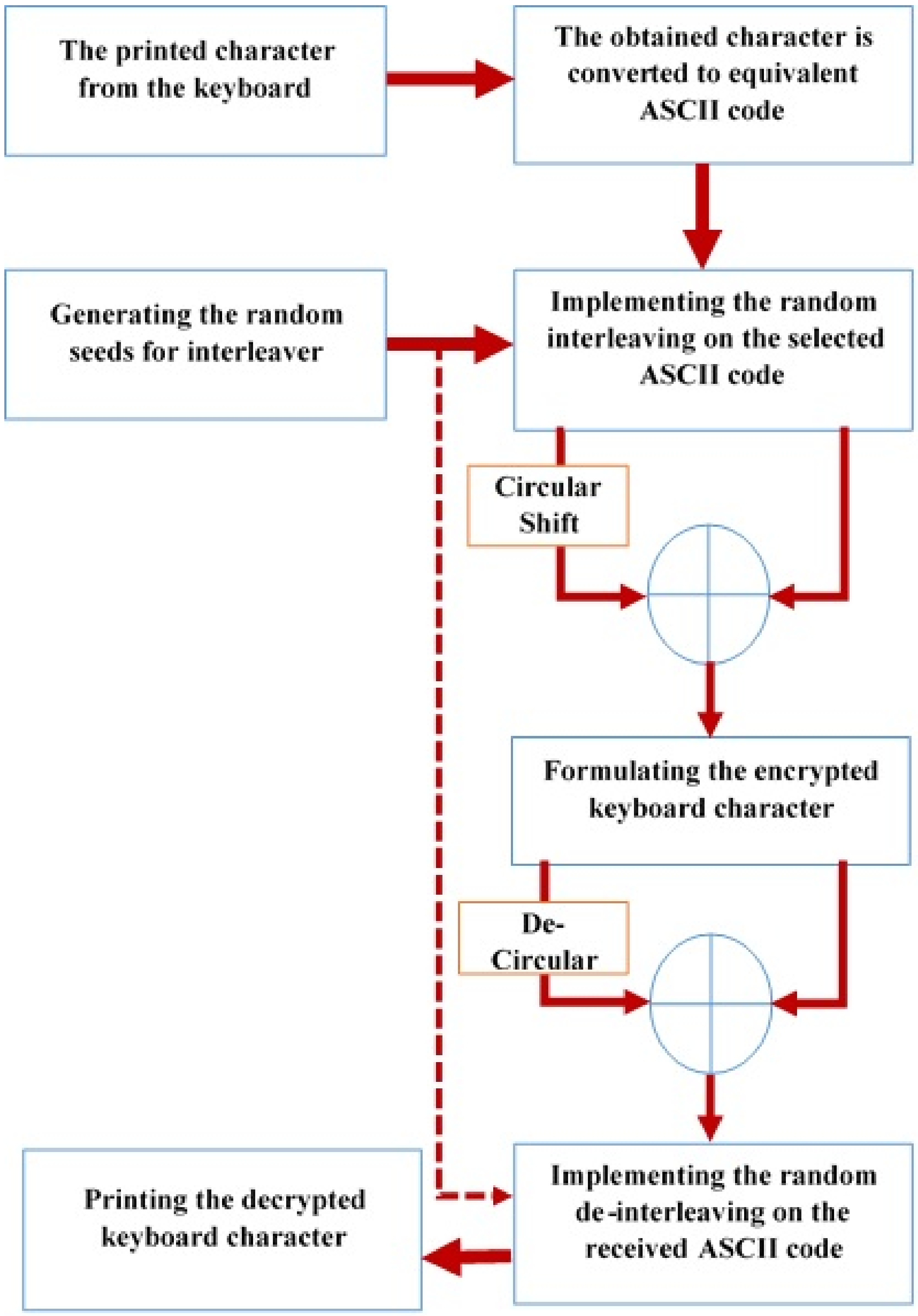
The printed character from the keyboard
The obtained character is converted to equivalent ASCII code
Generating the random seeds for interleaver
Implementing the random interleaving on the selected ASCII code
Circular Shift
Formulating the encrypted keyboard character
De-Circular
Printing the decrypted keyboard character
Implementing the random de-interleaving on the received ASCII code

Figure 2. Algorithm working flow block diagram

the de-interleaving operation.

- The final result is obtaining the same character that is already pressed on keyboard in real-time.
- This operation is continued whilst the user uses the proposed encrypted keyboard.

From Figures 1 and 2 explained above, it is concluded that the proposed system is managed well from the first step of designing. This is the results of following the software engineering life cycle model that classifies the working plan in sequential steps.

Moreover, the software engineering view provides the designer with a wide angle of imagination to build an algorithm with high flexibility in updating and developing. In addition, it has acceptable scalability in terms of occupying with increasing the number of character adopted by keyboard. This algorithm is also reliable due to the use of feedback between testing and evaluation phase with the design and implementation phases. At the other hand, the proposed algorithm has a noted advantage, which is the simplicity with high randomness. This leads to use it in real-time systems as it is light and does not have a noted delay. One of the most common systems that use real-time keyboard is the chatting applications. These applications require algorithms that are operated in fast way with a sufficient results and security level.

4. Results

In order to test and evaluate the proposed encryption algorithm, a simulator is designed using MATLAB software. The simulator follows the proposed algorithm working steps. As an initial requirement for applying the propose algorithm, different random seeds are adopted to generate indexing orders for the interleaver. These seeds are represented as polynomials that are fed to a shift-register for generating random 8-bit indexing order [2]. Table I. Listed the seed code in binary and the relevant polynomials. The polynomials are structured from 8-bit. These polynomials are generated using Cyclic Redundancy Check (CRC) for 8-bit output. The reducing of redundancy increases the efficiency of these polynomials in generating random 8-bit indexing orders for interleaver. At the other hand, the 4-bit seed code, which is the index of the chosen polynomial, has sent to the encryption side as a side information. These 4-bit is the index of 16 possible polynomials. These 4-bit is combined with the 8-bit encrypted code for each character to be 12-bit. The 4-bit is allocated at the first, while the 8-bit is followed as shown in Figure 4. The final form of 12-bit is called encrypted ASCII. It is important to note that the additional 4-bit increase the complexity, but the security for real-time encryption is guaranteed.

Different testing methods have been considered to check the randomness of the generated interleaver indexing orders. This is due to the main effect of randomness on resulting a secure encrypted ASCII code for each character in keyboard. The considered checkers for the generated indexing orders are:

- ITU tests.
- Dump Monkeys.
- Semi-Smart Monkeys.

Table II shows the results of the above tests for 300 indexing orders for each experience that adopts one seed code and related polynomial. It is notable that most of the generated interleaver indexing orders are passed through the considered tests. The experiment 10 suffers from high fail ratio and this is normal event in adopting the generation of random numbers. For testing the encryption and decryption processes, ten experiments are adopted. Each experiment represents a real-time printing sequence of character. Moreover, three ratios are considered as follows:

$$Encryption\ \ ratio = \frac{\gamma}{\sigma} \tag{2}$$

Where, γ is the number of right encrypted real-time printing characters. The symbol σ represents the total number of real-time printing characters.

$$Decryption\ \ ratio = \frac{\theta}{\sigma} \tag{3}$$

Where, θ is the number of right decrypted real-time printing characters. The efficiency ratio is the ration between the encrypted and decrypted ratios:

$$Efficiency\ \ ratio = \frac{Decryptionratio}{Encryptionratio} \tag{4}$$

Table III shows the results of the considered ratios. These ratios are calculated depending on equations (2)-(4). It is shown that the encryption process is performed in perfect way for all experiments. The decryption process is done in a satisfied way. The experiment 3, 4, and 10 reduces the decryption ratio due to adding Additive Wight Gaussian Noise (AWGN) to the encrypted ASCII code for realizing the real-time environments, in terms of transmission and surrounding circumstances.

For more explanation to the shape of character ASCII codes throughout the performance of the proposed algorithm, Table IV shows the ASCII code of original, encrypted, and decrypted codes. Each code of such a character is seen to be changed from 8-bit number in decimal representation to 12-bit numbers and returned to original 8-bit code in the decryption side. In this table, five cases are considered, while in real the algorithm covers all the characters available in keyboards with 101 keys. It is important to note that the codes ae in binary numbering, but

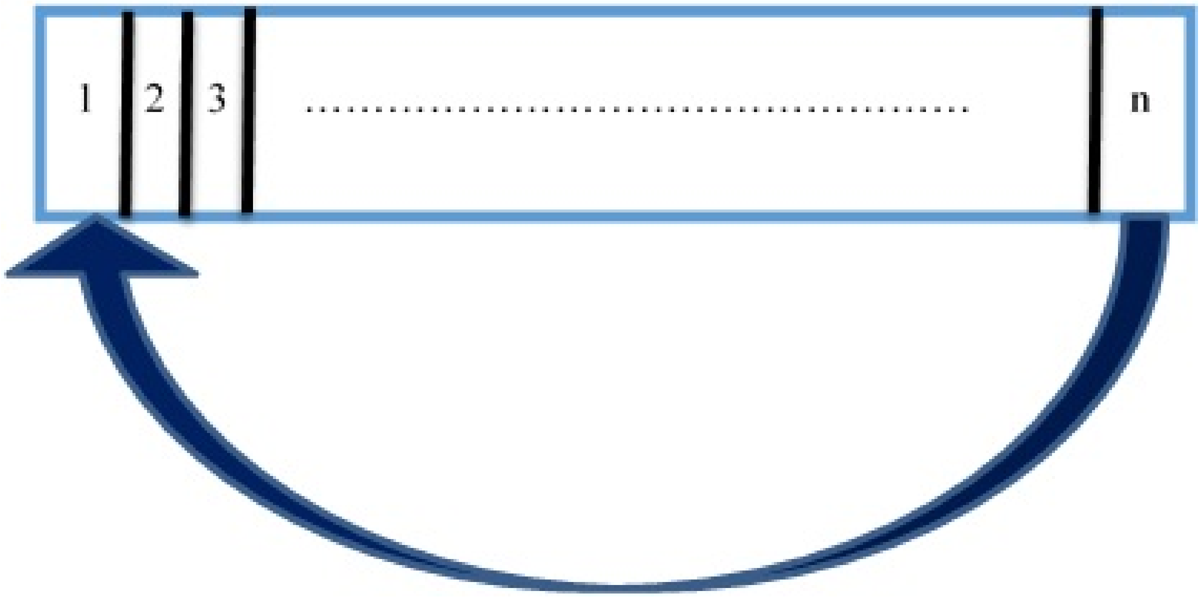

Figure 3. The right circular shift process

TABLE I. Polynomials of interleave indexing orders

No.	Seed Code	Seed Polynomial
1	0000	$X^7 + 13X^6 + 20^X + 30$
2	0001	$4X^7 + 13X^6 + 20X^3 + 5X + 3$
3	0010	$5X^5 + 10X^4$
4	0011	$X^7 + 6X^6 + 25X^4 + 30$
5	0100	$16X^7 + X^5 + 2X^2 + 3X$
6	0101	$X^7 + 10X^5 + 2X^3 + 30$
7	0110	$X^5 + X^4 + X^2 + 5$
8	0111	$X^4 + X^3 + X^2 + 30$
9	1000	$X^7 + X^6 + X^5 + X^4 + 2X^3 + X^2 + X$
10	1001	$X^7 + 30$
11	1010	$X^3 + X$
12	1011	$X + 20$
13	1100	$X^7 + 12X^3 + 2X$
14	1101	$X^6 + 30$
15	1110	$X^7 + 15X^2 + X + 3$
16	1111	$X^5 + 15$

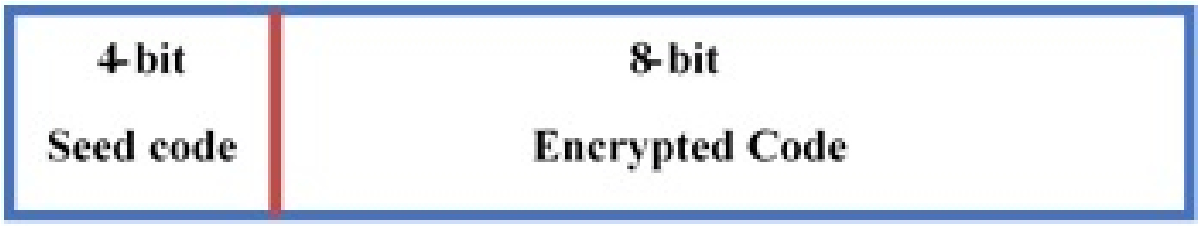

Figure 4. Final encoded ASCII

TABLE II. Random indexing order test

No.	Seed Code	Seed Polynomial
1	298	2
2	297	3
3	299	1
4	300	0
5	295	5
6	290	10
7	300	0
8	300	0
9	296	4
10	298	2
11	299	1
12	297	3
13	298	2
14	300	0
15	300	0
16	299	1

TABLE III. Testing ratios of ten experiments

Experiment No.	No. of Characters	Encryption Ratio	Decryption Ratio	Efficiency Ratio
1	1200	100%	100%	100%
2	1000	100%	100%	100%
3	2000	100%	99%	99%
4	500	100%	98%	98%
5	200	100%	100%	100%
6	300	100%	100%	100%
7	2500	100%	100%	100%
8	1500	100%	99%	99%
9	1300	100%	98%	98%
10	900	100%	95%	95%

here the decimal form is adopted to ease the representation in the limited size table.

In order to stop on the consumed time of the proposed encryption algorithm as it works in real-time, Table V expresses the run time of five experiments. These experiments are performed with different number of included characters in real-time to test the proposed algorithm behavior over these character sizes. It is shown that the consumed time is close for all experiments, run in PC with CPU of 1.2 GHz, in both sides of encryption and decryption. It is also noted that the decryption consumed time is less than related encryption due to the absence of printing speed of users on keyboards.

As a future work, the proposed keyboard encryption algorithm can be more developed in terms of hardware and software sides. These developments include the production of built-in security keyboard, and applying additional encryption techniques.

5. Conclusions

A keyboard encryption algorithm based on software engineering model was proposed. It included two parts: encryption and decryption coding. The initial ASCII code of the pressed key in a keyboard is encrypted in two steps of interleaving and XOR processes. The same steps were adopted in the decryption process, in which the same initial ASCII code is returned back. The proposed algorithm was tested over different cases. The generated random interleaving indexing order was checked using different attacking software and the ITU tests. The results were positive as most of the generated indices were passed. In terms of performance of the proposed algorithm, the algorithm is tested among the right encryption process. The results were encouraging as the efficiency ratio is varied between 98% to 100%. The process consumed time of the encryption and decryption was tested, and the results appeared that the decryption consumed less time that encryption by 5 ns. It is important to note that the proposed algorithm was light and efficient to be active in real-time applications that uses keyboard, such as chatting.

TABLE IV. Coding of characters using proposed algorithm

Character	ASCII Code (8-bit)	Encrypted Code (12-bit)	Decrypted Code (8-bit)
A	$(65)_{10}$	$(2762)_{10}$	$(65)_{10}$
B	$(66)_{10}$	$(2067)_{10}$	$(66)_{10}$
$*$	$(42)_{10}$	$(1233)_{10}$	$(42)_{10}$
a	$(97)_{10}$	$(1673)_{10}$	$(97)_{10}$
b	$(98)_{10}$	$(976)_{10}$	$(98)_{10}$

TABLE V. . Consumed time for the encryption and decryption processes

No. of Character	Encryption Time for each Character	Decryption Time for each Character
1000	$6e^{-6}$	$5.5e^{-6}$
2000	$6e^{-6}$	$5.5e^{-6}$
3000	$5.8e^{-6}$	$5.2e^{-6}$
4000	$5.85e^{-6}$	$5e^{-6}$
5000	$5.79e^{-6}$	$5e^{-6}$

References

[1] M. Caudle, *The Keystroke Killer: Transcendence Paperback*. New Orleans, 2018.

[2] D. Rudrapal and S. Das, *Keystroke Patterns: An efficient biometric for authentication*. LAMBERT, 2012.

[3] N. Jack, *The Key Logger: A Forbidden Glimpse into the True Nature of Women Kindle Edition*. Kindle Unlimitted, 2015.

[4] P. Merino, C. Kloos, M. Organero, and A. Pardo, "A software engineering model for the development of adaptation rules and its application in a hinting adaptive e-learning system," *Computer Science and Information Systems*, vol. 12, no. 1, 2012.

[5] I. Sarker, F. Faruque, U. Hossen, and A. Rahman, "Survey of software development process models in software engineering," *International Journal of Software Engineering and Its Applications*, vol. 9, no. 11, 2015.

[6] M. Kumar, "Compliance modelling and identification of 5-axis vertical articulated robort for machining applications," *International Journal of Science, Technology and Management*, vol. 6, no. 1, 2016.

[7] M. Agarwal, M. Mehra, R. Pawar, and D. Shah, "Secure authentication using dynamic virtual keyboard layout," in *International Conference and Workshop on Emerging Trends in Technology*.

[8] A. Parekh, A. Pawar, P. Munot, and P. Mantri, "Secure authentication using anti-screenshot virtual keyboard," *International Journal of Computer Science*, vol. 8, no. 3, 2011.

[9] D. Liu, E. Cuervo, V. Pistol, R. Scudellari, and L. Cox, "Screenpass: Secure password entry on touchscreen devices," in *The 11th annual international conference on Mobile systems, applications, and services*.

[10] S. Andhale and P. Rane, "Qwerty cipher an extended substitution cipher," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 5, 2015.

[11] S. Zhai and P. Kristensson, "The word-gesture keyboard: Reimagining keyboard interaction," *communications of the ACM*, vol. 55, no. 9, 2012.

[12] O. Coltell, J. Badía, and G. Torres, "Biometric identification system based in keyboard filtering," in *IEEE 33rd Annual International Carnahan Conference on Security Technology*.

[13] M. K. Shah, D. Kataria, S. Raj, and G. Priya, "Real time working of keylogger malware analysis," *International Journal of Engineering Research and Technology*, vol. 9, no. 10, 2020.

[14] S. Shinde and U. Wanaskar, "Keylogging: A malicious attack," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 6, 2016.

[15] I. Barankova, U. Mikhailova, and G. Lukyanov, "Software development and hardware means of hidden usbkeylogger devices identification," *Journal of Physics: Conference Series*, vol. 1441, 2020.

[16] P. Tuli and P. Sahu, "System monitoring and security using keylogger," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 3, 2013.

[17] K. Lakhtaria, "Protecting computer network with encryption technique: A study," *International Journal of u- and e- Service, Science and Technology*, vol. 4, no. 2, 2011.

[18] N. Jacob, "Review of various encryption algorithms," *Global Journal of Computer Science and Technology: E Network, Web and Security*, vol. 19, no. 1, 2019.

[19] S. Tayal1, N. Gupta, P. Gupta, D. Goyal, and M. Goyal, "A review paper on network security and cryptography," *Advances in Computational Sciences and Technology*, vol. 10, no. 5, 2017.

[20] S. Singla and J.Singh, "Cloud data security using authentication and encryption technique," *Global Journal of Computer Science and Technology Cloud and Distributed*, vol. 13, no. 3, 2013.

[21] M. Alanni, R. Almuttairi, and H. Alsaadi, "Un-imperceptible image stenography approach (u-iisa) in excel sheet," *International Journal*

of Computing and Digital Systems, vol. 9, no. 6, pp. 1263–1274, 2020.

[22] S. Rana and M. AbulKashem, "A modified split-radix architecture-based key scheduling technique for lightweight block ciphers," *International Journal of Computing and Digital Systems*, 2021.

[23] A. Navghane, A. Pise, and Y. Dandawate, "Machine vision based approach for automated keypad inspection," *International Journal of Computing and Digital Systems*, 2022.

[24] M. Shah, D. Kataria, S. BharathRaj, and G. Priya, "Real time working of keylogger malware analysis," *International Journal of Engineering Research Technology*, vol. 9, no. 10, pp. 569–573, 2020.

[25] G. Oligeri, S. Sciancalepore, S. Raponi, and R. DiPietro, "Broken-strokes: On the (in)security of wireless keyboards," *WiSec 20*, vol. 9, no. 10, pp. 1–11, 2020.

[26] N. Whiskerd, N. Körtge, K. Jürgens, K. Lamshöft, S. Ezen-nayaGomez, C. Vielhauer, J. Dittmann, and M. Hildebrandt, "Keystroke biometrics in the encrypted domain: a first study on search suggestion functions of web search engines," *EURASIP Journal on Information Security*, vol. 2020, no. 2, pp. 1–16, 2020.

[27] I. Sommerville, *Software Engineering*. Pearson Education, Inc, 2017.

[28] M. Kuutilaa, M. Mantylaa, U. Farooqa, and M. Claes, "Time pressure in software engineering: A systematic review," *Informayion and Software Technology*, vol. 121, 2020.

[29] L. Mottola, G. Picco, and F. Oppermann, "Simplifying the integration of wireless sensor networks into business processes," *IEEE Transaction on Software Engineering*, vol. 45, no. 6, pp. 576–596, 2019.

[30] J. Ayuba, H. Safwana, and Y. Abdulazeez, "Software development of integrated wireless sensor networks for real-time monitoring of oil and gas flow rate metering infrastructure," *Journal of Information Technology and Software Engineering*, vol. 8, no. 2, pp. 1–10, 2018.

[31] I. Almomani and A. Alromi, "Integrating software engineering processes in the development of efficient intrusion detection systems in wireless sensor networks," *Sensors*, vol. 20, no. 5, pp. 1–28, 2020.

Muayad S. Croock earned his Ph.D. (2012) in Computer Engineering at Newcastle University, School of Electrical, Electronics and Computer Engineering. He finished his M.Sc. degree in Computer Engineering from University of Technology-Iraq (2003), and a B.Sc. degree in Computer Engineering from the University of Technology (1998). Dr. Muayad is a staff member at University of Technology since 2004, where he currently holds assistant professor in Software Engineering. His research interest include Software Engineering, Security, and WSN.