Jurnal Riset Mahasiswa Matematika
Volume 5 (2) (2025), Pages 176-188
p-ISSN: 2808-1552; e-ISSN: 2808-4926

Code-Based Document Authentication Using SHA-3 and McEliece

Vadillatul Ni'ma Maulidya* and Muhammad Khudzaifah

Department of Mathematics, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia

Abstract

Personal data protection is essential to ensure the security and authenticity of digital documents. One approach to achieving document authenticity is through a code-based document authentication mechanism. This study proposes a document authentication scheme using the SHA-3 hash function combined with the McEliece code-based cryptosystem with Hamming code. The authentication process begins by hashing the document content using SHA-3 to generate a message digest. The resulting hash value is converted into binary form and then encoded and encrypted using the public key of the McEliece cryptosystem, producing an authentication code. During the verification phase, the received authentication code is processed through the decoding mechanism using the private key, allowing error correction and recovery of the original hash representation. A document is considered authentic if the decoded hash matches the hash value generated from the received document. Experimental results show that the authentication code depends on the McEliece key pair used, ensuring uniqueness and resistance to forgery. Furthermore, the combination of SHA-3 and the McEliece cryptosystem exhibits an Avalanche Effect close to 50%, indicating strong diffusion properties. These results demonstrate that the proposed code-based authentication scheme provides a reliable cryptographic solution for ensuring the authenticity and protection of digital documents, particularly against future cryptanalytic threats.

Keywords: SHA-3 algorithm, McEliece cryptosystem, code-based authentication, digital document, message digest.

1 Introduction

Code-based document authentication is a cryptographic mechanism used to verify the authenticity and integrity of digital documents. Unlike conventional authentication methods, this mechanism employs mathematical algorithms to generate an authentication code that is directly dependent on the content of the document [1]. Any modification to the document will result in a different authentication code, enabling the detection of unauthorized changes. Various cryptographic hash algorithms, such as SHA and MD5, have been utilized in document authentication systems. However, SHA is considered superior in terms of efficiency and security due to its higher resistance to cryptographic attacks compared to MD5 [2].

Secure Hash Algorithm (SHA) is a cryptographic hash function widely used to ensure data integrity in document authentication mechanisms. SHA operates as a one-way function that transforms input data of arbitrary length into a fixed-length message digest. Several variants of

*Corresponding author. E-mail: vadillatul1005@gmail.com

Submitted: November 21, 2025 Reviewed: November 30, 2025 Accepted: December 31, 2025
DOI: https://doi.org/10.18860/jrmm.v5i2.37652

SHA have been developed, including SHA-0, SHA-1, SHA-2, and SHA-3, each offering different levels of security [3]. In this study, SHA-3 is selected due to its enhanced security features and resistance to cryptanalytic attacks, as well as its support for flexible output sizes such as 224-bit, 256-bit, 384-bit, and 512-bit [4].

In addition to employing SHA-3 as a hash function, the McEliece cryptosystem is applied to strengthen the document authentication process through a code-based approach. McEliece is a public-key cryptographic algorithm based on error-correcting codes that utilizes binary matrices for encoding and decoding processes [5]. Although it requires a larger key size compared to RSA (Rivest–Shamir–Adleman) and ECC (Elliptic Curve Cryptography), McEliece offers strong resistance to quantum computing attacks, making it a promising candidate for post-quantum cryptographic systems [6]. This study implements Hamming codes within the McEliece framework, which are capable of detecting and correcting bit errors using parity-based principles. The integration of Hamming codes aims to enhance the robustness and reliability of the authentication mechanism by ensuring accurate recovery of hash values during the decoding process.

2 Research Methods

2.1 Cryptography

Cryptography originates from the Greek words *cryptos* (secret) and *graphein* (writing), meaning the art and science of securing information. According to Bruce Schneier, cryptography is a method used to protect information so that it can only be accessed by authorized parties [7]. Core objectives of cryptography include confidentiality, data integrity, authentication, and non-repudiation [8]. In practice, cryptographic systems apply mathematical transformations to data in order to ensure that information cannot be altered or forged without detection [9].

2.2 McEliece Algorithm

The McEliece algorithm is an asymmetric cryptographic scheme introduced by Robert J. McEliece in 1978 and is designed to provide resistance against quantum computing attacks. This algorithm is based on error-correcting codes, with simplified implementations commonly using Hamming codes capable of detecting and correcting bit errors during data transmission [10]. Unlike RSA or ECC, the McEliece cryptosystem relies on encoding and decoding processes, not on matrix inversion, to recover plaintext data [11] [12]:

1. The private key consists of the scrambler matrix S, the generator matrix G, and the permutation matrix M
2. The public key matrix G' is generated using the equation

$$G'_{k\times n} = S_{k\times k} \cdot G_{k\times n} \cdot M_{n\times n}$$

The encoding (encryption) process using the public key is defined as:

1. The message m is converted into binary blocks. If the message length is less than k, padding is applied.
2. A random error vector e is added.
3. The encoded ciphertext c is computed as

$$c_{n-bit} = m_{k-bit} \cdot G'_{k\times n} + e_{n-bit}$$

The decoding (decryption) process using the private key proceeds as follows:

1. The permutation effect is removed:

$$c'_{n-bit} = c_{n-bit} \cdot M^{-1}_{n\times n}$$

2. The resulting codeword c' is decoded using the Hamming decoding algorithm to correct errors.
3. The original message m is recovered through the decoding process, without applying a matrix inverse of G', since such an inverse is not defined for non-square matrices

2.3 Code-Based Document Authentication Mechanism

Code-based document authentication is a cryptographic mechanism used to verify the authenticity and integrity of digital documents Figure 1. Unlike classical digital signature schemes that rely on private-key encryption, this mechanism employs cryptographic hash functions combined with code-based cryptography to generate and verify authentication codes [13].

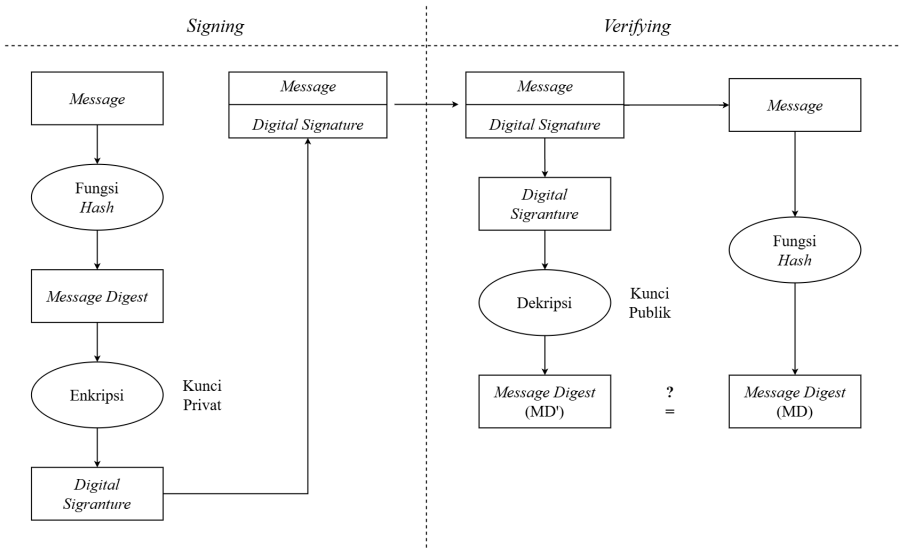

Figure 1: Illustrates the General Authentication Scheme

The document content is first transformed into a fixed-length hash value using SHA-3. This hash is then processed using a McEliece-based encoding procedure to produce an authentication code. During verification, the authentication code is decoded using the corresponding private key and compared with the recomputed hash value of the received document. If both values match, the document is considered authentic and unchanged [14].

2.4 SHA-3 Algorithm

The National Institute of Standards and Technology (NIST) organized the NIST Hash Function Competition in 2006 to establish a new cryptographic hash standard. In 2012, the Keccak algorithm was selected as the winner and later standardized as SHA-3 through FIPS 202 in 2014, with formal adoption in 2015 [15]. SHA-3 supports output sizes of 224, 256, 384, and 512 bits and was introduced as an alternative to SHA-2 to anticipate potential vulnerabilities, considering historical attacks on MD5, SHA-0, and SHA-1 [16]. SHA-3 is based on the sponge construction, which consists of two main phases in Figure 2:
Absorbing phase:

1. The message is converted to byte format (UTF-8), padded, and divided into P_i blocks of r-bit size.
2. Each block is XOR with the first r-bit of the internal state S.
3. A permutation function f is applied to update the state.
4. This process is repeated until all blocks are complete.

Squeezing phase:

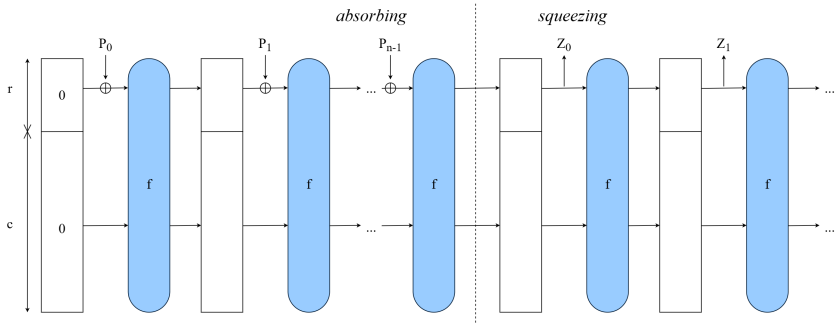

Figure 2: SHA-3 Sponge Construction

1. The output is stored in variable Z.
2. The first r-bit of the state is appended to Z until the desired output length d is reached.
3. If necessary, the permutation function f is reapplied.
4. The final value of Z represents the message digest.

2.5 McEliece-Based Code Authentication Using SHA-3

The proposed document authentication scheme integrates SHA-3 and the McEliece cryptosystem using Hamming codes with parameters (n, k). The process consists of key generation, authentication code generation, and verification [17]:
Key generation:

1. Public key matrix consists of parity-check matrix H, the scrambler matrix S, and the decoding algorithm of the Hamming code.
2. The public key is defined as the transformed parity-check matrix

$$H'_{n\times(n-k)} = H^T_{n\times(n-k)} \cdot S_{(n-k)\times(n-k)}$$

Authentication code generation:

1. The document message m is hashed using SHA-3 and converted into binary form.
2. If the binary length is less than n, padding is applied.
3. A controlled noise vector h is added.
4. The authentication code c is computed as

$$c_{(n-k)-bit} = m_{n-bit} \cdot H'_{n\times(n-k)} + h_{(n-k)-bit}$$

Verification process:

1. The received authentication code is transformed by removing the scrambling effect

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

2. Decoding the document

$$\begin{aligned} c' = cS^{-1} &= (m \cdot H' + h)S^{-1} \\ &= mH'S^{-1} + hS^{-1} \\ cS^{-1} &= mH^T SS^{-1} + hS^{-1} \\ c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)} &= m_{n-bit} \cdot H^T_{n\times(n-k)} + h_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)} \end{aligned}$$

If the result of decrypting the ciphertext c' is the same as decoding the document, then the document is considered valid.

3 Results and Discussion

3.1 Key Generation

The key used is the key with the parameter $(7, 3)$, as shown in Figure 3.

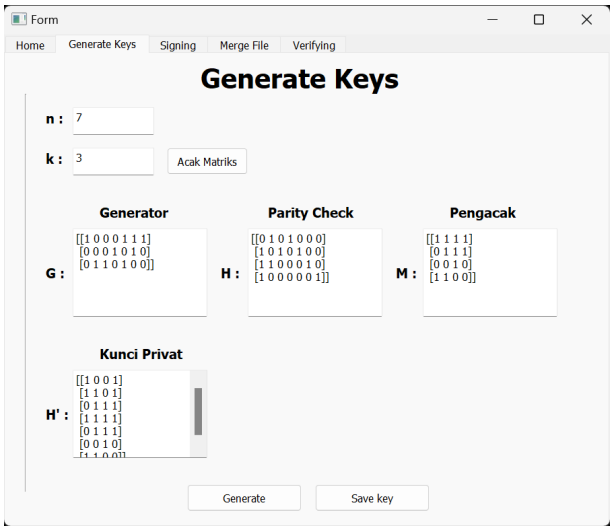

Figure 3: Key Generation

Based on the figure above, the public key matrix and the private key can be written, as follows:

1. Generator matrix (G) of size $(k \times n)$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

2. Parity-check matrix (H) of G size $((n-k) \times n)$

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Randomizer matrix (S) of size $(n-k) \times (n-k)$

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

4. Private key matrix (H') of size $(n \times (n-k))$

$$H' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

3.2 Document Authentication Code Generation

The hashing process applied to the document produces a message digest in the form of hexadecimal values, as shown in Figure 4.

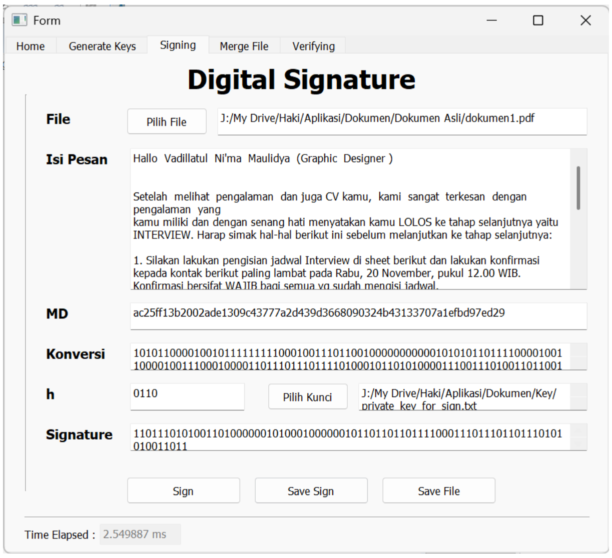

Figure 4: Document Authentication Result

The document produces a message digest as follows:

$$ac25ff13b2002ade1309c43777a2d439d3668090324b43133707a1efbd97ed29$$

The message digest is then converted into binary form as follows

$$1010110000100101111111111000100111011001000000000000101010110111100001001100001$$

$$0011100010000110111011101111010001011010100001110011101001101100110100000001 0$$

$$010000000110010010010110100001100010011001101110000011110100001111011111011110$$

$$1100101111110110100101001$$

After conversion, the binary message digest is divided into several blocks P_i, each with a length of n-bit (or 7-bit depending on the selected parameter). These blocks are then encoded using the McEliece-based code authentication scheme by applying the transformed parity-check matrix H' and adding a controlled noise vector. This process produces an authentication code rather than a conventional digital signature. The following result is obtained:

$$00011001011001011001110010101010100011010010010110110000100001010011000011010$$

$$101011111000011100101100010011001110111101111101100011100101110101101100$$

To facilitate practical verification and improve readability by the authentication system, the resulting authentication code is converted into a QR Code representation, as shown in Figure 5.

Figure 5: QR Code Document Authentication Code

For comparison, experiments were conducted using the same document with different McEliece parameter sets to observe variations in the generated authentication codes. The results are presented in Table 1.

Table 1: Authentication Results Using the Same Document

Message Digest	Parameter	Digital Signature
ac25ff13b2002ade1309c 4377a2d439d3668090324 b43133707a1efbd97ed29	(7,3)	10101010001110011011011011111101100 00110101100110100000000000110110111 101100
ac25ff13b2002ade1309c 4377a2d439d3668090324 b43133707a1efbd97ed29	(13,9)	00011010110010110011100110101001010001 10100100101101000000010101000100011 01010101111000011001110010110000101 1 01111111110110001110011110110110100

The experimental results show that different McEliece parameters generate different authentication codes for the same document hash. This demonstrates that the proposed system is sensitive to key variations and capable of distinguishing authentication results based on the applied code parameters.

3.3 Document Verification

The document verification process consists of two sequential stages, as illustrated in Figure 6.

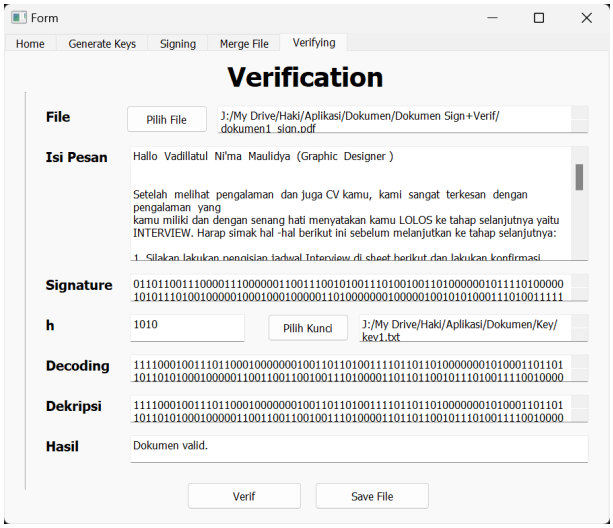

Figure 6: Document Verification Results

Based on testing using the developed program, the first stage of verification involves processing the received authentication code. The authentication code is divided into several blocks C_i of $(n-k)$-bit length (or 4-bit depending on the parameter) and is transformed by removing the scrambling effect using the inverse of the scrambler matrix S. This step prepares the data for decoding and does not represent a decryption process in the conventional public-key cryptography sense. The following result is obtained:

$$101101111111011001110001111011101100101000100110010100001100011010010000 1010$$

$$011001001100010000100001110100010011100001001010110000110110101011110001$$

The second stage of verification is decoding the transformed data using the Hamming decoding algorithm associated with the parity-check matrix H^T. This decoding process corrects possible

bit errors introduced during authentication code generation and recovers the hash representation. Based on program testing, the following decoding result is obtained:

101101111111011001110001111011101100101000100110010100001100011010010000101 0

011001001100010000100001110100010011100001001010110000110110101011110001

By comparing the decoded result with the recomputed SHA-3 hash of the received document, it is observed that both values are identical. Therefore, the document is declared authentic and unaltered.

Digital authentication codes generated using different keys, as obtained previously, are processed during the verification stage to compare the document authentication results. The verification results using the same document but different keys are presented in Table 2.

Table 2: Verifying Results Using the Same Document

Name of Document	Calculation	Result
doc1_sign_key1.pdf	**Decryption:** 10110111111101100111000111101110110010 01000100110010100001100011010010000101 00110010011000100001000011101000100 11000010010101100001101101010111100 01 **Decoding:** 10110111111101100111000111101110110010 01000100110010100001100011010010000101 00110010011000100001000011101000100 11000010010101100001101101010111100 01	Valid
doc1_sign_key2.pdf	**Decryption:** 10111011111010110100001110001000110110 00110110110001000000000100101011111000 10110 **Decoding:** 10111011111010110100001110001000110110 00110110110001000000000100101011111000 10110	Valid

The test results using two different keys indicate that the proposed system is capable of distinguishing verification outcomes based on the applied key variations. Differences in keys influence the encoding results and consequently produce different authentication codes. However, when the correct corresponding keys are used during verification, the decoding process successfully recovers the original hash representation, and the document is declared valid. Furthermore, even minor modifications to the document content, such as deletion, addition, or alteration of characters, can be accurately detected by the system. This demonstrates that the hashing process is highly sensitive to input changes and effective in maintaining the integrity and authenticity of digital documents.

3.4 Security and Efficiency Evaluation

The Avalanche Effect is an important metric in cryptographic systems that measures the degree of output change caused by a small variation in the input. Ideally, a one-bit change in

Name of Document	Calculation	Result
doc1_sign_delete.pdf	**Dekripsi:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 ***Decoding*:** 00100110011101001010011101011001001010101111000000111011111011010110110111101001	Invalid
doc1_sign_change.pdf	**Decryption:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 **Decoding:** 01110110100100010010110111011010100111101010001110000010001001000100100001101111	Invalid
doc1_sign_add.pdf	**Decryption:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 **Decoding:** 00100010100000010101100110110101011101100000101101101100111100100010000000101111	Invalid

the input should result in approximately 50% of the output bits changing, indicating strong diffusion characteristics [18]. In this study, the Avalanche Effect of the SHA-3 algorithm is evaluated using two approaches: (1) SHA-3 without the McEliece scheme, to measure the intrinsic diffusion property of the hash function by comparing binary hash values of the original input and a one-bit-modified input; and (2) SHA-3 combined with the McEliece-based authentication mechanism, to evaluate the additional impact on the sensitivity of the generated authentication code. The percentage of the Avalanche Effect is calculated using the following formula:

$$Avalanche\ Effect\ (\%) = \left(\frac{number\ of\ bits\ changed}{total\ output\ bits}\right) \times 100\%$$

Based on the results presented in Table 3, the combination of the SHA-3 and McEliece algorithms produces significant changes in the authentication code output even when only minor modifications are made to the input document. The results also show that incorporating the McEliece scheme increases the percentage of bit changes, indicating enhanced diffusion and greater sensitivity to both input variations and key differences.

In addition to security evaluation, processing time efficiency is also analyzed to assess the computational impact of incorporating the McEliece algorithm. This evaluation compares the processing times of authentication code generation and verification for various document sizes, focusing on the trade-off between enhanced security and computational cost, as shown in Table 4.

Table 4: Processing Time Test Results

Name of Document	**File Size (kb)**	**Encryption without McEliece**	**Encryption with McEliece**	File **Size (kb)**	**Decryption without McEliece**	**Decryption with McEliece**
file 1.pdf	12	1×10^{-6}	1.92715	86	0.41170	3.60128
file 2.pdf	63	0.47225	2.20701	118	0.82473	4.10519
file 3.pdf	154	0.60516	2.52096	246	0.90654	4.12660
file 4.pdf	1,315	0.63245	2.64007	1,331	1.06404	4.62662
file 5.pdf	5,614	0.71921	3.24225	5,484	1.78476	5.47170
file 6.pdf	11,231	1.51627	3.89267	10,906	3.23987	6.49738

The experimental results indicate that the McEliece algorithm significantly increases processing time during both authentication code generation and verification for small and large documents. While this additional computational overhead reduces time efficiency, it provides improved security characteristics, particularly in terms of resistance to forgery and sensitivity to data and key variations.

4 Conclusion

Based on the experimental results, the following conclusions can be drawn:

1. Each document generates a distinct authentication code, even when using the same key parameters, as the result is directly dependent on the hash value of the document content. Furthermore, variations in McEliece key parameters produce different authentication codes for identical document content, demonstrating key-dependent behavior.
2. The document verification process is able to accurately detect changes in document content. Unmodified documents produce identical hash values and decoded authentication results, while even minor modifications result in different hash values. This confirms the high sensitivity of the proposed system in maintaining document integrity and authenticity.
3. 3. The combination of the SHA-3 hash function and the McEliece cryptosystem exhibits a strong Avalanche Effect, where a one-bit change in the input affects nearly 50% of the output bits. However, the computational complexity introduced by the McEliece algorithm increases processing time, indicating a trade-off between enhanced security and efficiency, particularly for large documents.

Overall, the integration of SHA-3 and the McEliece cryptosystem using Hamming codes is effective in detecting document alterations and strengthening document authentication through a code-based approach. While the method improves security characteristics such as sensitivity to input and key variations, efficiency considerations remain important for practical implementation in large-scale or time-critical applications.

CRediT Authorship Contribution Statement

Vadillatul Ni'ma Maulidya: Conceptualization, Methodology Design, Research Document Collection, Writing – Initial Draft. **Muhammad Khudzaifah:** Supervisor I, Writing – Review and Editing.

Declaration of Generative AI and AI-assisted technologies

The ChatGPT version 4 model was used to assist in program development, initial draft preparation, and sentence structure correction.

Declaration of Competing Interest

The authors declare no competing interest.

Funding and Acknowledgments

This research did not receive funding from any external sources. The authors would like to express their gratitude to all parties who provided support and facilities throughout the research process. Special thanks are extended to Supervisor I, Muhammad Khudzaifah, M. Si., for their valuable assistance and guidance. The authors also express their appreciation to their families and friends for the motivation and support that served as a source of strength.

Data Availability

The research data were analyzed in the form of digital documents containing alphabets, numbers, and characters obtained from the author's personal documents.

References

[1] A. Lorien and T. Wellem, "Implementasi sistem otentikasi dokumen berbasis quick response QR code dan digital signature," *Rekayasa Sistem dan Teknologi Informasi RESTI*, vol. 5, no. 4, pp. 663–672, 2021. DOI: 10.29207/resti.v5i1.3316.

[2] M. H. Santoso et al., "Perbandingan algoritma kriptografi hash MD5 dan SHA-1," *Semantika*, vol. 2, no. 1, pp. 54–59, 2019.

[3] M. P. Sari, "Analisis algoritma SHA-3 keamanan pada data pribadi," *Tecnoscienza*, vol. 5, no. 2, pp. 231–242, 2021.

[4] F. Kurniawan, A. Kusyanti, and H. Nurwarsito, "Analisis dan implementasi algoritma SHA-1 dan SHA-3 pada sistem autentikasi garuda training cost," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 1, no. 9, pp. 803–812, 2017.

[5] N. F. Ilmiyah, "Kajian tentang kriptosistem mceliece dalam menghadapi tantangan komputer kuantum di era revolusi industri 4.0," *Prosiding Seminar Nasional MIPA*, pp. 216–226, 2018.

[6] M. Kumar, "Post-quantum cryptography algorithm's standardization and performance analysis," *Array*, vol. 15, 2022. DOI: 10.1016/j.array.2022.100242.

[7] Nurhayati, Kastari, and F. Fahrianto, "End-to-end encryption on the instant messaging application based android using AES cryptography algorithm to a text message," *CITSM*, 2022. DOI: 10.1109/CITSM56380.2022.993596.

[8] R. Munir, *Kriptografi*. Institut Teknologi Bandung, 2019.

[9] Basri, "Kriptografi simetris dan asimetris dalam perspektif keamanan data dan kompleksitas komputasi," *Jurnal Ilmiah Ilmu Komputer*, vol. 2, p. 2, 2016.

[10] K. J. Jaameri, "Code-based cryptography," *School of Science*, 2019.

[11] M. Baldi, M. Bianchi, and F. Chiaraluce, "Enhanced public key security for the mceliece cryptosystem," *Journal of Cryptology*, vol. 29, 2016. DOI: 10.1007/s00145-014-9187-8.

[12] S. Sinurat and E. R. Siagan, "Learning text data security in documents using mceliece's algorithm," *Jurnal Infokum*, vol. 10, p. 5, 2022.

[13] M. H. Harianja, "Analisa fungsi hash untuk mendeteksi otentikasi file video menerapkan metode n-hash," *Management of Information System Journal*, vol. 2, p. 1, 2023.

[14] E. C. Prabowo and I. Afrianto, "Penerapan digital signature dan kriptografi pada otentikasi sertifikat tanah digital," *Jurnal Ilmiah Komputer dan Informatika*, vol. 6, p. 2, 2017. DOI: 10.34010/komputa.v6i2.2481.

[15] I. Rahim et al., "Komparasi fungsi hash md5 dan sha256 dalam keamanan gambar dan teks," *Ikraith-Informatika*, vol. 7, p. 2, 2023.

[16] C. H. Romine, "Sha-3 standard: Permutation-based hash and extendable-output functions," *Federal Information Processing Standards Publications*, 2015.

[17] A. Alahmadi, S. Calkavur, et al., "A new code based signature scheme for blockchain technology," *Mathematics*, vol. 11, p. 1177, 2023. DOI: 10.3390/math11051177.

[18] D. Upadhyay, N. Gaikwad, et al., "Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications," *IEE Access*, vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3215778.

Table 3: Avalanche Effect Test Results

Test	**Original Input**	**Modified Input**	**Digital Signature**	**Different Bits**	**Persentage**
Without McEliece addition	Input123	Input12	**Original:** 10010111001100110010010100000 1 1011110101101110110011010001 01 0101011100011011010111000000 10 0011010110111101010110000110 01 1110111001101101100100100001 00 1000101101101001000111000101 00 0011010111001111110001100011 11 0000011101110110011010111011 00 0100000111000110 **Modified:** 0100100000101100001101000010110 0111011100010010110000010101 11 1010100110001000011000000000 01 0111000010010001101000100110 00 0111000000011010100100010111 01 0110001001001001001111001001 00 1100011101110101111101101011 01 1101011110011101011111011111 01 1000101110110000	125/256 bits	48.83%
With McEliece addition	Input123	Input12	**Original:** 0010101110010100100100011100 01 1110101100010111001101101111 10 0001001010000111011111011100 01 0100001111010011110101000011 00 01000000011100010101010100 00 **Modified:** 0000100011100001100101011000 00 0101000010110111101000100111 00 0001101101010100010010110001 11 1100001010001100000010100000 11 11110111000100101100000110 10	72/148 bits	48.65%