Evaluation of Digital Signatures Using RSA and HMAC Algorithms

Parinya Natho
Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi, Thailand
parinya.n@rmutsb.ac.th

Salinun Boonmee
Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi, Thailand
salinun.b@rmutsb.ac.th

Natthinee Khongkrairoek
Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi, Thailand
nattinee.k@rmutsb.ac.th (corresponding author)

Received: 18 August 2025 | Revised: 17 September 2025 and 21 September 2025 and | Accepted: 24 September 2025

 DOI: https://doi.org/10.48084/etasr.14149

ABSTRACT

Nowadays, communication can be conducted at high speed and conveniently through computer networks. In addition, devices in network systems can store data for extended periods. However, communication or storage over the network may lead to security issues or loss of data integrity. Therefore, data security mechanisms should be used to verify and maintain confidentiality and integrity. This research aims to evaluate which algorithm can verify data integrity and identify message authors in the least amount of time. The algorithms studied are Rivest–Shamir–Adleman (RSA), Hash-based Message Authentication Code–Secure Hash Algorithm 256 (HMAC-SHA256), HMAC-SHA384, and HMAC-SHA512, all of which verify data integrity and authenticate message authors. We compared the performance of all algorithms using ten different data sizes ranging from 100 to 1000 MB, with twenty datasets for each size, and evaluated their throughput and bandwidth. The results show that the HMAC-SHA256 algorithm requires the least execution time and therefore demonstrates the highest efficiency. In addition, the algorithm becomes more efficient as the data size increases, followed by HMAC-SHA512, HMAC-SHA384, and RSA, respectively. In future work, studying other algorithms that are capable of verifying data integrity and sender authenticity, such as digital signatures using RSA, Elliptic Curve Cryptography (ECC), and Galois Message Authentication Code (GMAC), or developing applications using HMAC-SHA256 for data integrity and sender authentication, will provide further insights into selecting the most appropriate algorithm for specific applications.

Keywords-digital signature; Rivest–Shamir–Adleman (RSA); Secure Hash Algorithm (SHA); Hash-based Message Authentication Code (HMAC)

I. INTRODUCTION

A. Background and Significance

At present, with the proliferation of transferring and storing large volumes of data via electronic means, the safety and reliability of data concerns everyone. A widely adopted solution to this challenge is the digital signature. Digital signatures are indispensable for certifying the source and integrity of the information transmitted in an electronic document or message. They play a critical role in securing and guaranteeing the reliability of electronic documents. Digital signatures also have legal attributes. Currently, the primary legal attribute is the writing attribute, whereas the two techno-legislative attributes are the certificate authorities and the submission format. The electronic signature ensures a standardized digital representation and accuracy of information [1]. What makes digital signatures special and gives them value is that they provide three critical security features simultaneously: authentication, integrity, and non-repudiation. Non-repudiation is particularly important in legal or business functions because it certifies that an author signed a document and has not retracted their agreement.

In the digital domain, a digital signature begins with a user taking a message and processing it through a hashing algorithm. The hash value is then encrypted asymmetrically with the sender's private key. The resulting encrypted value is the signature, which travels with the message to the recipient. The recipient is able to verify the signature by hashing the message and then decrypting the signature with the sender's public key. The resulting hash values are then compared to confirm authenticity and integrity [2-5].

The Rivest–Shamir–Adleman (RSA) algorithm works by utilizing the mathematical properties of large prime numbers and their factorization. The security of RSA relies on the difficulty of factoring large composite numbers, ensuring that only the intended recipient can decrypt the encrypted data [6]. RSA is a widely used public-key cryptosystem in secure communication protocols such as HTTPS, SSL/TLS, and SSH to establish a secure channel between two parties. The algorithm comprises three steps: generation of keys (public-private key pairs), encryption (public key), and decryption (private key) [7]. RSA security relies on complex mathematical computations, though this computational burden affects performance in resource-constrained environments.

Unlike the asymmetric RSA mechanism, Hash-based Message Authentication Code (HMAC) is a symmetric algorithm designed solely for the purpose of message authentication and integrity [8-11]. HMAC is a fusion of a cryptographic hash function, such as Secure Hash Algorithm (SHA)-1, SHA-256, or SHA-384, with a secret key to produce a fixed-size authentication code that can be created only if the user is aware of both the key and the message. The HMAC process involves the division of the original message into n-bit blocks, depending on the adopted hashing algorithm, and using an n-bit secret key. The key is subjected to XOR operations with inner and outer padding (iPad and oPad) values, and the obtained hash value is the authentication code [12]. HMAC's greatest advantage is its high speed due to shared secret keys, although this exposes it to key distribution challenges common in symmetric schemes, as shown in Figure 1.

System designers and users frequently encounter a complex balancing act between security and performance. Foundational security protocols for data protection and user trust, such as cloud storage security measures [13], secure user passwords and associated data [14], and ciphertext policy attribute-based file access [15], often add complexity that hinders user experience and system efficiency. On the other hand, emphasizing system performance can expose systems to cyber threats and compromise data integrity and data confidentiality [16, 17]. The security and performance balance is particularly evident in digital signature algorithms. RSA, for instance, is computationally heavy due to complex mathematical computations, whereas HMAC is computationally light because of the symmetric key operations. The choice of algorithm depends on the context, dictating the balance of security and performance for specific operations.

Recent research has extensively explored different aspects of digital signature algorithms and optimization. Authors in [18] emphasized the necessity of examining and adjusting digital signature algorithms due to advances in quantum computing and recommended a comparative analysis of post-quantum hash-based signature schemes. Authors in [19] compared hash-based signature schemes, specifically MSS and W-OTS algorithms, based on key generation, signature generation, and verification performance. Energy minimization has been another area of utmost significance. Authors in [8] suggested a novel approach to reduce HMAC energy usage by optimizing the underlying SHA-256 algorithm with the Energy Complexity Model and achieved energy conservation of up to 14%. This article demonstrated that deep efficiency gains are achievable by directly designing cryptographic primitives rather than focusing on optimizations at the upper system level.

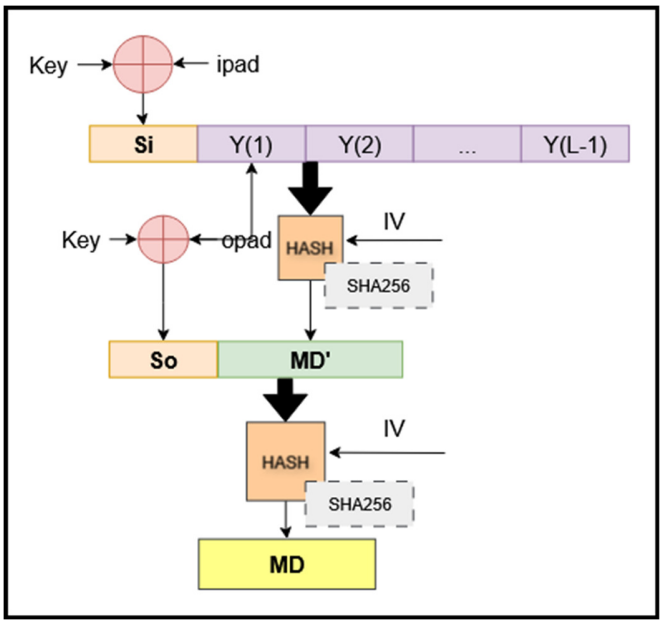

Fig. 1. Schematic of HMAC.

Performance comparisons between different algorithms have also been under the spotlight. Authors in [20] compared HMAC, RSA, ECC, and ZK-SNARK implementations according to banking industry standards, whereas authors in [14] provided in-depth comparisons of hash functions (MD5, SHA-1, SHA-2, and SHA-3) in password storage scenarios. Authors in [21] specifically used HMAC-SHA256 for file authentication, demonstrating its effectiveness in guaranteeing data integrity and preventing unauthorized modification. The continually evolving threat landscape, particularly the implications of quantum computing, have propelled the research in the direction of post-quantum cryptography. Authors in [22] provided a comprehensive review of classical and post-quantum digital signature schemes, in response to the urgent need for quantum-proof schemes. Meanwhile, authors in [23] studied homomorphic encryption techniques for secure image processing within the broader context of cutting-edge cryptographic applications.

The present research topic seeks to evaluate the performance of digital signature algorithms, specifically RSA and HMAC. Using performance metrics such as encryption speed, decryption speed, time to generate key, and memory

usage, this research provides valuable information about the disadvantages and advantages of RSA and HMAC as digital signature algorithms. This research will prove beneficial as it will shed light on how various algorithms are effective and efficient. This information will be useful to academicians, developers, and organizations in the selection and implementation of digital signature algorithms. Additionally, it may be used to create more reliable digital systems that provide safe and secure digital interactions across a range of industries, including e-commerce, online banking, and e-government.

II. METHOD

A. Dataset and System Preparation

In this section, an analysis of the performance of HMAC algorithms, including HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512, as well as the digital signature using the RSA algorithm, is presented. The researcher used a personal computer with an Intel Core i7 CPU clocked at 2.50 GHz and 40 GB of RAM, and installed the Ubuntu operating system and OpenSSL on Oracle VirtualBox software.

The algorithms were evaluated based on the time taken to process different data sizes. The researcher defined ten data sizes for testing: 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 MB, which are typical sizes used in communication, such as messaging apps, e-commerce, or cloud storage. Each size was randomly sampled into 20 datasets and saved as .txt files to serve as the benchmarks for each algorithm. The proposed system and workflow are illustrated in Figure 2.

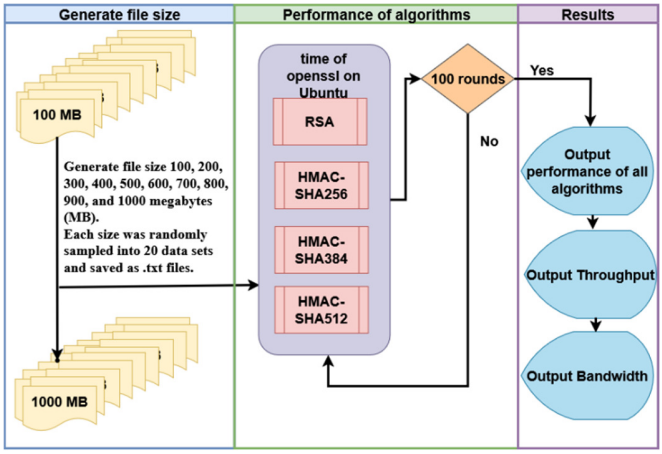

Fig. 2. Schematic of the proposed system.

B. File Generation

To generate the random character files, the researcher used the command shown in Figure 3, where 'head -c 100M' specifies the number of bytes for the random character file.

```
pn@pn-VirtualBox:~$ for i in {1..20}; do tr -dc A-Za-z0-9
 <dev/urandom | head -c 1000M > random1000_$i.txt; done
```

Fig. 3. Command to create random character files.

The command generates random character files of the following sizes: 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 MB.

C. Digital Signature Performance

1) Key Generation

To test the performance of digital signatures, the researcher used the RSA algorithm to generate private and public keys of 2048 bits, which were saved in .pem files. The commands used for generating private keys are shown in Figure 4, and the commands for generating public keys are shown in Figure 5.

```
pn@pn-VirtualBox:~$ openssl genrsa -out private-key.pem 2048
pn@pn-VirtualBox:~$
```

Fig. 4. Command to generate private keys.

```
pn@pn-VirtualBox:~$ openssl rsa -in private-key.pem
-pubout -out public-key.pem
writing RSA key
pn@pn-VirtualBox:~$
```

Fig. 5. Command to generate public keys.

2) Digital Signature Testing

The researcher used the '-sign' command with '-sha256' and the private keys created in the key generation step to create a digital signature for each data file ($i), which was stored in a file named $i.bin. The 'time' command was used to measure performance, and the results were exported to a file named according to the size of the test file, as shown in Figure 6.

```
pn@pn-VirtualBox:~$ for i in $(ls random1000_*.txt);
do (time openssl dgst -sha256 -sign private-key.pem "
$i" > "$i.bin") 2>> output1000.txt; done
```

Fig. 6. Command for digital signature testing.

D. HMAC Performance Testing

The researcher tested the performance of HMAC algorithms (HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512) using a 128-bit key. Execution time was measured with the 'time openssl' command, and the results were exported to files named according to the type of test and size of the data. The commands used for each algorithm are shown in Figures 7–9.

```
pn@pn-VirtualBox:~$ for i in $(ls random1000_*.txt);
do (time openssl sha256 -mac hmac -macopt hexkey:0123
456789ABCDEF0123456789ADCDEF $i) 2>> sha256_1000.txt;
 done
```

Fig. 7. Command for performance testing of HMAC-SHA256.

```
pn@pn-VirtualBox:~$ for i in $(ls random1000_*.txt);
do (time openssl sha384 -mac hmac -macopt hexkey:0123
456789ABCDEF0123456789ADCDEF $i) 2>> sha384_1000.txt;
 done
```

Fig. 8. Command for performance testing of HMAC-SHA384.

```
pn@pn-VirtualBox:~$ for i in $(ls random1000_*.txt);
do (time openssl sha512 -mac hmac -macopt hexkey:0123
456789ABCDEF0123456789ADCDEF $i) 2>> sha512_1000.txt;
 done
```

Fig. 9. Command for performance testing of HMAC-SHA512.

Table I summarizes the file sizes, algorithms, key sizes, and number of datasets per size used in the experiments.

TABLE I. DATA SIZES, ALGORITHMS, KEY SIZES, AND NUMBER OF DATASETS PER SIZE

File size	Number of datasets	Algorithm	Key size (bits)
100–1000	20	RSA	2048
100–1000	20	HMAC-SHA256	256
100–1000	20	HMAC-SHA384	384
100–1000	20	HMAC-SHA512	512

III. RESULTS AND DISCUSSION

A. Results

The results of the execution time comparison for the digital signature RSA algorithm, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512 are shown in Table II. The table shows the average execution time for each file size, calculated from 20 datasets and 100 testing rounds per size. The comparative performance of all algorithms is summarized in Table II and illustrated graphically in Figure 10.

TABLE II. COMPARATIVE EXECUTION TIME OF ALL ALGORITHMS

File size (MB)	RSA (s)	HMAC-SHA256 (s)	HMAC-SHA384 (s)	HMAC-SHA512 (s)
100	0.00720	0.00485	0.00450	0.00550
200	0.27995	0.26075	0.26615	0.27135
300	0.46745	0.23055	0.42775	0.42425
400	0.57605	0.29160	0.55865	0.56230
500	0.70285	0.36080	0.69135	0.69250
600	0.83610	0.42690	0.82580	0.83120
700	0.97495	0.49710	0.95955	0.96575
800	1.10845	0.56450	1.09355	1.09555
900	1.26315	0.64205	1.22740	1.22865
1000	1.37225	0.70490	1.36955	1.36270
Average	0.75884	0.39840	0.74243	0.74398

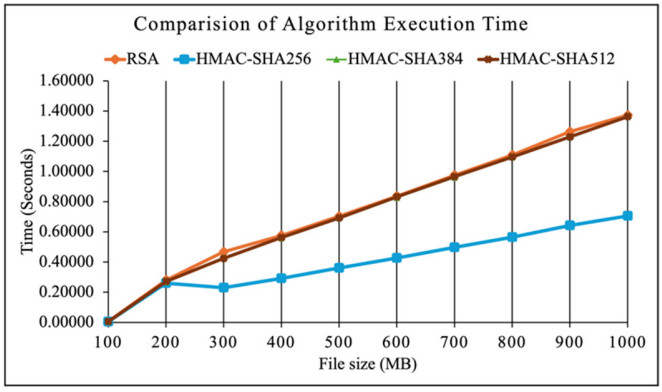

Fig. 10. Average execution time of RSA and HMAC algorithms for each file size.

In terms of throughput, shown in Figure 11, the HMAC-SHA256 algorithm performed the fastest, achieving the highest throughput of 1,247,509,128.27 bytes/s. The next most efficient algorithm was HMAC-SHA384, with a throughput of 688,389,845.09 bytes/s, followed by HMAC-SHA512 at 685,619,347.64 bytes/s. The RSA algorithm exhibited the lowest throughput, at 668,445,104.83 bytes/s.

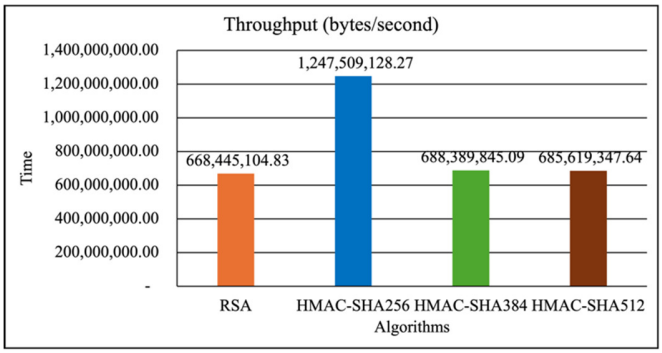

Fig. 11. Average throughput of all algorithms.

The bandwidth evaluation, captured within the time intervals, was in agreement with the throughput results. HMAC-SHA256 achieved the highest average bandwidth of 1189.71 MB/s. HMAC-SHA384 and HMAC-SHA512 demonstrated comparable bandwidths of 657.08 MB/s and 653.85 MB/s, respectively, whereas RSA exhibited the lowest bandwidth of 632.90 MB/s, as shown in Figure 12.

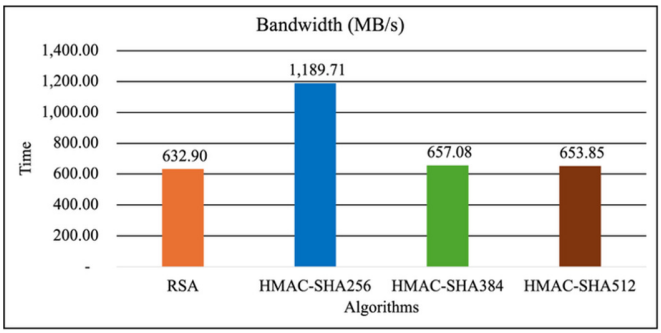

Fig. 12. Average bandwidth of all algorithms.

B. Discussion

The algorithm performance evaluation confirmed that HMAC-SHA256 achieved the best performance due to its optimized implementation and balance between security strength and processing efficiency. The SHA-256 hash function, employed as the basis of HMAC-SHA256, is designed to operate efficiently on modern processors and utilize hardware acceleration on multiple platforms. A key observation is the extreme difference in execution time between the 100 MB files and larger files across all algorithms. This behavior indicates that the encryption operation involves a high initial computational overhead, which diminishes as the file size increases. For file sizes of 200 MB and above, the performance is reasonably consistent, reflecting the optimal performance of the algorithm.

Although HMAC-SHA256 demonstrated the highest performance, the selection of an algorithm should also consider security requirements. HMAC-SHA384 and HMAC-SHA512 provide stronger cryptographic security, although with slightly

reduced performance compared with HMAC-SHA256. RSA is an asymmetric algorithm with different cryptographic purposes, such as digital signatures and key exchanges, and is expectedly slower due to its mathematical complexity.

IV. CONCLUSION

This research provides the first extensive comparative analysis of Rivest–Shamir–Adleman (RSA) digital signatures and Hash-based Message Authentication Code (HMAC) variants, including Secure Hash Algorithm (SHA)-256, SHA-384, and SHA-512, in large-scale file processing (100–1000 MB), with systematic evaluations of throughput and bandwidth. Although prior studies have examined these algorithms independently or in small-scale implementations, this work bridges the gap by explicitly comparing their performance using larger datasets that better represent actual use cases such as cloud storage, e-commerce platforms, and multimedia file sharing.

A standardized evaluation method was established using 20 random datasets per file size and 100 test rounds to ensure statistical reliability, an aspect often overlooked in earlier cryptographic performance studies. The results indicate that HMAC-SHA256 achieved the highest throughput among the four algorithms and is therefore the most suitable choice for operations requiring fast processing time while ensuring data integrity and sender authentication. Because SHA256 is a secure and widely used hash algorithm, HMAC-SHA256 achieves an optimal balance between security and performance for time-sensitive applications.

For future research, increasing the data size to gigabytes and beyond could reveal additional insights into the scalability of each algorithm. Similarly, increasing the number of test rounds would further enhance the accuracy and reliability of the results. Practically, for large files (≥200 MB), HMAC-SHA256 would be the best choice for symmetric authentication, offering nearly twice the performance of HMAC-SHA384 and HMAC-SHA512. However, for smaller files or scenarios where maximum cryptographic strength is essential, HMAC-SHA512 provides the strongest security with acceptable performance.

The findings align with those reported by authors in [8] on HMAC optimization, demonstrating that HMAC-SHA256's superior throughput (1,247,509,128.27 bytes/s) validates their energy-efficient design approach. This study extends those findings by including comprehensive bandwidth measurements (1189.71 MB/s) across varying file sizes. Compared with the quantum-focused research by authors in [22] and post-quantum signature schemes investigated by authors in [18], the present work offers baseline performance metrics for current cryptographic algorithms and provides benchmarks for evaluating future post-quantum alternatives. Future investigations of algorithms capable of ensuring data integrity and sender authentication, such as RSA-based digital signatures, Elliptic Curve Cryptography (ECC), and Galois Message Authentication Code (GMAC), or applications implementing HMAC-SHA256 for authentication and data integrity verification, will further guide the selection of the most suitable algorithm for specific applications.

REFERENCES

[1] S. Lallouche, "Legal Controls for the Validity of Electronically Signed Documents," *RIMAK International Journal of Humanities and Social Sciences*, vol. 5, no. 2, pp. 936–953, Mar. 2023, https://doi.org/10.47832/2717-8293.22.53.

[2] E. Barker, "Recommendation for Key Management: Part 1 – General," National Institute of Standards and Technology, NIST Special Publication (SP) 800-57 Part 1 Rev. 5, May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5.

[3] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Aug. 2001, https://doi.org/10.1007/s102070100002.

[4] J. Chandrashekhara, A. V B, P. H, and R. B R, "A Comprehensive Study on Digital Signature," *International Journal of Innovative Research in Computer Science & Technology*, vol. 9, no. 3, May 2021, Art. no. IRP1149, https://doi.org/10.21276/ijircst.2021.9.3.7.

[5] A. H. Mansour, "Analysis of RSA Digital Signature Key Generation using Strong Prime," *International Journal of Computer*, vol. 24, no. 1, pp. 28–36, Feb. 2017, https://doi.org/10.53896/ijc.v24i1.816.

[6] H. Siregar, E. Junaeti, and T. Hayatno, "Implementation of Digital Signature Using Aes and Rsa Algorithms as a Security in Disposition System af Letter," *IOP Conference Series: Materials Science and Engineering*, vol. 180, no. 1, Mar. 2017, Art. no. 012055, https://doi.org/10.1088/1757-899X/180/1/012055.

[7] E. S. I. Harba, "Secure Data Encryption Through a Combination of AES, RSA and HMAC," *Engineering, Technology & Applied Science Research*, vol. 7, no. 4, pp. 1781–1785, Aug. 2017, https://doi.org/10.48084/etasr.1272.

[8] C. E. Castellon, S. Roy, O. P. Kreidl, A. Dutta, and L. Bölöni, "Towards an Energy-Efficient Hash-based Message Authentication Code (HMAC)," in *2022 IEEE 13th International Green and Sustainable Computing Conference*, Pittsburgh, PA, USA, 2022, pp. 1–7, https://doi.org/10.1109/IGSC55832.2022.9969377.

[9] S. Suhaili, N. Julai, R. Sapawi, and N. Rajaee, "Towards Maximising Hardware Resources and Design Efficiency via High-Speed Implementation of HMAC based on SHA-256 Design," *Pertanika Journal of Science and Technology*, vol. 32, no. 1, pp. 31–44, Nov. 2023, https://doi.org/10.47836/pjst.32.1.02.

[10] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Engineering Task Force, Request for Comments RFC 2104, Feb. 1997. https://doi.org/10.17487/RFC2104.

[11] M. Bellare, R. Canetti, and H. Krawczyk, "Message Authentication using Hash Functions— The HMAC Construction," *RSA Laboratories' CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[12] K. Boppudi, "Efficient HMAC Based Message Authentication System for Mobile Environment," *Global Journal of Computer Science and Technology*, vol. 11, no. 19, pp. 55–59, Jul. 2011.

[13] P. Natho and P. Kuacharoen, "Cloud Storage Security Based on Group Key," *Advanced Science Letters*, vol. 21, no. 10, pp. 3156–3160, Oct. 2015, https://doi.org/10.1166/asl.2015.6510.

[14] P. Natho, S. Somsuphaprungyos, S. Boonmee, and S. Boonying, "Comparative study of password storing using hash function with MD5, SHA1, SHA2, and SHA3 algorithm," *International Journal of Reconfigurable and Embedded Systems*, vol. 13, no. 3, pp. 502–511, Nov. 2024, https://doi.org/10.11591/ijres.v13.i3.pp502-511.

[15] P. Wuttidittachotti and P. Natho, "Improved ciphertext-policy time using short elliptic curve Diffie–Hellman," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 4, pp. 4547–4556, Aug. 2023, https://doi.org/10.11591/ijece.v13i4.pp4547-4556.

[16] J. Yu, F. Luo, G. Pu, and M. Chen, "A Trade-off Design Approach for Integrating Cybersecurity, Safety, and Other Aspects of Intelligent Connected Vehicles," in *Proceedings of China SAE Congress 2022: Selected Papers*, Nantong, Jiangsu, China, 2023, pp. 577–592, https://doi.org/10.1007/978-981-99-1365-7_43.

[17] Y. Zhuo, Z. Song, and Z. Ge, "Security Versus Accuracy: Trade-Off Data Modeling to Safe Fault Classification Systems," *IEEE Transactions*

on Neural Networks and Learning Systems, vol. 35, no. 9, pp. 12095–12106, Sep. 2024, https://doi.org/10.1109/TNNLS.2023.3251999.

[18] O. Potii, Y. Gorbenko, and K. Isirova, "Post quantum hash based digital signatures comparative analysis. Features of their implementation and using in public key infrastructure," in *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology*, Kharkov, Ukraine, 2017, pp. 105–109, https://doi.org/10.1109/INFOCOMMST.2017.8246360.

[19] M. D. Noel, O. V. Waziri, M. S. Abdulhamid, A. J. Ojeniyi, and M. U. Okoro, "Comparative Analysis of Classical and Post-quantum Digital Signature Algorithms used in Bitcoin Transactions," in *2020 2nd International Conference on Computer and Information Sciences*, Sakaka, Saudi Arabia, 2020, pp. 1–6, https://doi.org/10.1109/ICCIS49240.2020.9257656.

[20] M. Ramadhoni and H. Santoso, "Performance Comparison between Signature Cryptography: A Case Study on SNAP Indonesia," *Sinkron : jurnal dan penelitian teknik informatika*, vol. 7, no. 4, pp. 2327–2335, Oct. 2023, https://doi.org/10.33395/sinkron.v8i4.12819.

[21] W. Uriawan, R. Ramadita, R. D. Putra, R. I. Siregar, and R. Addiva, "Authenticate and Verification Source Files using SHA256 and HMAC Algorithms." Preprints, Jul. 01, 2024, https://doi.org/10.20944/preprints202407.0075.v1.

[22] D. Rana, "Advancements and Comparative Analysis of Digital Signature Algorithms: A Review," *International Journal for Research in Applied Science and Engineering Technology*, vol. 12, no. 5, pp. 5778–5792, May 2024, https://doi.org/10.22214/ijraset.2024.62955.

[23] Q. Chen, H. Li, S. B. Ariffin, and N. A. B. Mustapa, "A Comprehensive Study on the Homomorphic Encryption for Secure Image Data Processing," *Engineering, Technology & Applied Science Research*, vol. 15, no. 2, pp. 21783–21790, Apr. 2025, https://doi.org/10.48084/etasr.10007.