Improvement of Data Exchange Security on HTTP using Client-side Encryption

Zuhar Musliyana\textsuperscript{1,a}, Mahendar Dwipayana\textsuperscript{1,b}, Ayu Helinda\textsuperscript{1,c} and Zahrul Maizi\textsuperscript{2,d}

\textsuperscript{1}Department of Computer Science, Faculty of Computer Science, Ubudiyah Indonesia University, Jalan Alue Naga, Desa Tibang, Banda Aceh, Indonesia.
\textsuperscript{2}Department of Informatics Engineering, Faculty of Computer Science, Ubudiyah Indonesia University, Jalan Alue Naga, Desa Tibang, Banda Aceh, Indonesia.

\textsuperscript{a}zuhar@uui.ac.id, \textsuperscript{b}mahendar@uui.ac.id, \textsuperscript{c}ayu@uui.ac.id, \textsuperscript{d}zahrulmaizi@uui.ac.id

Abstract. Hypertext Transfer Protocol (HTTP) is a standard data exchange protocol over the web. Currently, HTTP is still widely used to handle communication between client and server. However the related researches show that data exchange via HTTP is still vulnerable because the data sent is in the form of plaintext. To overcome the problem, the available solutions are using Hypertext Transfer Protocol Secure (HTTPS) and Secure Hypertext Transfer Protocol (SHTTP). However, some results of related researches show HTTPS and SHTTP still have some weaknesses. Based on these problems, this study proposes a method of securing HTTP using Client-side Encryption. In this research, the encryption algorithm used is Advanced Encryption Standard (AES) dynamic key. Encryption method was performed by encrypting the data on the side of the web browser before it was sent, then the decryption process was done on the web server side. The testings of this research were performed in case of login authentication. The results show the proposed method has successfully secured data exchange on HTTP.

1. Introduction

Data exchange security is one of the main issues in various web-based applications. In the data communication side, Hypertext Transfer Protocol (HTTP) is still widely used as one of the standard protocols that governs data exchange via web. Whereas the use of this protocol can cause security problems because the data exchange process is not encrypted [1]. The application of encryption process is one of the most commonly used ways in securing submitted data. This has been widely applied on Content Management System (CMS) WordPress, Joomla, Drupal which use cryptographic hash function of Message-Digest Algorithm 5 (MD5). But this encryption process works on the side of web server so that data sent from web browser is still in the form of plaintext.

Currently, it is known that there are several methods that can be used to improve data exchange security on HTTP by using Hypertext Transfer Protocol Secure (HTTPS) [2]. However, related researches show this solution is still vulnerable to SSL Stripping and Man-In-The-Middle (MITM) attacks either through the Poisoning Address Resolution Protocol (ARP) or through Fake Certificate
Attack or Phishing Attack [1], [3], [4]. Another solution is to use Secure Hypertext Transfer Protocol (SHTTP) by encrypting the application layer using the combination of Rivest Shamir Adleman (RSA) and MD5 algorithms [5]. However SHTTP is still rarely used and researches on this method have not been reviewed widely.

Based on the problems presented above, this paper proposes the improvement of security during data exchange process through HTTP using client-side encryption method by performing data encryption process at application level on the web browser side. In this method, there are some cryptographic algorithms that can be used, including Data Encryption Standard (DES), Advanced Encryption Standard (AES), Triple Data Encryption Standard (3DES) and Rivest Shamir Adleman (RSA). This research uses AES dynamic key 128 bits algorithm [6]. The selection is based on the consideration that AES has balance between security and flexibility in various softwares and hardwares [7]. In addition, AES published by the National Institute of Standards and Technology (NIST) as a substitute for DES is also one of the symmetric-key algorithms used as standard algorithms until today [8]. Testings of this research were done in case of login authentication using HTTP. The research results show that the proposed method can secure the data exchange on login authentication process of HTTP. The next section of this paper describes the research methods used, materials, methods and results.

2. Methodology

In this study, the proposed method to secure data exchange on HTTP is using Client Side Encryption. Client Side Encryption is one of methods used to improve the security of data stored on cloud server [9]. The use of this method was utilized to perform data encryption which can be on web browser side before data is sent to the web server via HTTP like Figure 1.

![Figure 1. Client side encryption](image)

As discussed in the introduction, this study uses AES dynamic key as Client-side Encryption algorithm. AES dynamic key is the development of AES algorithm that generates dynamic encryption keys based on the time value. The dynamic key consists of 3 main components i.e. time value, salt value and user password with length of 128 bits respectively. The generation of 128 AES dynamic keys can be shown in equation 1 [6].

\[
K = K \oplus K2
\]  

Where
K = AES Key (128 bit)
K1 = Time value when User logs in \( \oplus \) Salt Value
K2 = Password User \( \oplus \) Salt value

Security is an important factor in an algorithm. In standard AES algorithm, to be able to break AES key will take 1.02 x 1018 years [10]. When viewed from the key usage period, this dynamic key generator concept has a better security value because it can minimize the use of static key in long-term
period. This happens because the key that is used will change each time the encryption process. This change is caused by an ever-changing mix of time values. This is also in line with NIST's recommendation as an agency that publishes AES on the possibility of security concerns over the long-term use of AES static keys [11].

The implementation of Client-side Encryption on HTTP using the AES dynamic key algorithm can be seen in Figure 2.

Login authentication process like Figure 2 can be explained as: client / web browser sends plaintext data in the form of username and password to web server. Before sent, the data is encrypted first using Client Side Encryption method so that the data through HTTP will be in the form of ciphertext. Furthermore, on the web server the ciphertext data is decrypted into plain text.

2.1 Material
In this study, observation of data exchange on HTTP was analyzed using wireshark software. XAMPP is a web server software used in the simulation process. On the client side the implementation of AES dynamic key algorithm was written using client-side programming (javascript). While on the server side, PHP programming language was used. In addition, this study also tested the response time of the proposed method. The number of responded time samples observed is 25 samples.

2.2 Methods
As described in the previous section, this section will describe the Client-side Encryption proposed to secure the data exchange on HTTP. Before this method is applied, first of all it is necessary to understand the standard scheme of data exchange process on HTTP as in Figure 3 [12].
Scheme in Figure 3 is the flow of data exchange process on HTTP. In the scheme, the problem occurs is data on HTTP request namely header and body are sent in the form of plaintext. To solve the problem, HTTP request data will be encrypted before it is sent as described in Figure 2. While pseudocode of AES dynamic key encryption and decryption process proposed in this research can be seen in Figure 4 and Figure 5.

### AES Dynamic Key Encryption

1. **Input:** U (Username), P (Password), K (Key), Salt, WM (Time_Start)
2. WM = get time when the user logged in [key length 128 bit]
3. Salt = value of salt (static/dynamic) [Key length 128 bit]
4. K1 = Salt + WM
5. K2 = Salt + P
6. K = K1 + K2
7. **Output:** CT (Ciphertext)
8. \( (N_r, w) \rightarrow \text{key expansion}(K) \) \( \{N_r: \text{Total round}, w: \text{Array byte key round} \}
9. CT = P
10. AddRoundKey(CT, w[0...3])
11. for \( i = 1 \rightarrow N_r \) do
12. SubBytes(CT)
13. ShiftRows(CT)
14. if \( i \neq N_r \) then
15. MixColumns(CT)
16. end if
17. AddRoundKey(CT, w[\((i^*4)\ldots(i^*4+3)]
18. end for

### AES Dynamic Key Decryption

1. **Input:** CT (Ciphertext), K (Key), Salt, WM, WP (Time of Process), TW (Time tolerance), BW (Time limit), use_key (UK)
2. WP = get the time value when the authentication process
3. TW = 36 second / optional
4. BW = WM + TW
5. UK = 0
6. **Output:** P (Plaintext Username dan Password)
7. while \( (WM = BW) \) do
8. if WM = WP
9. K1 = Salt + WP
10. K2 = Salt + P
11. K = K1 + K2 (Key length 128 bit)
12. if use_key == 0
13. \( (N_r, w) \rightarrow \text{Key Expansion}(K) \) \( \{N_r: \text{Total round}, w: \text{Array byte key round} \}
14. P = CT
15. AddRoundKey(P, w[\(w^*4 \ldots w^*4+3)]
16. for \( i = 1 \rightarrow N_r \) do
17. InvSubBytes(P)
18. InvShiftRows(P)
19. AddRoundKey(P, w[\(w^*4 \ldots w^*4+3)]
20. if \( i \neq N_r \) then
21. InvMixColumns(P)
22. end if
23. end if
24. use_key++
25. end if
26. end if
27. WM = WM + 1
28. end while

Pseudocode of AES dynamic key encryption
Figure 5. Pseudocode of AES dynamic key decryption

Figure 4 describes the generation of dynamic key stage process in AES encryption algorithm. The Salt value, Password (P), and the start time value (WM) i.e. the time value when user login is the initial input data to be used in the key generator process with the length of 128 bit, respectively. The first stage of key generation is to perform an Exclusive Or (XOR) operation between Salt and WM values to generate K1. The same operation is also performed between the Password value and the salt value to generate K2. The last step is to perform XOR operation between K1 and K2. These values will then be used as AES (K) encryption key with length of 128 bit. While P is the plaintext to be encrypted. The next stage is performing 4 main processes of AES encryption which are AddRoundKey, SubByte, ShiftRows, and MixColumns to produce ciphertext (CT). After going through encryption process, then ciphertext data will be sent from web browser to web server through HTTP.

On the web server side, data of ciphertext and K were received as inputs in the AES description process as shown in Figure 5. In the decryption process, looping time was performed until it satisfied BW value condition to obtain a match between WM and WP values. When a match was found, the salt value would be subjected to XOR operation with WP value to generate K1 and password with salt values to generate K2. Furthermore, the value of K was obtained by performing XOR operation between K1 and K2. This value would be used as a decryption key with a length of 128 bits. The next process was the inverse of the encryption process by using inverse transformation of all basic transformations in the encryption process i.e. InvSubBytes, InvShiftRows, InvMixColumns. While the AddRoundKeys transformation is self-inverse on condition of using the same key.

3. Results
This section will explain the implementation of Client Side Encryption on HTTP. The observations were performed before and after the application of AES encryption on HTTP communication path. In the initial test, data transmission test through the login form was performed as shown in Figure 6. Account user information used is username = tes and password = tes13579.
After user has sent data like Figure 6, then data packet was analysed on HTTP. This data was analysed using Wireshark software. The results of the analysis can be shown in Figure 7.

Figure 7 shows the data of account login user sent via the HTTP is still in the form of plaintext so it is very risky if this information known by others. Figure 8 below shows the analysis results of data packet after AES encryption is applied on the web browser side. After using Client-side Encryption, data of account login user is successfully encrypted on the web browser side so the data sent via HTTP protocol is in the form of ciphertext.
The next observation was performed to know the response time of the proposed method. This observation used 25 test samples. The response time value obtained in each test can be seen in Figure 9.

Figure 9. Response time of HTTP

4. Discussion
Based on the observed data, Client Side Encryption method with AES Dynamic Key encryption can be applied to solve data security problem on HTTP request by encrypting data on web browser side before data is sent. However, the application of the method has an impact on increasing the response time value generated with an average of 0.0054s. Meanwhile, the initial response time value before this method applied is 0.0025s. The increase of response time value is due to the computational time required for data encryption process using AES algorithm.

5. Conclusion
The results of this study show that Client-side Encryption method with AES encryption dynamic key can improve data exchange security on HTTP because it is able to encrypt data at application level in web browser side before data is sent to web server.

References
[1] I. Homoliak, D. Ovsonka, M. Gregr and P. Hanacek, "NBA of obfuscated network vulnerabilities' exploitation hidden into HTTPS traffic," The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014), London, 2014, pp. 310-317.

[2] W. M. Shbair, T. Cholez, J. Francois and I. Chrisment, "A multi-level framework to identify HTTPS services," NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, 2016, pp. 240-248.

[3] S. Stricot-Tarboton, S. Chaisiri and R. K. L. Ko, "Taxonomy of Man-in-the-Middle Attacks on HTTPS," 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, 2016, pp. 527-534.

[4] Puangprontpatig, S.; Sriwiboon, N., "Simple and Lightweight HTTPS Enforcement to Protect against SSL Stripping Attack," International Conference of Computational Intelligence, Communication Systems and Networks (CICSyN), 26 Jul. 2012

[5] Adam S., (1999, Mei.). An Overview of SHTTP. National Academy of Sciences Cryptography. USA. [Online]. Available: http://www.homeport.org/~adam/shttp.html

[6] Z. Musliyana, T.Y. Arif, R. Munadi, "Security Enhancement of Advanced Encryption Standard (AES) using Time-Based Dynamic Key Generation," ARPN Journal of Engineering and Applied Sciences, Vol. 10, No. 18, Oct. 2015

[7] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov and V. Rijmen, "Trade-Offs for Threshold Implementations Illustrated on AES," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 7, pp. 1188-1200, July 2015.

[8] M. Bedoui, H. Mestiri, B. Bouallegue and M. Machhout, "A reliable fault detection scheme for the AES hardware implementation," 2016 International Symposium on Signal, Image, Video and Communications (ISIVC), Tunis, Tunisia, 2016, pp. 47-52.

[9] S. Kumar Da, Md. A. Hossain, Md. Ar. Sardar, R. K. Biswas, "Performance Analysis of Client Side Encryption Tools," International Journal of Computer Science Issues, Vol. 2, 16 Sept. 2014.

[10] M. Arora. (2012, Juli.). How secure is AES against brute force attacks?. Freescale Semiconductor,USA.[Online].Available:http://www.eetimes.com/document.asp?doc_id=1279619.

[11] NIST. (2012, Juli.). Recommendation for Key Management. National Institute of Standard and Technology(NIST),USA.[Online].Available:http://csrc.nist.gov/groups/ST/toolkit/key_management.html.

[12] NTU. (2012, May.). HTTP (HyperText Transfer Protocol), USA. [Online]. Available: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html