ISSN: 0975-766X
CODEN: IJPTFI
Available Online through ***Research Article***
www.ijptonline.com

DATA SECURITY FOR IOT HEALTHCARE DEVICES USINGX-REA ALGORITHM

M.Nirmala, Nivash J P,Swati Chadha, Desai Priyal Hemant
School of Information Technology and Engineering, VIT University, Vellore, Tamilnadu.
Email: nirmaladhinesh@gmail.com

Received on 25-10-2016 *Accepted on 02-11-2016*

Abstract

Providing security to the data collected through the mobile/contextual sensors is necessary. In the concept of Internet of Things (IoT), we discuss one new Cryptography Algorithm X-REA for the Healthcare Devices. This article concentrates on the particular application of the IoT using AAL – Ambient Assisted Living in the field of medical by giving inconspicuous backing to frail and elderly individuals for their day by day life taking into account their environment and circumstance. IoT can meet this requirement by connecting elderly and fragile individuals to the corresponding group of caregivers. For IoT-based frameworks to achieve their maximum capacity, sound answers need to be given to the essential security questions emerged, especially those on thesafety of data being stored at backend side. This issue is solved by proposed X-REA algorithm which uses multi-key security concept to provide a high level of security.

Keywords: Internet of Things, Healthcare Devices, Ambient Assisted Living, Wearable sensors, X-REA cryptographic algorithm.

1. Introduction

With the improvement of society and the advancement of humanity, individuals perceive that well-being is just not one of the objectives which social improvement seeks after, additionally the essential condition of advancing the monetary advancement. In context to our research, we aim to discuss security to the data collected from the i-health care devices which are sensors. These sensors can be of two types. The mobile sensors and the contextual sensors continuously and periodically sense data. Like pulse rate, room temp or air quality of the patients. This data is sent through wireless interfaces toan IoT gateway that has computational and memory management systems. Easy access and pay per use methodology have made many enterprises and health sectors to use cloud services [1],[2]. Even though many predefined architectures are available to suit the need of the users, security remains the biggest issue in the cloud computing services [3],[4]. So before sending the data to a cloud infrastructure[5],we aim to provide maximum security for the encryption/decryption technique using algorithm X-REA. Encryption in databases is required not only for the safety but also should take into account the performance. X-REA algorithm provides the maximum safety and limits the time needed for encryption and decryption when the database is being queried.

The X-REA algorithm limits the encryption time and provides higher security because we are using three different alphanumeric random keys and series of rounds to secure the given input. The database values will be guaranteed by converting into ASCII values followed by grouping into 32 bits of blocks. Each block will undergo XOR operation with a random alphanumeric key followed by two more XOR operations with new alphanumeric random keys.In comparison with REA which only adds the key to plaintext while anencryption, X-REA the key undergoes XOR operation with a 32-bit block of data. This will increase the robustness of the encrypted data against intruders. In comparison with the traditional DES algorithm which uses 16 rounds of processing, the proposed algorithm includes XOR operation with three unique alphanumeric keys and reverse operation on the binary code which is not being provided by any current algorithm.

2. Preliminaries

To understand the perspectives about the performances of various algorithms we will discuss the knowledge gathered from other sources in this section. After performance evaluation on both hardware and software platforms, NSIT (National Institute of Standards and Technology) replaced the old Data Encryption Standard (DES) by selecting Rijndael [6] and considering it as the new Advanced Encryption Standard (AES) in October 2000.

AES was concluded faster and more efficient as compared to other encryption algorithms in the results in [7]. For transmission of data most resource consumption is for transmission of data rather than computation on data. While considering the transmission of data, there is very less difference in the performance of different symmetric key schemes. It is better to use AESif at one end the encrypted data is stored and will be decrypted multiple times even for the case of data transfer.

In another study, popular algorithms were not only implemented but also their performances were compared. Various files of varying contents and sizes were encrypted. This study [8] was done on various secret key algorithms like AES, DES, 3DES, and Blowfish. This was followed by testing of these algorithms on two different hardware platforms. The testing on various platforms was done for comparison of their performances. PII 266 MHz and P4 2.4 GHz were the two different machines on which they were conducted.

The results proved that Blowfish had magnificent performance when compared to other algorithms and AES had a better performance than 3DES and DES. It was also shown that 3DES has almost 1/3 throughput of DES algorithm. As proved by experimental results an encryption algorithm, named "Reverse Encryption Algorithm (REA)" ensured better security as well as performance for most widely used software by reducing the added time cost for encryption and decryption. It also improves the time taken for execution of the query over encrypted database. To evaluate query processing performance, the study in the paper [9] observes a method to evaluate query processing performance on a database which is encrypted with the proposed algorithm (REA) and Advanced Encryption Standard. The performance is measuredregarding query execution time. The paper emphasizes on the advantages that this algorithm has over another encryption algorithm AES i.e. execution time taken for the query. The proposed algorithm is proved to improve the performance by reducing the time cost of encryption/decryption.

3. X-REA Algorithm

We recommend theEncryption Algorithm (X-REA) which is fast, straightforward and efficient. X-REA is limiting the cost added in time in encryption and description to provide the security with least degradation to the database system.

Encryption Algorithm for X- REA

Step1: Input the text (Plain Text).

Step 2: Take each character of the plain text and convert it to the corresponding ASCII value.

Step 3: Convert the ASCII code of each character into binary data.

Step 4: Group the binary data into 32 bits of blocks.

Step 5: Reverse each 32-bit block.

Step 6: Perform XOR operation on each 32 bits of ablock of data with a random alphanumeric key.

Step 7: Reverse each block of output obtained.

Step 8: Perform XOR operation on the previous 32 bits block of data with the 2nd random alphanumeric key.

Step 9: Perform XOR operation on the previous 32 bits block of data with the 3rd random alphanumeric key.

Step 11: Return the encrypted text.

Algorithm 1: X_REA_Encryption

INPUT: Plaintext (StrValue), Key[i] (StrKey[i]).

OUTPUT: Ciphertext (EncryptedData).

1. Convert the Text(StrFullVlaue) to ascii code (hexdata).

```
2.      Convert the ascii code (hexdata) to Binary data(StrBinaryData).
3.  Foreach (4 byte b in StrBinaryData).
a. if (StrBinaryData.Length>= 0 and StrBinaryData.Length<= 32).
            For ( j from 32 to StrBinaryData.Length  , 32- StrBinaryData.Length  )
{ StrBinaryData = "0" + StrBinaryData}
        b. StrEncrypt += StrBinaryData. (where, StrEncrypt= ””)
4.  Reverse the Previous Binary Data (StrEncrypt).
5. Foreach (4 byte b in StrEncrypt).
a.  XOR (strKey1,b)
b.  Reverse the previous binary data(StrEncrypt).
c.  XOR (strKey2,b)
        d. XOR (strKey3,b).
6: Return (EncryptedData).
```

Decryption Algorithm for X- REA

Step 1: Input the encrypted text

Step 2: Perform XOR operation on the encrypted 32 bits of theblock with the 3rd alphanumeric key.

Step 3: Perform XOR operation on the previous output of the 32 bits blocks with the 2nd alphanumeric key.

Step 4: Reverse the previous 32 bits blocks.

Step 5: Perform XOR operation on the previous output of the 32 bits blocks with the 1st alphanumeric key.

Step 6: Reverse the previous 32 bits blocks.

Step 7: Convert Binary data to ASCII values.

Step 8: Covert ASCII to plaintext.

Step 9: Return Decrypted data.

Algorithm 2: X- REA_Decryption

INPUT: Ciphertext (EncryptedData), R_Key[i] (StrKey[i]).

OUTPUT: Plaintext (DecryptedData).

```
    1. Foreach (4-byte b in StrEncrypt).
a.  XOR (strKey3,b)
b.  XOR (strKey2, b)
```

c. Reverse the previous binary data(StrEncrypt)

d. XOR (strKey1, b).

2. Reverse the Previous Binary Data (StrEncrypt).
3. Convert the Binary data (StrBinaryData) to ascii code (hexdata).
4. Convert the ascii code (hexdata) to Text(StrFullVlaue).
5. Return (DecryptedData).

4. Results and Descriptions

The algorithm is animprovement of the existing REA algorithm and DES algorithm. So by taking one text example, we will show that how the proposed algorithm is secure and efficient.

The explanation is provided below:

The Plain Text: AbCd@29!

Three keys are: key1 – 52P9 (00000101 00000010 01010000 00001001)

Key2 – 68S3 (00000110 00001000 01010011 00000011)

Key3 – 2314(00000010 00000011 00000001 00000100)

Encipherment:

Step 1: Input the text AbCd@29!

Step 2: Take each character of the plain text and convert it to the corresponding

ASCII value.

A-65, b-98, C-67, d-100, @-64, 2-50, 9-57, !-33

Step 3: Convert the ASCII code of each character into binary data.

65 – 01000001

98 – 01100010

67 – 01000011

100 – 01100100

64 – 01000000

50 – 00110010

57 – 00111001

33 – 00100001

Step 4: Group the binary data into 32 bits of blocks.

Block1- 01000001011000100100001101100100

Block2-01000000001100100011100100100001

Step 5: Reverse each 32-bit block.

Block1 – 00100110110000100100011010000010

Block2 – 10000100100111000100110000000010

Step 6:

Perform XOR operation on each 32 bits of ablock of data with a random alphanumeric key.

Block1 ⊕ key1

B1: 00100110110000100100011010000010

⊕

K1: 00000101000000100101000000001001

Results – 11011100001111111101001011110100

Block2 ⊕ key1

B2: 10000100100111000100110000000010

⊕

K1: 00000101000000100101000000001001

Results – 01111110011000011110001111110100

Step 7: Reverse each block of output obtained.

B1: 00101111010010111111110000111011

B2: 00101111110001111000011001111110

Step 8:

Perform XOR operation on the previous 32 bits block of data with the 2nd random alphanumeric key.

B1: 00101111010010111111110000111011

⊕

K2: 000000110000010000101001100000011

Results – 11010111011000000101000011000111

B2: 00101111110001111000011001111110

⊕

K2: 00000110000010000101001100000011

Results - 11010110001100000010101010000010

Step 9: Perform XOR operation on the previous 32 bits block of data with the 3rd random alphanumeric key.

B1: 11010111011000000101000011000111

$\oplus$

K3: 00101010100111001010111000111100

Results – 00101010100111001010111000111100

B2: 11010110001100000010101010000010

$\oplus$

K3: 00000010000000110000000100000100

Results – 00101011110011001101010001111001

Step 11: Return the encrypted text.

Encrypted Text: “*16517430+264212y”

Decipherment:

Step 1: Input the encrypted text

*16517430+264212y (0010101010011100101011100011110000101011110011001101010001111001)

Step 2: Perform XOR operation on the encrypted 32 bits of theblock with the 3rd alphanumeric key.

B1: 00101010100111001010111000111100

$\oplus$

K3: 00000010000000110000001100000100

Results – 11010111011000000101001011000111

B2: 00101011110011001101010001111001

$\oplus$

K3: 00000010000000110000001100000100

Results – 11010110001100000010100010000010

Step 3:

Perform XOR operation on the previous output of the 32 bits blocks with the 2nd alphanumeric key.

B1: 11010111011000000101001011000111

$\oplus$

K2: 00000110000010000101001100000011

Results – 00101110100101111111111000111011

B2: 11010110001100000010100010000010

$\oplus$

K2: 00000110000010000101001100000011

Results – 00101111110001111000010001111110

Step 4:

Reverse the previous 32 bits blocks.

B1: 11011100011111111110100101110100

B2: 01111110001000011110001111110100

Step 5:

Perform XOR operation on the previous output of the 32 bits blocks with the 1st alphanumeric key.

B1: 11011100011111111110100101110100

$\oplus$

K1: 00000101000000100101000000001001

Results – 00100110100000100100011010000010

B2: 01111110001000011110001111110100

$\oplus$

K1: 00000101000000100101000000001001

Results – 10000100110111000100110000000010

Step 6: Reverse the previous 32 bits blocks.

B1: 01000001011000100100001101100100

B2: 01000000001100100011100100100001

Step 7: Convert Binary data to ASCII values.

B1: 65 98 67 100

B2: 64 50 57 33

Step 8: Covert ASCII to plaintext.

65 – A

98 – b

67 – C

100 – d

64 - @

50 – 2

57 – 9

33 -!

Step 9: Return Decrypted data.

Decrypted data: “AbCd@29!”

5. Conclusion

In this paper, we have discussed a new algorithm for encryption and decryption for protecting it from being misused by intruders (unauthenticated person). In this algorithm, we have made improvement by using thereverse functionality of REA and functionality of DES algorithm. In this algorithm, we have used a new concept by taking three unique key for encryption and decryption to boost the security of algorithm. This algorithm is efficient for many applications (software systems) because it is compact and time effective and provides ahigh level of security.

References

1. Krishna, P. V. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing.2013;13(5): 2292-2303.
2. Raj E.D, Babu L.D, Ariwa E, Nirmala M, Krishna,P.V. Forecasting the Trends in Cloud Computing and its Impact on Future IT Business. Green Technology Applications for Enterprise and Academic Innovation; 2014.p.14.
3. Babu L. D, & Krishna P. V. An execution environment oriented approach for scheduling dependent tasks of cloud computing workflows. International Journal of Cloud Computing. 2014; 3(2): 209-224.
4. DhineshBabu L. D, Gunasekaran A, Krishna, P. V. A decision-based pre-emptive fair scheduling strategy to process cloud computing work-flows for sustainable enterprise management. International Journal of Business Information Systems. 2014; 16(4): 409-430.
5. Daemen J, Rijmen V. The block cipher Rijndael. In International Conference on Smart Card Research and Advanced Applications, Springer Berlin Heidelberg; 1998, p. 277-284.
6. El Fishawy N. F, Zaid O. M. A. Quality of Encryption Measurement of Bitmap Images with RC6, MRC6, and Rijndael Block Cipher Algorithms, IJ Network Security.2007;5(3): 241-251.
7. Salama D, Kader H. A, Hadhoud M. Studying the Effects of Most Common Encryption Algorithms. International Arab Journal of e-Technology.2011;2(1):1-10.
8. Franklin M. J, Jónsson B. T, Kossmann D. Performance tradeoffs for client-server query processing. In ACM SIGMOD Record.1996;25(2):149-160.