Mathematical Journal of Modelling and Forecasting
Vol. 3, No. 2, December 2025, pp. 9-17

Digital Signature Using the SHA-512 Hash Function and the ElGamal Cryptographic Algorithm for Document Security

Adib Yusron Bokari[1,*] Meira Parma Dewi[2]

[1,2] Department of Mathematics, Universitas Negeri Padang, Indonesia
[*] adibyusron8@gmail.com

Abstract. This study proposes a secure digital signature scheme using the SHA-512 hash function and the ElGamal cryptographic algorithm to ensure the authenticity and integrity of digital documents. The increasing reliance on digital communication has made document security a critical concern, as digital documents are vulnerable to unauthorised access, tampering, and fraud. To address this issue, a digital signature approach is implemented, leveraging the SHA-512 hash function to generate a unique message digest and the ElGamal algorithm to encrypt and decrypt the signature. The proposed system ensures that any alteration or modification of the document will result in a mismatch between the expected and actual hash values, thereby detecting potential security breaches. The combined use of SHA-512 and ElGamal algorithms provides a robust and secure digital signature scheme, guaranteeing the confidentiality, integrity, and authenticity of digital documents. This study demonstrates the effectiveness of the proposed scheme in preventing fraud and protecting digital documents from various security threats, making it a reliable solution for secure digital communication.

Keywords: Digital Signature, Document Security, Cryptography.

1 Introduction

The communication process, which involves sending messages, is now primarily conducted through digital documents. Digital documents are data in the form of text that have an open nature, meaning the content can be easily read and modified by unauthorised parties. This renders the security factor of digital documents highly insecure. Security refers to the measures we take to prevent or detect fraud in a system based on digital documents, thereby protecting against unauthorised access, use, dissemination, destruction, alteration, and other threats. For this reason, an approach is needed to secure digital documents, such as encryption, steganography, digital signatures, and hashing of the digital documents [1].

The security of digital documents can be ensured with a digital signature, which is a form of modern cryptography implementation. A digital signature is a cryptographic method used to verify the authenticity and integrity of a digital document, having a dynamic nature, which means that the same signature cannot be used on different documents. Therefore, digital signatures are considered safer and an effective solution for maintaining message authentication, especially in digital documents. This is because a digital signature does not mean a digitised signature, but a collection of codes that are processed using cryptography [2].

Public key cryptographic algorithms, such as RSA, DSA, GOST, and ElGamal, are used in digital signature creation to prove message authentication and provide authentication during the encryption and decryption of messages [3]. For example, the ElGamal algorithm, a public-key cryptographic algorithm, performs the encryption and decryption processes using two distinct keys. The ElGamal algorithm is recognised as a highly secure standard. The digital signature scheme using the ElGamal algorithm is based on the algebraic properties of large modulo prime exponents along with discrete algorithm problems [4].

In addition to the use of the ElGamal public key algorithm, digital signatures also require the combination of two cryptographic algorithms to provide maximum protection for the validity of a digital signature [5]. Performing hash calculations aims to reduce the size of ElGamal's digital signatures, thereby obtaining a smaller message digest [6]. The hash function used in creating a digital signature ensures that the signature is only valid for the specific document to which it is applied. This is because the hash function is a one-way function that can produce a signature in the form of a hash value or a concise message (message digest) of a

document. Even the slightest changes to the document will significantly impact its hash value. So that the hash function can be applied appropriately to create a digital signature of a document, thereby maintaining data integrity [7].

The Secure Hash Algorithm (SHA) is a simple set of cryptographic hashes developed to maintain and improve data security integrity. SHA-512 is one of the SHA algorithms that boasts a high level of security, mainly due to the development of several previous SHA algorithms [8]. SHA-512 has a larger block size and transforms with a size of 1024 bits, twice as large as SHA-256. This provides a higher level of security and resistance to more sophisticated cryptographic attacks [9]. A study titled "Implementation of the Secure Hashing Algorithm-512 (SHA-512) for Sign-Up Page Security in the Class of Fun Tutoring System" in 2024 concluded that SHA-512 offers a higher level of security than other hashing algorithms by producing hashes that are difficult to crack, even in the situation of a hacked database so that this system can ensure the confidentiality of user data is maintained [10]. Sinlae, in his research entitled "Cryptosystem Analysis Using Digital Signatures Based on SHA-512 and RSA Algorithms" in 2012, explained that the SHA-512 hash function can be used in conjunction with the RSA public key algorithm for encryption and decryption on digital signatures [11]. Digital signatures using the ElGamal public-key cryptographic algorithm and the SHA-512 hash function are a combination of cryptographic methods that must be implemented to provide maximum protection for the validity of a digital signature.

Therefore, based on the phenomenon, exposure, and supported by previous studies, researchers are interested in conducting digital signature-based research using the ElGamal and SHA-512 cryptography methods. With the research to be carried out, it is hoped that the application of the SHA-512 hash function can be used in conjunction with the ElGamal algorithm to produce maximum authentication of documents using digital signatures.

2 Research Methods

2.1 Data Collection Methods

The method employed in this research is the literature study method, which examines the topics to be discussed in the research using books, journals, scientific articles, and final projects related to the application of cryptographic algorithms for digital signatures. The data used in this study is a document with a minimum of one page, containing content in the form of letters, numbers, and characters in PDF (Portable Document Format) format, obtained from the author's documents. The black box testing method is used to perform validation analysis for software or system testing by testing functions based on the software requirements specifications.

2.2 Planning Stages

1) Signing Stages

   At this stage, a system design is carried out that presents a process of document signing stages.

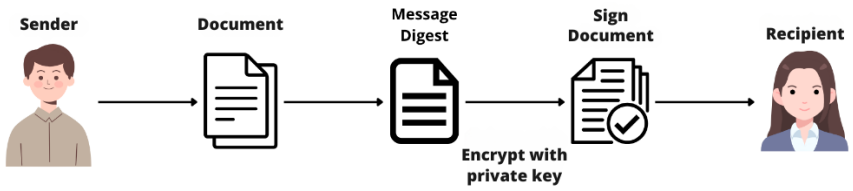

Fig 1. Signing Stages

2) Verification Stages

At this stage, the design of a system that presents a document verification

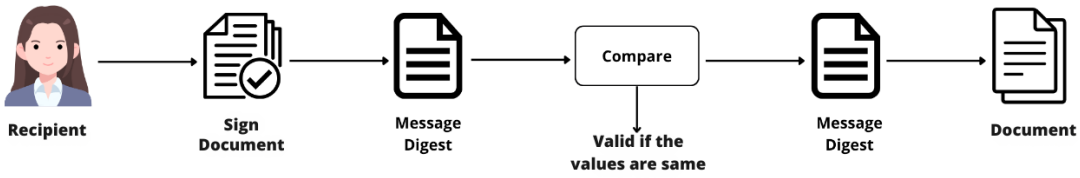

Fig 2. Verification Stages

2.3 Implementation of the SHA-512 Hash Function

A crucial step in creating a digital signature on a document is to hash the document using the SHA-512 hash function, which aims to summarise the document into a 512-bit message. Furthermore, the document can be signed with the sender's private key [12]. To obtain a 512-bit message digest, the following process is required:

1) Append the Bits with Padding

The first process is to append an additional number of bits so that the length of the message (in bits) becomes congruent with 896 mod 1024. Since SHA-512 processes messages in blocks of 1024 bits, and there are the last 128 bits of the final block that must contain the hexadecimal value of the message length, the calculation used is modulo 1024 [13].

$$message\ length \equiv 896 \bmod 1024 \tag{1}$$

2) Append Length Value

Adding the value of the message length by adding a message back with 128 bits, expressing the length of the original message in hexadecimal. If the message length is greater than 2218, then the message length is taken from the modulo 2218. If the original length of the message is equal to K bits, then 128 bits are added with K modulo 2218 [14]. The result of these first and second steps is the message that is a multiple of integers with a length of 1024 bits.

3) Initialise Hash Buffer

A 512-bit hash buffer is used to store intermediate results and final results of hash functions. A buffer hash can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialised with the following 64-bit integers (hexadecimal values) [15]:

Table 1. Register 64-bit SHA-512

Hash Buffer	Value
a	$H_0^{(0)} = 6a09e667f3bcc908$
b	$H_1^{(0)} = bb67ae8584caa73b$
c	$H_2^{(0)} = 3c6ef372fe94f82b$
d	$H_3^{(0)} = a54ff53a5f1d36f1$
e	$H_4^{(0)} = 510e527fade682d1$
f	$H_5^{(0)} = 9b05688c2b3e6c1f$
g	$H_6^{(0)} = 1f83d9abfb41bd6b$
h	$H_7^{(0)} = 5be0cd19137e2179$

These values are then stored in big-endian format, where the most significant byte of a word is at the low-address byte position.

4) Message Processing

At the heart of this algorithm is a module consisting of 80 rounds. Messages are processed in 1024-bit (128-word) blocks using a module consisting of 80 rounds F. Module F is used to process the message block M_i. which has two inputs, namely the current 512-bit buffer hash content and the 1024-bit message block. Each round takes a 512-bit buffer as input and updates its contents. Each t-spin uses a value of 64 bits W_t, which is obtained using a message schedule from the 1024-bit block that is being processed [16]. The message schedule consists of eighty 64-bit words labeled $\{W_0, W_1, \dots, W_{79}\}$. The first sixteen 64-bit words in the 1024-bit message block are M_i, while the equation obtains the remaining words in the message schedule

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{(512)}(W_{t-2)}) + W_{t-7} + \sigma_0^{(512)}(W_{t-15)}) + W_{t-16} & 16 \le t \le 79 \end{cases} \quad (2)$$

5) Output

After all of 1024-bit N blocks have been processed, the output of the N stage is a 512-bit message digest.

2.4 Application of the ElGamal Algorithm

The ElGamal algorithm is an asymmetric cryptography algorithm based on the difficulty of solving discrete logarithm problems. The ElGamal algorithm consists of three processes: key generation, encryption, and decryption [17]. The steps are as follows.

1) Generate Key

The process of generating a key in the ElGamal algorithm involves two key pairs: the public key and the private key. This process requires a prime number p, a random number g which is the primitive root of p, as well as a random number x. So that the public key pair (p, g, y) and a private key x can be obtained. In the ElGamal algorithm system, the prime number p must be greater than 225, because the ElGamal algorithm uses integer operations in its calculations, so the message sent must be converted into an integer. The conversion uses ASCII code, a numerical representation of characters commonly used on computers, with a minimum value of 0 and a maximum value of 255 [18]. Here's the process of generating key pairs:

a. Prime number $p < 225$
b. A random number $g < p$, where g is the primitive root p
c. Random number of x, where $x \in \{1, 2, 3, \dots, p-2\}$
d. Calculate the equation of y:

$$y = g^x \bmod p \quad (3)$$

e. Generate a pair of public keys (p, g, y) and private key x.

2) Encryption Algorithm

To encrypt a message, a public key (p, g, y) is required. Here's the process of the encryption algorithm.

a. Cut plaintext into message blocks $m_1, m_2, \dots,$ with values in intervals $[0, p-1]$.
b. Convert message blocks into decimal values.
c. Selects a random number k, with the condition $1 \le k \le p-1$.
d. Encrypt blocks m into value pairs (a, b) with equations.

$$a = g^k \bmod p \quad (4)$$

$$b = y^k m \bmod p \quad (5)$$

3) Digital Signature Algorithm

To sign a document message, calculations are made to produce signatures (r, s). The following is the process of signing the document message.

a. Choosing a random number k, provided that $1 \le k \le p-1$
b. Signing to generate a signature value (r, s) with that equation

$$r = g^k \bmod p \quad (6)$$

$$s = k^{-1} \times (m - x \cdot r) \bmod (p - 1) \quad (7)$$

4) Decryption Algorithm

To decrypt a message, a private key (x, p) is required. Here's the process of the decryption algorithm.

a. Decrypt a value pair (a, b) into a plaintext m with that equation
$$m = b(a^x)^{-1} \bmod p \quad (8)$$
b. Convert the obtained m value to an ASCII value.
c. Compile plaintext in sequence $m_1, m_2, m_3, \dots, m_n$
d. Perform the calculation of the verification value $v_1 = v_2$ using the equation
$$v_1 = y^r \cdot r^s \bmod p \quad (9)$$
$$v_2 = g^m \bmod p \quad (10)$$

3 Result And Discussion

The implementation of the method used to create a digital signature includes generating a key using the ElGamal algorithm, creating a message digest using the SHA-512 hash function, inserting a digital signature, and verifying the process of verifying documents that have been inserted with a digital signature.

3.1 Key Generating Process

The key generation process using the ElGamal algorithm aims to generate the public key pair (p, g, y) and the private key x. So that the prime number $p = 277$, the random number $g = 11$ which is the primitive root of p, and the random number $x = 257$ were chosen. So that the value of y can be obtained through the equation y as follows

$y = g^x \bmod p$

$y = 11^{257} \bmod 277$

$y = 199$

From the key generation process, the public key pair $(p, g, y) = (277,11,199)$ and the private key $x = 257$ were obtained.

3.2 Signing Process

After completing the key generation process, the next stage is the formation of a digital signature. Here are the stages of the digital signature formation process:

1) Hashing Process

The initial process of forming a digital signature on a message or document involves converting the message into a message digest using the SHA-512 algorithm. Then, a secret message, "password of central server: P523V3R," is selected, allowing a 128-bit message digest to be generated as follows.

Table 2. Message Digest

Message	Message Digest
password of central server: P523V3R	e9 41 94 22 c7 f0 92 d6 2b 90 4e 9d ed fa b6 a1 d3 a3 e4 23 7b 5b f2 f5 7a 58 1f fc db 6d 73 c4 99 6d 40 f2 4a 3d 77 20 39 c4 ba 91 5f ae c9 8f 6f e2 14 72 1d 8a 0e a6 c6 00 36 7e 31 6e 25 e0

2) Encryption Process

The private key generated during the key generation process is used to encrypt messages, resulting in a digital signature. Before encrypting a message, first convert the message digest to a decimal value.

Table 3. Message Digest in Decimal Value

Message Digest	Decimal
e9 41 94 22 c7 f0 92 d6 2b 90 4e 9d ed fa b6 a1 d3 a3 e4 23 7b 5b f2 f5 7a 58 1f fc db 6d 73 c4 99 6d 40 f2 4a 3d 77 20 39 c4 ba 91 5f ae c9 8f 6f e2 14 72 1d 8a 0e a6 c6 00 36 7e 31 6e 25 e0	233 65 148 34 199 240 146 214 43 144 78 157 237 250 182 161 211 163 228 35 123 91 242 245 122 88 31 252 219 109 115 196 153 109 64 242 74 61 119 32 57 196 186 145 95 95 174 201 143 111 226 20 114 29 138 14 166 198 0 54 126 49 110 37 224

The encryption process can be done after the message digest has been converted into ASCII code. The public key pair $(p, g, y) = (277, 11, 199)$ and the private key $x = 257$ are used for message encryption, with $k = 5$ and a value (a, b) can be obtained. Then continue with the signature value of s.

Table 4. Encryption Result

Message Digest (Decimal)	233 65 148 34 199 240 146 214 43 144 78 157 237 250 182 161 211 163 228 35 123 91 242 245 122 88 31 252 219 109 115 196 153 109 64 242 74 61 119 32 57 196 186 145 95 95 174 201 143 111 226 20 114 29 138 14 166 198 0 54 126 49 110 37 224
$a = 11^5 \text{mod } 277$	144
$b_i = 199^5 . m_i \text{ mod } 277$	109 228 131 132 217 41 76 37 11 106 49 238 208 34 19 7 239 161 239 13 99 186 210 60 65 116 244 113 6 141 2 113 199 99 192 94 123 86 246 141 128 135 248 57 13 251 164 164 26 241 1 120 154 92 208 0 173 49 34 135 231 224
$r = 11^5 \text{ mod } 277$	144
$s = 5^{-1} \times (m - 257 \cdot 144) \text{ mod } 276$	259 14 115 123 76 22 113 250 99 11 69 124 56 173 82 109 28 51 224 229 232 111 191 206 203 36 102 37 139 187 17 160 91 187 220 191 83 87 229 44 69 160 30 96 147 246 33 164 202 68 6 9 251 102 198 172 232 125 110 1 59 66 50 203

3.3 Verification Process

In the verification process, calculations were carried out using the ElGamal algorithm decryption process by paying attention to the signature value of the encryption value (a, b) and the signature value (r, s) that were previously generated.

Table 5. Digital Signature Verification Result

$v_1 = 199^r \cdot r^s \text{ mod } 277$	146 256 258 70 224 264 109 31 276 40 208 185 123 44 197 188 16 270 268 200 17 138 26 38 84 186 185167 61 8 83 50 72 8 192 26 238 252 251 19 22 50 252 15 188 108 243 29 138 33 200 51 96 19 237 43 148 1 248 8 110 98 48 116
Message Signature (m)	233 65 148 34 199 240 146 214 43 144 78 157 237 250 182 161 211 163 228 35 123 91 242 245 122 88 31 252 219 109 115 196 153 109 64 242 74 61 119 32 57 196 186 145 95 95 174 201 143 111 226 20 114 29 138 14 166 198 0 54 126 49 110 37 224
$v_2 = 11^m \text{ mod } 277$	146 256 258 70 224 264 109 31 276 40 208 185 123 44 197 188 16 270 268 200 17 138 26 38 84 186 185167 61 8 83 50 72 8 192 26 238 252 251 19 22 50 252 15 188 108 243 29 138 33 200 51 96 19 237 43 148 1 248 8 110 98 48 116
Information $v_1 = v_2$	VALID

3.4 Digital Signature Testing System

Testing application for the implementation of the SHA-512 hash function and the ElGamal cryptographic algorithm to perform digital signatures on documents. Testing will be carried out on the results of the digital signature produced. The following is a digital signature application on the document using the SHA-512 hash function and the ElGamal cryptographic algorithm. Fig 3. presents the main page of the PDF Digital Signature App, an application that implements the SHA-512 and ElGamal algorithms.

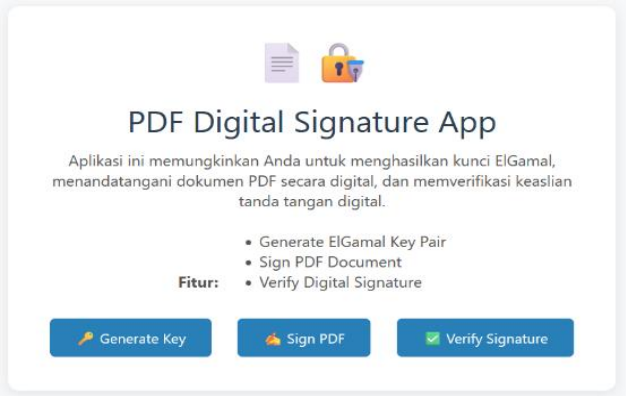

Fig 3. Home Page of PDF Digital Signature App

In the PDF Digital Signature App, users can access menus that include Generate Key, PDF Sign, and Verify Signature options. Here is the display and testing carried out on each menu.

1) Generate Key Menu

   The Generate Key menu page contains a form to input the number required by the ElGamal algorithm for key generation. Additionally, a password input form is available to limit access to digitally signed documents that have been opened. The generate key menu generates a public key (p, g, y) and private key (x) pairs that will be used in the PDF sign menu and the verify signature menu. It also indicates the time required to complete the process on this menu.

2) Sign PDF Menu

   The PDF sign menu page contains forms for uploading documents to be signed, as well as forms for the private key and public key. The PDF sign menu produces documents that have been digitally enclosed, which are automatically saved in the output folder and display the time required during the signing process. In addition to making the output document, the signature value (r, s) that will be used in the verification process is also displayed, as well as the time required in this signing process.

3) Verify Signature Menu

   The verify signature menu page contains a form for uploading documents that have been affixed with a digital signature, a document password form, a public key form, and a signature value form. The Verify Signature menu generates the statement "Signature is VALID" for documents that are proven to match the value of the generated signature, and "Signature is NOT VALID" for documents that do not match the value of the generated signature. The menu to open the document appears after the caption "Signature is VALID"; the file will then automatically open in a new tab of the browser. The time required for the verification process is also presented.

4) Application Testing

   Application testing conducted for validation purposes uses the black box testing method, which helps verify whether each component created on the system has performed as expected. Table 6 presents the black box testing method used to test the application of the ElGamal Digital Signature App.

Table 6. ElGamal Digital Signature App Test Result

Test Items	Test Scenarios	Expected Result	Test Result
The Generate Key button	Enter the prime number p, the primitive root g, and a random number x.	Entering the prime number p, the primitive root g, and the random number x, the system automatically calculates the value of y. The result is a public key pair (p, g, y) and private key x, also presented with the time of the system uses.	Succeed
The Brows button	Pressing the browse button and uploading PDF documents during the signing process, and PDF documents at the time of verification	The document (*.pdf) to be uploaded is as displayed in the "Sign PDF Document" menu and the "Verify Digital Signature" menu after pressing the Browse button and selecting the desired document.	Succeed
The Sign PDF button	Pressing the Sign PDF button	The digital signature will be displayed when the "Sign PDF" button is pressed after entering the private key, public key, and the document to be signed. The time required for the signing process will also be displayed.	Succeed
The Verify Signature button	Pressing the Verify Signature button	The caption "Signature is VALID" or "Signature is NOT valid" will be displayed when you press the Verify Signature button after entering the public key and the value of the generated signature. The time it takes for the verification process will also be displayed.	Succeed

4 Conclusion

Based on this research, it can be concluded that:

1) The implementation of ElGamal's cryptographic algorithm in creating digital signatures is applied to the key generation process, encryption process, and decryption process. Meanwhile, the SHA-512 hash function is applied to calculate the hash value, also known as a message digest, which will be used in the process of creating and verifying a digital signature.
2) The process of encrypting messages in the form of PDF documents using the SHA-512 hash function and the ElGamal cryptographic algorithm produces new PDF documents that have gone through the signing process, and the signature value will be used in the verification process.
3) The process of decrypting messages in the form of PDF documents that have gone through the signing process using the ElGamal cryptographic algorithm produces information to prove the validity or invalidity of the signature.
4) The message digest and hash value of the SHA-512 hash function are used to verify the signature generated from the signing process on a PDF document.
5) Securing documents with digital signatures using the SHA-512 hash function and the ElGamal cryptographic algorithm results in a new PDF document that has been affixed. The document will be verified for authenticity based on the signature value generated and the public key used.
6) The SHA-512 hash function and ElGamal's cryptographic algorithm can be used simultaneously to generate digital signatures that are very useful for securing PDF documents.
7) The digital signature generated using a combination of two algorithms, namely the SHA-512 hash function and the ElGamal cryptographic algorithm, is very effective for securing PDF documents. The discrete algorithm's calculation of the prime modulo number in the ElGamal algorithm, as well as the

output of fixed-length messages resulting from the SHA-512 hash function, is longer than other SHA families, thus improving the security of the document messaging process.

References

[1] D. P. Precilia dan A. Izzuddin, "Digital Signature Application Using Message Digest Algorithm 5 (MD5)," *Energy*, vol. 5, no. 1, pp. 14–19, 2015.

[2] R. A. Azdy, "Digital signatures using keccak algorithms and RSA," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 5, no. 3, pp. 184–191, 2016.

[3] G. Nguyen *dkk.*, "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 77–124, 2019.

[4] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. theory*, vol. 31, no. 4, pp. 469–472, 1985.

[5] F. S. Octora Ginting, Veithzal Rivai Zainal, dan Aziz Hakim, "Digital Signature Standard Implementation Strategy by Optimizing Hash Functions Through Performance Optimization," *J. Account. Financ. Manag.*, vol. 3, no. 6, pp. 362–371, 2023, doi: 10.38035/jafm.v3i6.175.

[6] D. Y. Islami, "Implementasi Algoritma ElGamal dan Fungsi Hash SHA3 untuk Tanda Tangan Digital pada Audio," 2019.

[7] F. Kpieleh, "Cryptographic Hash Functions For Digital Stamping," *Adv. Multidiscip. Sci. Res. J. Publ.*, vol. 10, no. 4, pp. 65–72, 2022, doi: 10.22624/aims/digital/v10n4p9.

[8] B. U. I. Khan, R. F. Olanrewaju, M. A. Morshidi, R. N. Mir, M. L. B. M. Kiah, dan A. M. Khan, "Evolution and Analysis of Secured Hash Algorithm (Sha) Family," *Malaysian J. Comput. Sci.*, vol. 35, no. 3, pp. 179–200, 2022, doi: 10.22452/mjcs.vol35no3.1.

[9] H. N. Bhonge, M. K. Ambat, dan B. R. Chandavarkar, "An experimental evaluation of sha-512 for different modes of operation," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–6.

[10] Agusniar, Cut., Fazira, Ira., Wahyunita, Laili., (2024). Implementation of the Secure Hashing Algorithm-512 (SHA-512) for Sign-Up Page Security in th Kelas Seru Tutoring System. Journal of Advanced Computer Knowledge and Algorithms: 2(1), 19-23.

[11] Sinlae, Alfry A. J., (2012). Cryptosystem Analysis Using Digital Signatures Based on SHA-512 and RSA Algorithms, Salatiga.

[12] M. Yang et.al., "A SHA-512 hardware implementation based on block RAM storage structure," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2022, pp. 132–135.

[13] J. Suganya dan K. J. J. Kumar, "Cellular Automata based SHA-512 Hashing Algorithm," *Int. J. Eng. Res. Technol.*, vol. 5, no. 17, pp. 5–7, 2018.

[14] H. Cheng, D. Dinu, dan J. Großschädl, "Efficient implementation of the SHA-512 hash function for 8-bit AVR microcontrollers," in *International Conference on Security for Information Technology and Communications*, 2018, pp. 273–287.

[15] Y. Kong, F. Wu, L. Gao, dan Y. Wu, "FBC: a flat binary code scheme for fast Manhattan hash retrieval," in *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*, 2018, vol. 10615, pp. 1359–1367.

[16] A.-T. Hoang, K. Yamazaki, dan S. Oyanagi, "Pipelining a multi-mode SHA-384/512 core with high area performance rate," *IEICE Trans. Inf. Syst.*, vol. 92, no. 10, pp. 2034–2042, 2009.

[17] E. H. Omran dan R. J. S. Al-Janabi, "Modified elgamal algorithm using three paring functions," in *Next Generation of Internet of Things: Proceedings of ICNGIoT 2022*, Springer, 2022, pp. 405–413.

[18] B. Umapathy dan G. Kalpana, "A Key Generation Algorithm for Cryptographic Algorithms to Improve Key Complexity and Efficiency," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2023, pp. 647–652