A Data Partitioning and Scrambling Method to Secure Cloud Storage with Healthcare Applications

Shu-Di Bao[1,2], *IEEE Member*, Yang Lu[1], Yan-Kai Yang[1], Chun-Yan Wang[1], Meng Chen[1], Guang-Zhong Yang[2], *IEEE Fellow*
[1]School of Electronic and Information Engineering, Ningbo University of Technology, Ningbo, China
[2]The Hamlyn Center, Imperial College of London, London, United Kingdom
sdbao@ieee.org

***Abstract*— With increasing use of cloud storage for healthcare applications, potential security risks and the need for enhanced security solutions are becoming a pressing issue for clinical adoption. In this paper, a data partitioning and scrambling method at the application layer is proposed for healthcare data, where a tiny part of the original data is used to scramble the remaining data without any cryptographic key, and the former is kept locally while the latter under extra protection is sent to cloud platforms. Theoretical and experimental analyses have been carried out to demonstrate the security performance of the proposed method, which can be easily deployed in any existing communication systems as an add-on for security.**

Keywords—Healthcare; cloud; data partitioning; data scrambling

I. INTRODUCTION

Cloud-enabled pervasive healthcare service is becoming a general trend in the development of future IT infrastructure for healthcare. This is propelled by the myriad of wearable and point-of-care devices, integrated with sensing and wireless or Internet connectivity. In healthcare, the need for continuous sensing, monitoring, and intervention has motivated the research in body sensor networks [1], facilitating the development of sensor platforms that offer "ubiquitous" and "pervasive" monitoring of physical, physiological, and biochemical parameters in any environment without activity restriction. While most of today's devices use secure methods to communicate information to the cloud, they are vulnerable to malicious attacks [2-4]. To better understand this challenge, it is necessary to look into the TCP/IP five-layer reference model as well as risks and threats at different layers.

Physical layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes. In wireless communications, security solutions in the physical layer focus on exploiting the physical layer properties of the wireless channels, such as multi-path fading and interference [5-7]. Link layer defines the group of methods and communication protocols that only operate on the link between adjacent network nodes, and is responsible for media access control, flow control and error checking. Security solutions in the link layer should address attacks ranging from address resolution protocol cache poisoning for wired clients to de-authentication of wireless clients [8-11]. Network layer provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host via one or more networks. Security solutions in the network layer normally focus on encryption of IP datagrams and source authentication, such as IPsec [11-12]. Transport layer provides host-to-host communication services, including reliability, flow control and multiplexing. It has received significant attention in terms of potential security risks. For example, the well-known Transport Layer Security (TLS) is a protocol that ensures privacy between communicating applications and their users on the Internet [13]. Application layer contains communication protocols and interface methods used in process-to-process communications across an IP network. According to a recent survey, 80% of networking and security proposals are concerned about application layer gateways. Unfortunately, it is also reported that 80% of companies that host web applications have been attacked in 2010. As most of existing security proposals at the application layer are firewall related, further investigation of other aspects are expected.

For Cloud-enabled pervasive healthcare services, much attention is directed to protecting confidential healthcare data or medical records, as malicious attacks may cause disclosure or tampering of transmitted data, thus making personal privacy, health or even safety at risk [14-15]. It is also well known that patients who trust their health information will be kept private and secure will be more willing to accept wireless wearable sensors or Internet enabled healthcare services.

In this study, we focus on an add-on security mechanism at the application layer that can significantly improve security level without compromising real-time performance for healthcare cloud applications, and propose a data partitioning and scrambling method which uses a tiny part of original data to scramble the remaining data. The former is kept locally while the latter under extra protection is sent to cloud platforms. The proposed method is justified with theoretical and experimental analysis in terms of security performance.

The remainder of this paper is organized as follows. In Section II, the background of application-layer security and existing work related to data scrambling are briefly introduced. In Section III, the data partitioning and scrambling method is detailed, followed by theoretical and experimental analysis in Section IV. A conclusion is finally given in Section V.

II. BACKGROUND

Application-layer security normally refers to methods of protecting web applications at the application layer from malicious attacks that may expose private information. Current strategies are focused on the following aspects: securing and controlling data at its source, applying

This work was supported in part by Zhejiang Provincial Natural Science Foundation of China (No. LY14F010005), National Natural Science Foundation of China (No. 61102087) and Natural Science Foundation of Ningbo City (No. 2014A610074).

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

application layer controls across the network and at the gateways, and enforcing controls at the endpoints, among which the securing and controlling data at its source is of most importance [16].

Data scrambler is a practical approach, which usually refers to an algorithm that converts an input string into a seemingly random output string of the same length, thus avoiding long sequences of bits of the same value. It was originally proposed for the physical layer in the analog domain in order to make extraction of the original signal difficult. There are two typical scramblers, i.e., additive scramblers and multiplicative scramblers. The former transform the input data stream by applying a pseudo-random binary sequence by modulo-two addition, while the latter perform a multiplication of the input signal by the scrambler's transfer function in Z-space.

In current cryptography systems, scrambling is an important idea for non-secret encryption which ultimately led to the invention of both the RSA encryption algorithm and Diffie-Hellman key exchange scheme. In this context, data scrambling is the process to obfuscate or remove sensitive data [17], which is considered as one of the important security add-ons for application-oriented networking cases.

III. THE PROPOSED METHOD

A. Method Overview

Built upon the existing security mechanisms that are standardized in the communication protocols, such as IEEE 802.15.1 (Bluetooth) and 802.15.4 (Zigbee), the proposed security mechanism will be deployed at the application layer of the TCP/IP five-layer model. Its application scenario is depicted in Fig. 1. A smart phone or pad could be a gateway that connects a biosensor or even body sensor network to remote cloud servers. The biosensor or body sensor network is the front end for capturing physiological signals, which will be sent to the gateway for further processing and transfer. Before transferring the data to the Internet, the original health data is adequately scrambled into random data to enhance security at the gateway.

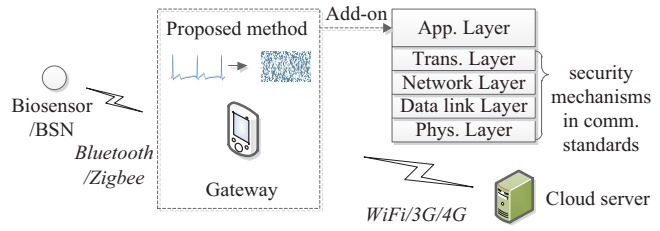

Fig. 1. Application scenario of the proposed method

The overall method of encryption is depicted in Fig. 2, where the first data block of original health data, denoted as A_0, is locally stored while the remaining data is partitioned and scrambled consecutively for extra security before being sent out for cloud storage. The partitioning and scrambling process should be repeated enough times to eliminate statistical properties of original data.

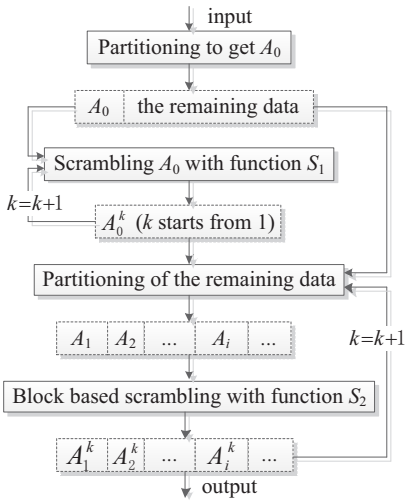

Fig. 2. Overall method of encryption (the parameter k means the k^{th} round of partitioning and scrambling)

Firstly, the size of the first block, denoted as l_0, is randomly selected between $[L_{\min}, L_{\max}]$, where $L_{\min}$ and $L_{\max}$ are the minimum and maximum size in bytes of data blocks, respectively. The values of $L_{\min}$ and $L_{\max}$ should be configured in advance, conforming to the requirements given in the subsection III.C.

Secondly, the first block is scrambled with a function, denoted as S_1, as depicted in Fig. 3, where a_i represents one single byte of A_0, and $\oplus$ represents the operation of exclusive OR. Prior to the exclusive OR operation, a bitwise reversal should be done to the foregoing byte if it has an odd number of Hamming weight. For example, if we assume the binary formats of a_0 and a_n are 10100001 and 10100101, respectively, a_n needs to be bitwise reversed to 01011110 as its Hamming weight is 3, before an exclusive OR operation is carried out to produce a_0^1. The purpose of this reversal operation is to expand the range of data changes, and make the method flexible for any other kind of data as well. The difference analysis between A_0 and A_0^1 will be given in Section IV.

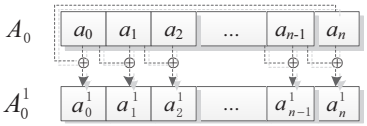

Fig. 3. Scrambling function S_1 with A_0

Thirdly, the remaining data is partitioned with varying sizes calculated as follows.

$$\begin{cases} l_1 = w(A_0^1) \bmod (L_{\max} - L_{\min} + 1) + L_{\min} \\ l_i = w(A_{i-1}) \bmod (L_{\max} - L_{\min} + 1) + L_{\min} \end{cases} \quad (1)$$

where $i \geq 2$, l_i is the size of A_i in bytes, and $w(\cdot)$ means the Hamming weight of a binary sequence. The partitioning process ensures that the data is partitioned in a random way so that block sizes are varying randomly and cannot be easily

guessed without knowing the size of A_0^1. More importantly, there is no need to keep the size information locally as it can be derived from A_0.

Fourthly, the partitioned data blocks are scrambled with a function, denoted as S_2, as depicted in Fig. 4, where A_0^1 is used to scramble A_1 while each of other data blocks is scrambled by its foregoing block as follows.

$$\begin{cases} A_1^1 = A_0^1 \oplus A_1 \\ A_j^1 = A_{j-1} \oplus A_j \end{cases} \quad (2)$$

Similar to the scrambling function S_1, a reversal operation of bitwise should be done earlier on the foregoing data block if it has an odd number of Hamming weight. For example, if we assume that the binary format of A_1 is 0100110011000101, before scrambling A_2, A_1 should be bitwise reversed because its Hamming weight is 7. The purpose of this reversal operation is to expand the range of data changes, especially for cases where two consecutive data blocks are with quite similar value ranges. The reversal operation meets the reversible requirement for decryption.

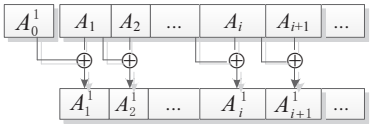

Fig. 4. Scrambling function S_2 with the remaining data

Considering that two neighboring blocks may have different sizes, during the exclusive OR operation the foregoing block should be used cyclically to match the size of the current block, if it has less bytes. Let data blocks be denoted as $A_i = \{a_{k|i} \mid 0 \le k < l_i\}$, where k is an integer and $a_{k|i}$ means the k^{th} byte of A_i. The exclusive OR operation can be further detailed as follows.

$$A_i^1 = \{a_{k|i} \oplus a_{k|i-1} \mid 0 \le k < l_i\}$$

while $l_{i-1} \ge l_i$, or

$$A_i^1 = \{a_{k|i} \oplus a_{k|i-1} \mid 0 \le k < l_{i-1}\} \\ || \{a_{k'|i} \oplus a_{(k'-l_{i-1})|i-1} \mid l_{i-1} \le k' < l_i\}$$

while $l_{i-1} < l_i$, where $1 \le i \le N$ and $||$ is a concatenation operation. It should be noted that A_1 is scrambled by A_0^1 to produce A_1^1 as a special case.

Finally, a number of rounds shall be carried out starting from the 2nd step, i.e., scrambling of the very first data block A_0, to completely eliminate the statistical characteristics of original data. The final result after the n^{th}-round of partitioning and scrambling can be denoted as $\{A_i^n \mid i = 1,2,\cdots\}$, which is sent for cloud storage while the original first data block A_0 is kept locally.

B. Decryption Process

The decryption of $\{A_i^n \mid i = 1,2,\cdots\}$ is an inverse process of the encryption, including partitioning and de-scrambling as depicted in Fig. 5. If there are a number of n rounds during the encryption process, the same number of rounds should be carried out for the decryption process. Right before the round of partitioning and de-scrambling, the scrambled results of A_0 for k rounds should be firstly calculated, denoted as $A_0^k (1 \le k \le n)$, which will be used to calculate the first block size of the input data and also de-scramble it. For all the other blocks, it is always the case that a foregoing data block is used to calculate the size of the current block and its de-scrambled result will be used to de-scramble the current block.

For example during the 1st round of decryption, A_0^n is used to calculate the first block size of the input data. Once the first data block is determined, denoted as A_1^n, the result of de-scrambling can be obtained as follows.

$$\begin{cases} A_1^{k-1} = A_0^k \oplus A_1^k \\ A_i^{k-1} = A_{i-1}^{k-1} \oplus A_i^k \end{cases} \quad (3)$$

where the exclusive OR operation is the same as that during the scrambling process. Once A_1^{n-1} is obtained, it is used to calculate the size of A_2^n and then A_2^{n-1} can be obtained by applying the Eqn. 3, i.e., $A_2^{n-1} = A_1^{n-1} \oplus A_2^n$.

The partitioning and de-scrambling processes detailed above should be carried out n times, till the original data is recovered. It is easy to check if the recovered data is the same as the original data by applying a message authentication code.

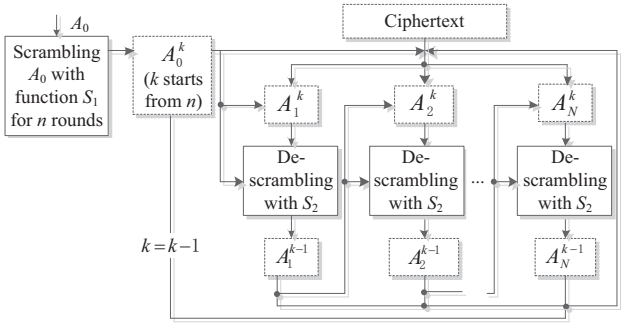

Fig. 5. Decryption process

C. Selection of Parameters

In the proposed method, only two parameters need to be configured in advance, i.e., $L_{\min}$ and $L_{\max}$ which are the minimum and maximum lengths in bytes of data blocks, respectively. There is a tradeoff in setting a value for $L_{\min}$ between the required security level and the local database size. A larger value of $L_{\min}$ leads to a higher security level as it will result in an increased difficulty for an unauthorized party to find out the exact information of A_0. However, the local storage space will also be expanded due to a bigger size of A_0 for each record of health data. To offer an acceptable security level, it is suggested to set $L_{\min}$ as 32, i.e., the minimum length of data blocks is 32, for physiological data.

Another condition for selecting $L_{\min}$ and $L_{\max}$ is that the sizes of data blocks should vary evenly between $L_{\min}$ and $L_{\max}$

in a statistical sense, i.e., $w(A_{i-1}) \bmod (L_{\max} - L_{\min} + 1)$ in Eqn. 1 should vary evenly between $[0, L_{\max} - L_{\min}]$. Since the size in bytes of A_0 is between $[L_{\min}, L_{\max}]$, the averaged Hamming weight of A_0 in a statistical sense is

$$\overline{w(A_0)} = \frac{(L_{\max} + L_{\min})}{2} \times 4 = 2(L_{\max} + L_{\min}) \quad (4)$$

Therefore, the statistically averaged result of the modulo operation in Eqn. 1 is

$$\begin{aligned} &\overline{w(A_0) \bmod (L_{\max} - L_{\min} + 1)} \\ &= 2(L_{\max} + L_{\min}) \bmod (L_{\max} - L_{\min} + 1) \\ &= (4L_{\min} - 2) \bmod (L_{\max} - L_{\min} + 1) \end{aligned} \quad (5)$$

In order to get a statistically even distribution, it is required that $4L_{\min} - 2 >= L_{\max} - L_{\min} + 1$, i.e.,

$$L_{\max} \le 5L_{\min} - 3 \quad (6)$$

IV. Performance Analysis

A. Security Performance

The main idea of the proposed method is to randomly partition the data and make use of the first data block A_0 to scramble the remaining data which will be sent for cloud storage. To eavesdrop, an attacker has to try to decrypt the cipher either by guessing the first data block or by attacking directly on the cipher with potential statistical characteristics as he or she may know what kind of health data it is, e.g., electrocardiogram (ECG) or photoplethysmogram (PPG).

The process of scrambling A_0 before it is used for subsequent encryption eliminates those attacks using statistical characteristics which usually can significantly decrease the difficulty of attacks. Because of the randomness characteristics, a brute-force attack on A_0^k will averagely take $2^{(l_0 \times 8 - 1)}$ times of search. For example, if l_0 is set to 32 which is a suggested value for $L_{\min}$ as described in the subsection III.C, the average number of attacks will be 2^{255}. On the other hand, it is even more difficult to attack directly on the cipher because statistical characteristics are eliminated due to the unique encryption process and more importantly, the size of each data block is randomly varying, which greatly increases the difficulty of attacks. An experimental analysis on the randomness of A_0^k and scrambled data will be detailed later. It is noted that security mechanisms against other attacks including man-in-the-middle attacks and denial-of-service attacks should be further deployed, which however is not the main concern of this study.

B. Experimental Analysis

To demonstrate the practical value of the proposed technique, a detailed analysis with MIT-BIH database and self-built databases is performed. Specifically, ECG signals from MIT-BIH arrhythmia database and ECG signals self-collected from healthy subjects are used as health data. All data were converted into binary format based on a specific quantization resolution.

Fig. 6 shows the randomness performance of A_0^1 which is self-scrambled from A_0 for one round. As can be seen from the figure, after scrambling the statistical characteristics of A_0 have been eliminated. Actually, if more rounds of scrambling were carried out, a perfect randomness performance of A_0^k can be achieved. Using A_0^k while not A_0 to scramble the remaining data blocks can obtain an enhanced security level, and make the proposed method applicable to any kind of data as well.

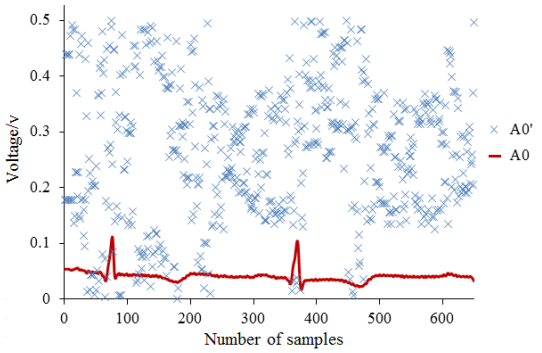

Fig. 6. Comparison of A_0 and its one-round scrambled version

Examples of scrambling results with a different number of rounds are shown in Fig. 7, where Fig. 7a and 7b are the results without/with scrambling of A_0, respectively. It can be seen that by using the scrambled A_0 better results can be obtained in terms of randomness performance, and with more rounds of scrambling an even better performance of randomness can be achieved.

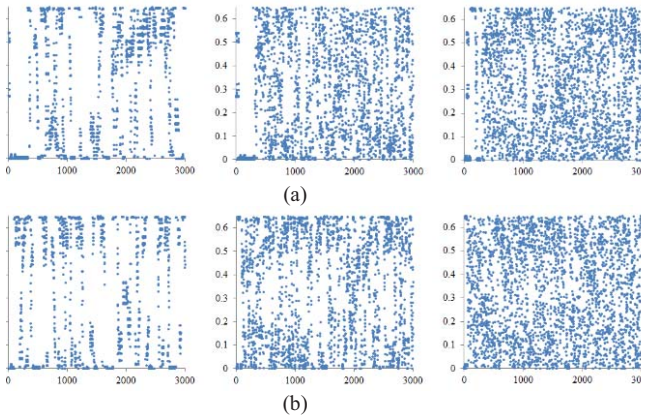

Fig. 7. Examples of scrambling results (a) using A_0; (b) using scrambled verion of A_0 (from left to right: 1st-round, 2nd-round, and 3rd-round)

With the two databases, a number of 5000 binary sequences after the rounds of partitioning and scrambling processes are randomly selected for randomness evaluation using NIST standardized tests. The numeric results are given in Table I. It can be seen from the table that with more rounds of data scrambling a better randomness performance can be achieved. After the 3rd round of scrambling, the outputs have an acceptable level of randomness.

Table I. Numeric rsults of randomness test using NIST tests

Database	Num of rnds	Percentage of passing the test	
		Cusum test	Freq. test
MIT-BIH	1	99.04%	46.90%
	2	99.76%	78.56%
	3	99.96%	94.02%
	4	99.84%	98.74%
	5	99.92%	98.94%
Self-built ECG database	1	99.86%	54.46%
	2	99.92%	81.34%
	3	99.92%	94.98%
	4	100%	98.08%
	5	99.94%	98.90%

C. *Example with Image Data*

To check the applicability to other types of data, image files are selected as the input to the encryption process. An example both in BMP and JPG formats is shown in Fig. 8, where the sizes of A_0 for BMP and JPG images are set to 128 and 64 in bytes, respectively. Only the pixel data is encrypted while the file header remains as plain text. Both scrambled results of A_0 and the remaining data are connected together for image display. Theoretically, a larger size of A_0 can result in a better scrambling result because of more information involved for the subsequent scrambling. However, it can be seen from the figure that the results with JPG format are much better than that with BMP format. The reason is because there are many long continuous 1's or 0's in BMP format which make the scrambling process less effective, compared to the case with JPG format. Therefore, it is suggested to have original data compressed before applying the proposed security mechanism, which is common practice in any case.

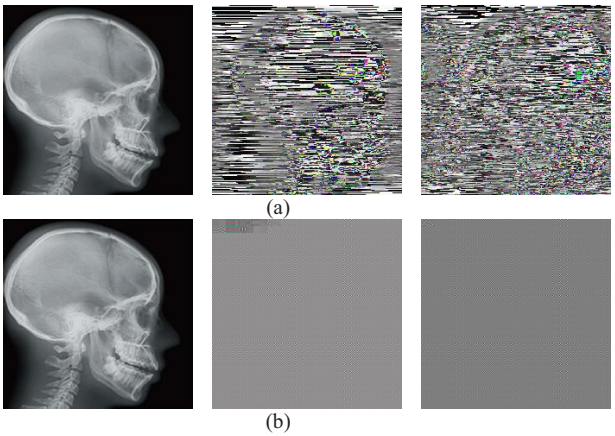

Fig. 8. Examples of scrambling results with image data (a) BMP format; (b) JPG format (from left to right: original, 1st-round, and 2nd-round)

I. CONCLUSION

In this paper, a data partitioning and scrambling method at the application layer has been proposed for health data, where a tiny part of data is used to partition and scramble the remaining data and the former is kept locally while the latter under extra protection is sent for cloud storage. Both theoretical and experimental analyses have been carried out to demonstrate the security performance of the method. As a security add-on at the application layer, it can be easily embedded into existing communication systems. Furthermore, the proposed method can also be used flexibly with any kind of data that require strengthened security protection as well as real time processing. In future, studies on the management of local database will be carried out to make the method suitable for practical and easy use.

REFERENCES

[1] G.Z. Yang, "Body sensor networks", Second Edition, Springer, 2014, ISBN 978-1-4471-6374-9.

[2] S. Subashini, V. Kavitha, "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer Applications, 2011, 34(1): 1–11.

[3] R.H. Weber, "Internet of things – new security and privacy challenges," Computer Law & Security Review, 2010, 26(1): 23–30.

[4] A.J. Jara, M.A. Zamora, A.F.G. Skarmeta, "An architecture based on internet of things to support mobility and security in medical environments," 7th IEEE Consumer Communications and Networking Conference, 2010, pp: 1–5.

[5] M. Bloch, J. Barros, "Physical-layer security: from information theory to security engineering," Cambridge University Press, 2011, pp. 1–5.

[6] Y.S. Shiu, S.Y. Chang, H.C Wu, et al., "Physical layer security in wireless networks: a tutorial," IEEE Wireless Communications, 2011, 18(2): 66–74.

[7] L. Dong, Z. Han, A.P. Petropulu, et al., "Improving wireless physical layer security via cooperating relays," IEEE Transactions on Signal Processing, 2010, 58(3): 1875–1888.

[8] A. Rajaram, S. Palaniswami, "The trust-based MAC-layer security protocol for mobile ad hoc networks," 6th International Conference on Wireless Communications Networking and Mobile Computing, 2010, Chengdu, China, pp. 1–4.

[9] B. Huang, "A survey WLAN security defenses based on the link layer," International Journal of Engineering and Technology, 2012, 4(4): 418–420.

[10] H. Modares, R. Salleh, A. Moravejosharieh, "Overview of security issues in wireless sensor networks," 3rd International Conference on Computational Intelligence, Modelling and Simulation, 2011, Langkawi, pp. 308–311.

[11] H. Seo, H. Kim, "Network and data link layer security for DASH7," Journal of Information and Communication Convergence Engineering, 2012, 10(3): 248–252.

[12] J. Granjal, E. Monteiro, J. Sa Silva, "Enabling network-layer security on IPv6 wireless sensor networks," IEEE Global Telecommunications Conference, 2010, Miami, FL, pp. 1–6.

[13] T.S. Gopal, R. Jain, P.R.L. Eswari, et al., "Securing UDT protocol: experiences in integrating transport layer security solutions with UDT, " 3rd International Conference on Innovative Computing Technology, 2013, London, pp. 87–91.

[14] S.D. Bao, C.C.Y. Poon, Y.T.Zhang, and L.F. Shen, "Using the Timing Information of Heartbeats as an Entity Identifier to Secure Body Sensor Network," IEEE Transactions on Information Techonology in Biomedicine, 2008, 12(6): 772–779.

[15] L.M.R. Tarouco, L.M. Bertholdo, L.Z. Granville, et al., "Internet of things in healthcare: Interoperatibility and security issues," IEEE Conference on Communications, 2012, Ottawa, ON, pp. 6121–6125.

[16] J. Granjal, E. Monteiro, J.S. Silva, "Application-layer security for WoT: extending CoAP to support end-to-end message security for internet-integrated sensing applications," Lecture notes in Computer Science, 2013, vol. 7889, pp. 140–153.

[17] G.D. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," Pattern Recognition Letters, 2010, 31(5): 347–354.