DOI: http://dx.doi.org/10.26483/ijarcs.v16i4.7305

ISSN No. 0976-5697

Volume 16, No. 4, July-August 2025

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

IMPROVE DATA ENCRYPTION BY USING DIFFIE-HELLMAN AND DNA ALGORITHMS, AUTHENTICATED BY HMAC-HASH256

Nada Abdul Aziz Mustafa
Information Technology section
University of Baghdad, Collage of Languages
Baghdad, Iraq

Ali T. Al-Quraishi
Information Technology Department
Ministry of Planning
Baghdad, Iraq

Abstract: The need for means of transmitting data in a confidential and secure manner has become one of the most important subjects in the world of communications. Therefore, the search began for what would achieve not only the confidentiality of information sent through means of communication, but also high speed of transmission and minimal energy consumption, Thus, the encryption technology using DNA was developed which fulfills all these requirements [1]. The system proposes to achieve high protection of data sent over the Internet by applying the following objectives:

1. The message is encrypted using one of the DNA methods with a key generated by the Diffie-Hellman Ephemeral algorithm, part of this key is secret and this makes the process of predicting the key very difficult.
2. Ensuring the integrity and reliability of the transmitted data using the HMAC-HASH256 algorithm that is resistant to attacks, where the 256 hash function is used with a key generated from the Diffie-Hellman Ephemeral algorithm.
3. Analyzing the system by trying to measuring the impact of using encryption with authentication on cost and speed and calculating the time taken to implement the HMAC-SHA256 algorithm. System implementation was done by using IntelliJ IDEA with java FX.

Keywords: DNA encoding, Diffie-Hellman Ephemeral, HASH256, MDC and MAC, HMAC-HASH256.

1. INTRODUCTION

The purpose of encryption is to transfer data and information in a way that is difficult for an attacker to reveal its contents. Many encryption algorithms have been developed and their uses have varied. DES-AES algorithms are used to maintain the confidentiality of data, while MDF-SHA512 algorithms are used to ensure the integrity of the data and prove that it is not subject to change or modification by an unauthorized person [2], [3]. With the development of attack methods and the increased need to store massive data, there has become an urgent need for algorithms that not only provide high levels of security, but also have massive storage and low energy consumption, and this is what was obtained from encryption using DNA computing [4]. Many methods are used for encryption using DNA, such as generating keys and hiding information in an image or video and it is considered one of the fields known as biological technology that has provided large areas of storage [5]. Encryption using DNA is considered a new technology for using the properties of DNA molecules, whose algorithms can confront statistical and differential attacks by using one of the four nitrogenous bases when encrypting to transform each letter of the message into a different form, which are (adenine A, cytosine C, guanine G, and thymine T) [4]. The encryption process is carried out by converting secret data into a sequence of nitrogenous bases, storing it and retrieving data from it through analyzing the stored DNA sequence [5].

Encryption algorithms using DNA are among the rapidly developing technologies, and complex algorithms can be designed with high memory capacity and low storage energy consumption, thus it is possible to store all the world's data in a few milligrams [4]. One gram of DNA contains 1021 DNA bases (that is 108 terabytes of data). This technology is considered resistant to attacks because of the difficulty of manipulation with it [5].

2. DIFFIE-HELLMAN

The Diffie-Hellman algorithm is a method of creating or exchanging keys between the sender and the recipient in a way that ensures confidentiality when sending them over an unsecured network, and the generated keys are used only once [6]. Both the sender and the recipient agree on two declared values (N, M), provided that N is a prime number, in addition to a random number for each of them (a, b), which is private and undeclared [7].
The algorithm has proven its strength in generating, exchanging keys and the difficulty of guessing the key due to the complexity of the mathematics used [8]. Diffie-Hellman used in TLS protocols may be ADH It is considered ineffective, because it is not possible to verify the identity of both the sender and the recipient, it is vulnerable to a man-in-the-middle attack [9], or it may be DH, which is ineffective due to the use of fixed keys by the sender and recipient [10]. As for DHE, keys are used once for each connection, if the key currently used is revealed, it cannot be used to decrypt previous conversations that took place with different keys [11]. To achieve high confidentiality, the DHE method is used with encryption algorithms such as AES [12].

3. HASH256 ALGORITHM

It is one of the hashing algorithms used in encryption, where each letter of the message is converted into a hexadecimal and the input is divided into blocks with a length of 512 bits (the input is less than 64^2 bits) [13]. 64 rounds of complex mathematical operations are applied to each block, and the result is used again as input for the second block until the last block is reached. The result we obtain is a value of fixed length, equal to 256 bits [14]. The input cannot be obtained from the hash output, and any change in the input leads to a significant change in the hash value [13].

4. MDC AND MAC

The security and integrity of data during its transfer between server devices is vulnerable to the possibility of modifications to it by a hacker, and this gives the impression that the data may have been changed and modified [15]. Thus, the focus is on the issue of data integrity here [16]. That is, even if this data is encrypted in complex ways such that it is difficult to decrypt, this does not guarantee that it will not be tampered with and possibly changed during its transmission [15]. Here appears the urgent need to discover the modification made by the hacker using the MDC hash function, where a summary of any message is obtained [17]. This is done by producing a fixed-sized string that represents the contents of the message, and this summary will serve as a single fingerprint of the sender's original message [18]. The recipient will use the same function as the sent message to obtain the fingerprint [19]. The integrity of the data will be verified when the fingerprint is identical to the sent fingerprint, meaning the sent message has not been tampered with, provided that the hash function used is the same by the sending and receiving parties [18]. Examples of hash functions are SHA-256 and MD5 [19].
The message and summary might be modified by the hacker during sending, and the recipient would not know that the message has been modified [17]. Therefore, we need to ensure the integrity of the message by using MAC and detecting whether its contents have been modified, the message is correct and intact [20]. In MAC, a secret key is used between the sender and the recipient, then the message is sent with the MAC output, after which the recipient separates the message and calculates the MAC and the secret key, provided that he is aware of the hash function used [21]. A comparison is taken between the two functions, and if a match occurs, it means that the message is original and has not been tampered with [20]. As long as the hacker does not obtain the secret key, the message can't be modified and a MAC that matches the MAC generated by the recipient can't be created [21].

5. HMAC-HASH256 ALGORITHM

It is an algorithm that represents an authentication code consisting of a hash function that is used to verify the authenticity of data sent using a secret key [22], it is considered more secure than MAC because of its use of a complex hash function in addition to padding [17]. It is implemented by specifying the key (a shared key of length 32 bytes) [23]. If the length of the key is greater than 32 bytes, it is hashed using the same algorithm (hash 256), and after preparing the secret key, the hash function is combined with a secret key to prove the integrity and authenticity of the transmitted data and prevent impersonation [18].

6. THE PROPOSED SYSTEM

The system is implemented in two basic stages: the first stage is encrypting the sent data and including it within the DNA sequence and using the HMAC-HASH256 algorithm to confirm the integrity of the sent data. The second stage is decrypting, confirming the reliability and integrity of the received data. The first stage takes place in two basic steps:
The first step is to encrypt the data by applying the following points:
1.Generate a key using the Diffie-Hellman Ephemeral (K1) algorithm.
2. Executing XOR to the first letter of the message after transforming it to 8 bits with the key created from the Diffie-Hellman's algorithm from the first step (K1)
3. Choosing the base of the DNA sequence and converting it to the binary. In our system, A=00, T=11, C=01, G=10 were used, and based on the value of the second key agreed upon between the two parties, and in our system K2=2, where the chain is divided based on it. (The user may choose a different value for the second key).
4. Each bit of the first letter of the secret message from the result of the second step is added at the beginning of every 2 bits of the DNA sequence of the string (from the result of step 3), then the string is merged and converted to nucleotide base and the process is repeated for all letters of the message, as shown in figure 1.

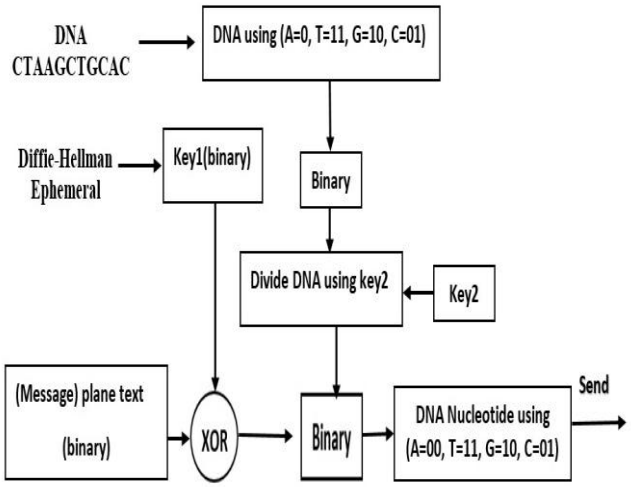
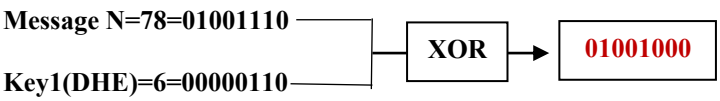


Figure1. Shows the encryption in the proposed system

An example of encoding the first letter of the message (NADA), which is the letter N



when A=00, T=11, G=10, C=01 and Key2=2 then DNA=CTAAGCTGCAC =01 11 00 00 10 01 11 10 01 00 01 = 001 111 000 000 110 001 011 010 01 00 01 = ATTAAATACCGGCAC.

Step Two: Confirm the integrity of data arrival (HMAC-HASH256 implementation steps):
The process of implementing HMAC-HASH 256 is done by creating a 32-byte key generated by the Diffie-Hellman Ephemeral algorithm and two secret keys, IPad and Opad, by applying the following steps:
1.Preparing the key generated by the Diffie-Hellman algorithm (if the key is larger than 512 bits, it is hashed using the Hash256 function in order to obtain a key with a length of 32 bytes (512 bits). However, if the key is less than 512 bits, the key is filled with zeros to reach the required block size.
2.Create an internal fill: (IPad) It is created by performing an XOR operation between the key generated from step 1 and the repeated value 0x36 to fill the block size.
3.The result (IPad) is added with the letters of the secret message to obtain Stage1, and then hashed using the Hash256 algorithm.
4.Create an external fill: (OPad) It is created by performing an XOR operation between the key resulting from step 1 and the value 0x5c duplicated to fill the block size.
5.The output of step 4 is hashing from the output of step 3 to obtain stage2.
6.Hash stage2 using the same hash algorithm to obtain HMAC-HASH 256 which will be sent to the recipient, see Figure 2,3,4.

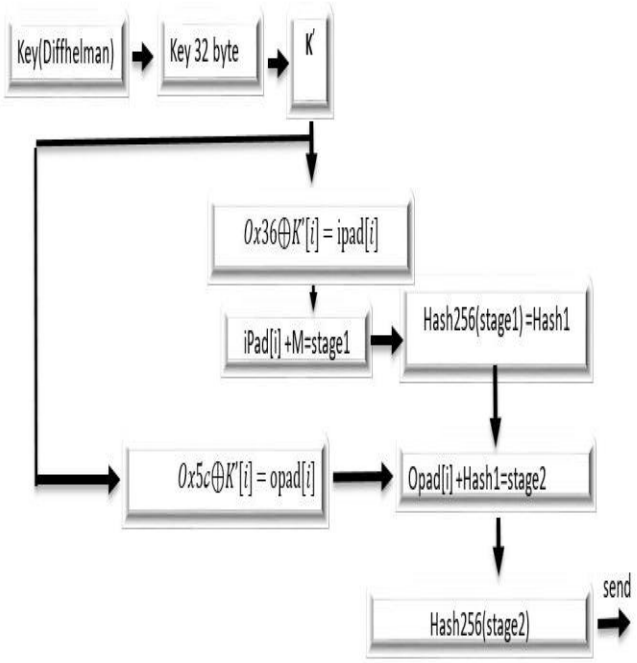


Figure 2. Shows the HMAC-Hash256 with Diffie-Hellman keys.

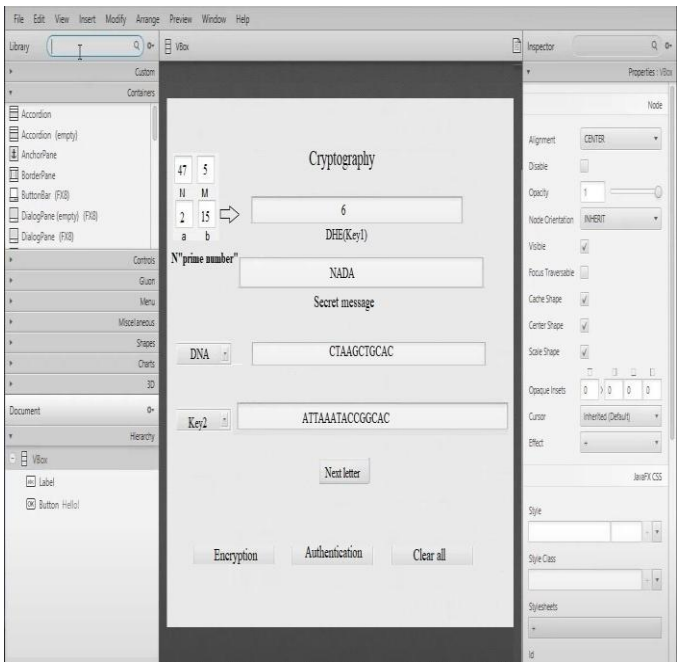


Figure 3. Shows the encryption of the first letter of the secret message using Intellij with JavaFx

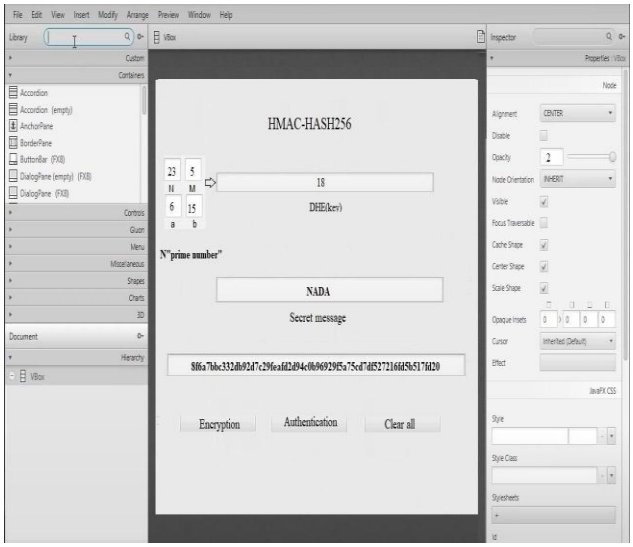


Figure 4. Shows the HMAC_HASH256 algorithm with Diffie-Hellman keys using Intellij with JavaFx.

The second stage is the process of decryption and confirming the integrity of the transmitted data, which is carried out through the following steps:
1. Convert the sent nucleotide base to the binary sequence (A=00, T=11, C=01, G=10).
2. Divide the result from step 1 using Key 2+1, which equals 3, to get a sequence.
3. Collect the first bit of each segment and obtain a sequence consisting of 8 bits (which represents the first letter of the message).
4. XOR the result of step 3 with Key1.
5. Convert the binary sequence to ASCII text value.
6. To confirm the integrity of the message, the key K3 generated from the Diffie-Hellman algorithm is used and added to the message resulting from step 4 (after obtaining the complete secret message) and applied to the hash

function 256, the result represents HMAC-HASH 256, then it is compared with the result received from the message sender.

Example: Decoding the first letter of a message
ATTAAATACCGGCAC=001111000000110001011010010001
Dived DNA by using key2+1=3= 001 111 000 000 110 001 011 010 01 00 01
Take the first bit from each segments= 01001000 XOR with Key1
Key1(DHE)=6=00000110 XOR 01001000 = 01001110 = 78=N (first character from the message).
To confirm the integrity of the message and verify that it has not been hacked:
Hash 256 from the sender=
8f6a7bbc332db92d7c29feafd2d94c0b96929f5a75cd7df527216fd5b517fd20
Hash256(Key3+message) = Hash256(18NADA) =
8f6a7bbc332db92d7c29feafd2d94c0b96929f5a75cd7df527216fd5b517fd20
It is equal to the hash sent by the sender, see Figure 4.

7. Analyzing the Time Taken to Execute the HMAC-HASH256 Algorithm, Measuring the Storage Speed Using the DNA Algorithm

The time taken to execute the HMAC algorithm was calculated using the Java code '() System.nanoTime', where the time is measured before and after the HMAC execution, and the difference between them is calculated, the result is then converted from nanoseconds to milliseconds by dividing the result in nanoseconds by 1,000,000 (1 Millisecond = 1000000 nanoseconds). The storage rate can be calculated based on the amount of data stored and the time taken to write and read it. For example, if 1MB of data is stored in 10 minutes, the storage rate would be 0.1MB per minute, see figure 5 and table 1.

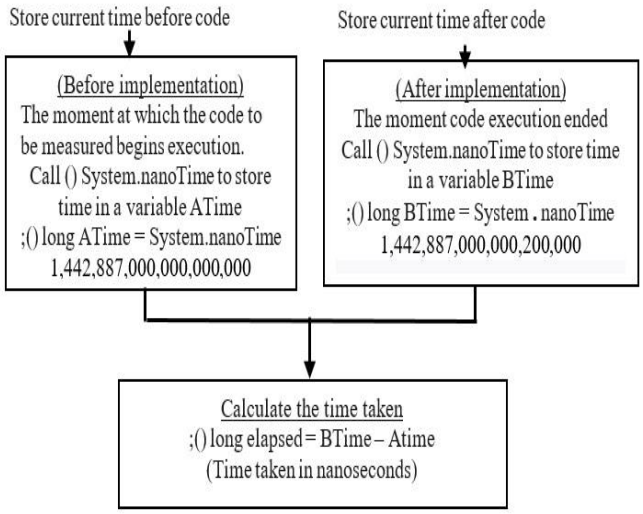


Figure 5. Shows Calculate the time taken to execute the HMAC algorithm in the system

Example:
Time taken= After implementation- Before implementation
1,442,887,000,000,200,000- 1,442,887,000,000,000,000
= 200,000 nanoseconds= 0.2 milliseconds.

Table 1. Execution time of HMAC algorithm for system using Java functions

Time in milliseconds	Execution speed time	System output
less than 1 ms	Very fast	0.2 ms The system implementation speed is considered very fast (less than 1 ms)
2-11 ms	Between normal and acceptable	
More than 12	slow	

The time evaluation depends on the data size, algorithm type, and device performance. If the difference is (for example, from 100,000 to a few million nanoseconds), this is considered a normal and very fast time and does not indicate slow execution.

8. Conclusions

1.The system has proven to provide high levels of security by encrypting confidential data using the DNA algorithm which has huge storage capacity, with keys generated using the Diffie-Hellman Ephemeral algorithm, where keys are generated instead of exchanging and sending them through unsafe paths that expose them to hacking. The advantage of generating keys using the Diffie-Hellman Ephemeral algorithm is that they replace their keys in each session, which adds additional complexity to guessing the keys. That is, when a key is hacked within a session, it does not affect other sessions that have different keys. In addition, the error rate may be non-existent when compared to other keys, provided that numbers are not used Simple or fixed prime.

2.The HMAC-HASH256 algorithm was made complex by using it with a key generated by the Diffie-Hellman Ephemeral algorithm to ensure that the transmitted confidential data would not be modified or tampered with. In addition, it provided stronger resistance to attacks, such as collision attacks and man-in-the-middle attacks, in addition to any change in the data sent will be detected by the recipient. The recipient must then combine the key with the message using the same method as the sender (the key then the message or vice versa), otherwise it won't generate the same summary to confirm reliability.

3.Using an encryption system with authentication achieves a high level of security at the expense of cost and speed. See Table 2.

Table 2. The difference between using encryption and encryption with reliability.

4.When using the reading speed measurement law (read speed = data read / time spent), i.e. (MBs = Mb/second), it was found that the storage speed of DNA is Below average

Encryption with reliability	Encryption
It may affect the speed of implementation	The speed of encryption and decryption is high
High costs (licenses, equipment, maintenance, technical support and training)	Low costs
Reduces security risks	Security risks are high

even though it provides high storage, see Table 3.

Table 3. Shows the time taken to execute HMAC-HASH256, the storage speed (DNA).

Data size	Time taken HMAC-HASH256	Reading speed MBs=Mb/seconds To obtain faster storage DNA
2KB	Less than MS	fast
2MB	20-60 MS	middle
30MB	1-many seconds	Below average

5. Through the results of Table No. 3, which measures the time taken to implement the HMAC-HASH256 algorithm, it is shown that the larger the size of the data, the greater the time taken for implementation with Ensuring high security for the transmitted data. The execution speed of the HMAC-HASH256 algorithm ranges from a few milliseconds to several seconds, and the execution speed depends on the memory and processor.

9. REFERENCES

[1] M.K. Padmapriya, Pamela Vinitha Eric,"A Technique of Data Security using DNA Cryptography with Optimized Data Storage", Journal of System and Management Sciences, ISSN 1816-6075 (Print), 1818-0523 (Online), Vol. 12 (2022) No. 4, pp. 412-438, DOI:10.33168/JSMS.2022.0425.

[2] Yash Shah, Riddhi Rane, Siddhesh Kharade , Rutuja Patil," Analysis of AES and DES Algorithm", International Journal of Trend in Research and Development, Volume 7(2), ISSN: 2394-9333, IJTRD | Mar – Apr 2020 Available Online@www.ijtrd.com.

[3] Salah Taha Allawi, Nada Abdul Aziz Mustafa," Image encryption based on combined between linear feedback shift registers and 3D chaotic maps", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 30, No. 3, June 2023, pp. 1669~1677, ISSN: 2502-4752, DOI: 10.11591/ijeecs. v30.i3. pp1669-1677.

[4] Khobzaoui Abdelkader, Benyahia Kadda, Mansouri Boualem, Sofiane Boukli Hacene," DNA-Based Cryptographic Method for the Internet of Things", International Journal of Organizational and Collective Intelligence 12(1):1-12, January 2022, DOI:10.4018/IJOCI.2022010101

[5] Bahubali Akiwate, Latha Parthiban," A DNA Cryptographic Solution for Secured Image and Text Encryption", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 2, 2021.

[6] Aryan, Chaithanya Kumar, Durai Raj Vincent P M," Enhanced diffie-hellman algorithm for reliable key exchange", IOP Conf. Series: Materials Science and Engineering 263 (2017) 042015 doi:10.1088/1757-899X/263/4/042015.

[7] Om Pal, Bashir Alam," Diffie-Hellman Key Exchange Protocol with Entities Authentication", International Journal of Engineering and Computer Science ISSN:2319-7242 Volume 6 Issue 4, April 2017, Page No. 20831-20839, Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i4.06

[8] Parth Sehgal ,Nikita Agarwal, Sreejita Dutta, P.M.Durai Raj Vincent, "Modification of Diffie-Hellman Algorithm to Provide More Secure Key Exchange" ,June 2013, International Journal of Engineering and Technology 5(3):2498-2501, License CC BY 4.0.

[9] Akshat Puri, Piyush Saxena, Gitesh Kumar, Gitesh Kumar," Implementation of Diffie-Hellman Algorithm for Information Security", International Journal of Engineering Research in Computer Science and Engineering (IJERCSE), ISSN (Online) 2394-2320, Vol 10, Issue 5, May 2023.

[10] V. Vinothini, C. Muthukumaran," Key Exchange Technique in Cryptography Using Diffie-Hellman Algorithm", International Journal of Computer Sciences and Engineering, E-ISSN: 2347-2693, Vol.-7, Special Issue, 4, Feb 2019.

[11] CH.Bhanu Prakash, Shaik Shavali," FPGA Implementation DIFFIE-HELLMAN key Exchange Algorithm using DES, International Journal of Innovative Research in Electronics and Communications (IJIREC), ISSN 2349-4042 (Print) & ISSN 2349-4050 (Online), Volume 1, Issue 4, July 2014, PP 26-36.

[12] Demba Sow, Mamadou Ghouraissiou Camara & Djiby Sow1, "Attack on Strong Diffie-Hellman-DSA KE and Improvement", Journal of Mathematics Research; Published by Canadian Center of Science and Education, ISSN 1916-9795 E-ISSN 1916-9809; Vol. 6, No. 1; 2014.

[13] Fariha Jahan, Mayel Mostafa, Shahrin Chowdhury," SHA-256 in Parallel Blockchain Technology: Storing Land Related Documents, International Journal of Computer Applications (0975 – 8887), Volume 175 – No. 35, December 2020.

[14] Nada Abdul Aziz Mustafa, "Analysis attackers' methods with hashing secure password using CSPRNG and

PBKDF2", Wasit Journal of Engineering Sciences, Vol. 12 No. 2 (2024), 12(2), 60-70. https://doi.org/10.31185/ejuow.Vol12.Iss2.502.

[15] Muhamad Rais Rabtsani, Agung Triayudi, Gatot Soepriyono," Combination of AES (Advanced Encryption Standard) and SHA256 Algorithms for Data Security in Bill Payment Applications", Journal of Technology and Information Systems, ISSN: 2985-8933 (Media Online), Vol 2, Issue 1, February 2024, Page 175-189, DOI: 10.58905/SAGA.vol2i1.250.

[16] Prateek Baranwal, Ritik Katiyar, Prajusha Kundu, Meenakshi Yadav," Implementation of SHA256 for NFT Management Using Blockchain", International Research Journal of Engineering and Technology (IRJET), E-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 11 Issue: 06 Jun 2024.

[17] Nureni Ayofe Azeez and Onyema Juliet Chinazo, "Achieving Data Authentication With HMAC-SHA256 Algorithm", Publisher: Georgian and MICM - Muskhelishvili Institute of Computational Mathematics of the Georgian Technical University, Computer Science and Telecommunications 2018|No.2(54), ISSN 1512-1232, Impact Factor РИНЦ 2018: 0,025,2018.

[18] Dilli Ravilla; Chandra Shekar Reddy Putta, "Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs", International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), DOI: 10.1109/EDCAV34670.2015 29-30 Jan. 2015, Publisher: IEEE.

[19] Alaa B. Baban, Safa A. Hameed, "Securing a Web-Based Hospital Management System Using a Combination of AES and HMAC", Iraqi Journal for Electrical and Electronic Engineering, College of Engineering, University of Basrah Vol. 19, Issue 1, June 2023, DOI: 10.37917/ijeee.19.1.12.

[20] K V V N L Sai Kiran, Harini N," Evaluating Efficiency of HMAC and Digital Signatures to Enhance Security in IOT", International Journal of Pure and Applied Mathematics, Volume 119 No. 12 2018, 13991-13997, ISSN: 1314-3395, url: http://www.ijpam.eu.

[21] Lihui Lin, Kaizhi Chen, and Shangping Zhong," Enhancing the Session Security of Zen Cart based on HMAC-SHA256", Ksii Transactions on Internet and Information Systems, VOL. 11, NO. 1, Jan. 2017, Copyright 2017 KSII, January 31, 2017, ISSN: 1976-7277, DOI: 10.3837/tiis.2017.01.025

[22] N. A. Azeez, and O.J. Chinazo, "Achieving Data Authentication with Hmac-Sha256 Algorithm", Computer Science & Telecommunications, Vol. 54, No. 2, 2018.

[23] BoSun Park, JinGyo Song and Seog Chung Seo, "Efficient Implementation of a Crypto Library Using Web Assembly", Electronics 2020,9, 1839; doi:10.3390/electronics9111839, November, License CC BY 4.0.

