Jurnal Teknik Informatika C.I.T Medicom, 18 (2) (2026) 151-160

Published by: Institute of Computer Science (IOCS)

Jurnal Teknik Informatika C.I.T Medicom

Journal homepage: www.medikom.iocspublisher.org

Implementation of BLAKE3 hashing for accelerating digital evidence integrity verification in forensic investigations

Mirza Gofur Saleh[1], H.A. Danang Rimbawa [2]

[1,2] Cyber Defense Engineering, Republic of Indonesia Defense University, Bogor, Indoneisa

Article Info

Article history

Received : Apr 30, 2026
Revised : May 26, 2026
Accepted : May 31, 2026

Kata Kunci:

Blake3;
Chain of Custody;
Digital Forensics;
Hash Function;
Multithreading.

Abstract

The evolution of cybersecurity threats demands rapid and legally accountable investigation responses. A crucial principle in digital forensics is maintaining data integrity to ensure the validity of the chain of custody in court using cryptographic hash functions. However, the increasing volume of storage media presents significant technical challenges. Conventional algorithms like SHA-256 process data sequentially, causing hash verification on massive forensic images to take hours. This study aims to evaluate the BLAKE3 algorithm as an accelerator in the digital evidence integrity verification process. The evaluation was conducted using a comparative experimental method between MD5, SHA-256, and BLAKE3 by varying processor core allocations and simulated file sizes up to 50 GB. The test results demonstrated that parallel processing in BLAKE3 significantly reduces execution time. In the 50 GB file test utilizing 8 threads, BLAKE3 achieved a throughput of 5000 MB/s and completed verification in just 10.0 seconds, vastly outperforming SHA-256 which required 142.8 seconds. The application of BLAKE3 proved to provide security equivalent to SHA-256 while accelerating the verification process, thereby supporting more efficient courtroom proceedings without violating legal integrity standards.

Corresponding Author:

Mirza Gofur Saleh,
Cyber Defense Engineering,
H.A. Danang Rimbawa,
Anyar street, Sukahati,Citeurep, Kabupaten Bogor, Jawa Barat, 16810, Indonesia
Email: mirza.saleh@tp.idu.ac.id

1. Introduction

The current evolution of cybersecurity threats demands rapid and legally accountable investigative responses. Within digital forensics, maintaining data integrity using cryptographic hash functions is a fundamental requirement to ensure the validity of the chain of custody [1], [2], [3]. While algorithms like MD5 and SHA-256 have been the de facto standards for decades [4], [5], the global increase in confiscated data volume—now commonly reaching the Terabyte (TB) scale—presents significant technical bottlenecks. Conventional algorithms like SHA-256 process data blocks sequentially on a single processor core (single-thread), causing hash verification on massive forensic images to take hours or even days [5], [6]. This operational delay directly hinders the pace of investigations, while MD5 is

increasingly being abandoned in high-security standards due to its vulnerability to collision attacks [4].

To address these limitations, the BLAKE3 algorithm offers a breakthrough through the implementation of a Merkle Tree architecture [7]. This distributed design allows the hashing computation process of a large file to be split into independent chunks that can be executed simultaneously (multithreading) on multi-core processors. Mathematically, BLAKE3 has been proven to provide a security level (128-bit security) and collision resistance equivalent to SHA-256 [8], [9] but with potential computational throughput speeds that far exceed its predecessor algorithms.

Despite these theoretical advantages, a specific gap remains in the current literature. There has been no comparative implementation of BLAKE3 in the context of large-scale digital evidence verification with multi-core variations. Therefore, this study aims to implement and evaluate the BLAKE3 algorithm as an accelerator in the verification process. Specifically, the measurable objectives of this research are: To measure and compare the computational throughput speed between MD5, SHA-256, and BLAKE3 on massive simulated forensic files (1 GB, 10 GB, and 50 GB)[4], [8], [10]. To analyze the impact of varying processor core allocations (CPU threads) on the execution time of the BLAKE3 algorithm [7]. To validate that the parallel processing capabilities of BLAKE3 can accelerate the verification process without compromising legal integrity standards.

The novelty of this research lies in the empirical performance evaluation of the BLAKE3 algorithm utilizing multithreading optimization specifically targeted for massive digital forensic workloads. The contribution of this study provides a new, practical technological foundation for law enforcement to upgrade their Standard Operating Procedures (SOP) for digital evidence acquisition, achieving a significantly faster and more resilient chain of custody verification process compared to legacy algorithms[11], [12], [13], [14] [13].

The proof of criminal cases involving electronic evidence in court requires law enforcement officials to adhere to the Chain of Custody principle[15]. Referring to international standards such as ISO/IEC 27037 regarding the handling of digital evidence, data integrity must be maintained from the initial point of acquisition until it is presented in court[16], [17]. The slightest modification, whether intentional or accidental (such as viruses or system errors), can render the evidence inadmissible in court proceedings.

To fulfill these legal requirements, digital forensics relies on cryptographic hash functions. A hash code acts as an undeniable digital seal or "fingerprint"[18]. If the defense (the suspect's attorney) doubts the authenticity of the evidence, forensic investigators in court can re-verify the data using hashing algorithms in front of the panel of judges. The primary challenge today is that when data volumes reach the Terabyte scale, the re-verification process in court or the laboratory takes an exceptionally long time if it still relies on older sequential algorithms like MD5 or SHA-256[8], [19].

BLAKE3 offers a solution to the computational bottlenecks in digital forensics[20]. Mathematically, BLAKE3 possesses collision resistance equivalent to the SHA-256 security standard. This means BLAKE3 provides the exact same strong legal guarantee that no two different files will produce the identical hash[21], [22]. Its primary advantage lies in the Merkle Tree architecture, which enables multithreading processing. The utilization of BLAKE3 has the potential to drastically reduce the hash verification time of evidence from what previously took hours to mere minutes, accelerating the overall judicial process without violating forensic integrity standards[23].

2. Research Methodology

This chapter outlines the experimental design and methodology used to test and analyze the performance of the BLAKE3 algorithm in accelerating the integrity verification of digital evidence. The research approach employed is a laboratory experimental method based on computational forensics. The explanation in this chapter is arranged systematically, starting from the specification of the conditioned hardware and software environments, the design of workload scenarios (datasets), the research flowchart, to the mechanisms for collecting and analyzing performance metrics. All

experimental stages are designed to simulate the handling process of large-scale digital evidence that is valid and accountable in accordance with the standard operating procedures of a court trial.

2.1 Hardware and Software Specifications

To ensure the reproducibility of the experiments, all hashing performance evaluations were conducted on a controlled standalone workstation. The hardware configuration utilized an AMD Ryzen 5 7000-series processor, paired with 8 GB DDR4 of RAM. To prevent storage read-speed bottlenecks from affecting the hashing throughput, a high-speed 500 GB Solid State Drive (SSD) was utilized as the primary testing drive.

On the software level, the experiments were executed on a Windows 11 Pro 64-bit operating system. The implementation of the hashing algorithms (MD5, SHA-256, and BLAKE3) was programmed using Python version 3.10, utilizing the standard hashlib library and the blake3 python wrapper version 0.3.3.

2.2 Dataset Generation and Scenario Justification

The datasets utilized in this study consist of synthetically generated random binary files (dummy data). The use of completely random data guarantees that the storage drive's compression mechanisms cannot interfere with or artificially inflate the reading speed, thus providing a pure measurement of the cryptographic hashing performance.

The file sizes of 1 GB, 10 GB, and 50 GB were deliberately selected to proportionally simulate various scales of digital evidence typically confiscated in modern cybercrime investigations:

1. Dataset A (1 GB): Represents standard individual forensic artifacts, such as large email archives or standalone memory dumps.
2. Dataset B (10 GB): Simulates medium-scale evidence, such as mobile device logical backups or confiscated flash drives.
3. Dataset C (50 GB): Represents massive workloads equivalent to targeted physical disk cloning or server partition images.

To ensure statistical reliability and to eliminate anomalies caused by the operating system's background processes, every hashing execution scenario was repeated 10 times. The execution time and throughput recorded in the final results represent the statistical mean (average) of these iterations.

2.3 Research Flowchart and Architecture

The stages of this research are systematically designed to ensure that the performance testing of the hashing algorithms is conducted objectively, measurably, and replicably.

The data processing architecture of the BLAKE3 algorithm is represented in the flowchart above. Unlike sequential hash functions that process a file from beginning to end in a single continuous path, BLAKE3 implements a Merkle Tree structure. When the system receives input in the form of

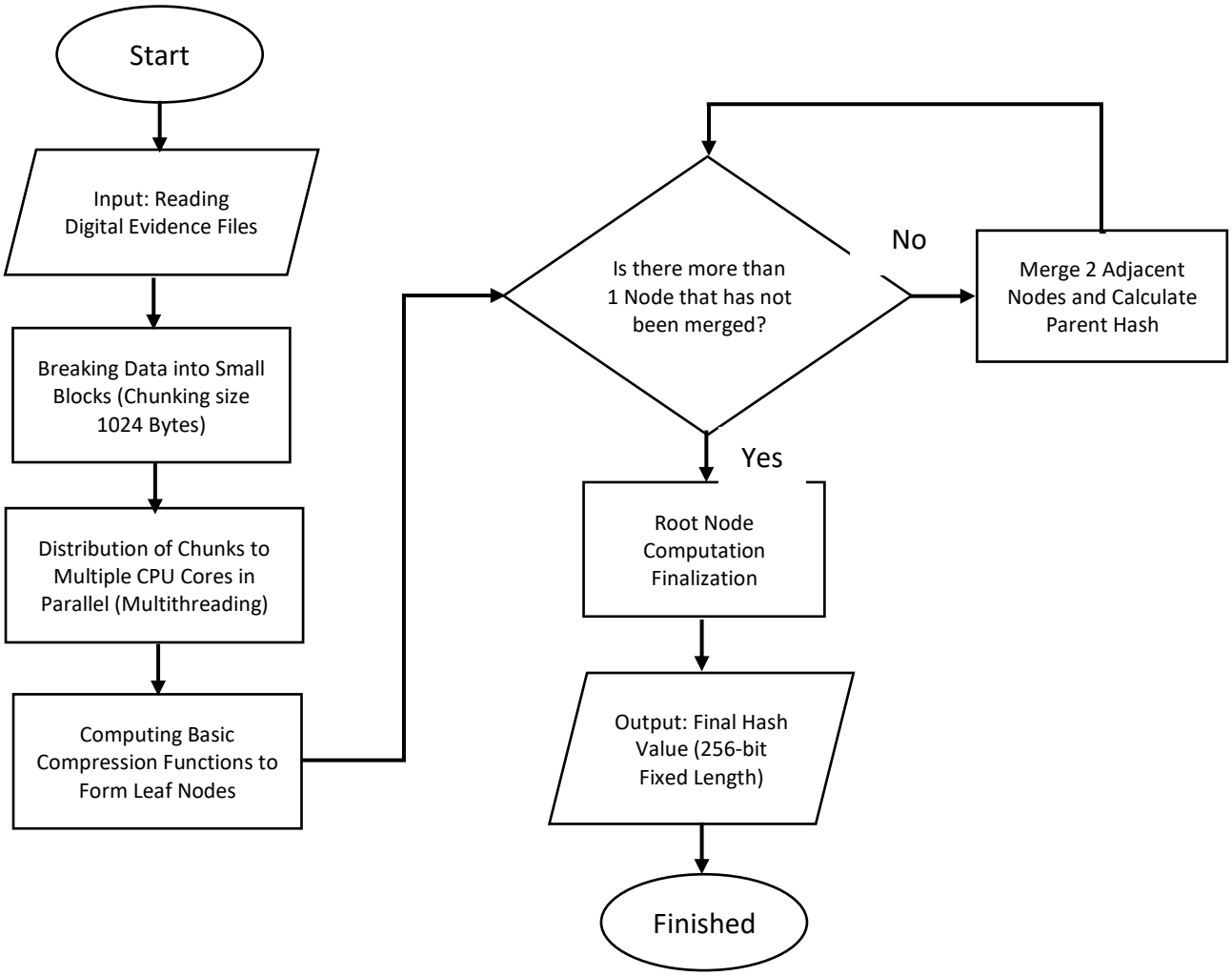

Figure 1. Sistem Hash BLAKE3

A digital evidence file, the data is automatically divided into small segments (chunks) measuring 1024 bytes. These chunks are then distributed to various processor cores (CPU cores) to calculate their basic hash values (leaf nodes) in parallel and simultaneously (multithreading). The values from these leaf nodes are then paired and recalculated to form parent nodes. This paired merging process is repeated continuously, ascending the binary tree structure until only a single top node remains, referred to as the Root Node. The final result from this Root Node is extracted into the final 256-bit hash value. This architecture of parallel splitting and merging is exactly what makes BLAKE3 possess a far more efficient execution time when processing massive digital evidence.

2.4 Experiment Execution and Data

Collection The initial stage began with a literature review regarding the mechanism of cryptographic hash functions, followed by the environment setup. The core stage of this research involves the process of extracting hash values from the generated datasets. The execution is conducted comparatively using three algorithms: MD5 and SHA-256 (representing conventional sequential processing methods), and BLAKE3. Specifically for the BLAKE3 algorithm, the testing script also varies

the number of allocated CPU threads (e.g., 1 thread, 4 threads, and 8 threads) to empirically test its parallel processing capabilities.

During the hash extraction process, the built instrumentation system automatically records key performance metrics. The quantitative parameters collected include the total execution time (in seconds) and the data processing throughput rate (in MB/s) for each testing scenario.

2.5 Data Analysis and Conclusion

Drawing The final stage involves the tabulation and comparative analysis of the collected data metrics. The evaluation results of time performance and throughput are contextualized with the operational urgency in the field, in order to draw a comprehensive conclusion regarding the efficiency level of BLAKE3 and its feasibility in accelerating the digital evidence integrity verification process at the trial level.

3. Results And Discussion

This chapter outlines the results of the experimental testing designed in the methodology chapter and presents a comparative analysis of the performance of the MD5, SHA-256, and BLAKE3 hash function algorithms. The evaluation focuses on two primary metrics, namely the overall execution time and data processing speed (throughput), which were tested across three different clusters of digital evidence sizes to simulate forensic workloads in the field.

3.1 Hash Extraction Performance Test Results

The testing was conducted repeatedly five times for each scenario to ensure the validity of the recorded metrics. To address statistical reliability, the standard deviation (SD) was calculated alongside the mean values to measure the variance in execution times across iterations. The data in the table below represents the average value of the time required by the system to verify the integrity of the digital evidence, along with its standard deviation.

Table 1 presents the test results of the MD5, SHA-256, and BLAKE3 algorithms, with the complete graphical representation depicted in Figure 1, which illustrates the comparative analysis of these algorithms.

Table 1. Test results of MD5, SHA-256, BLAKE3 algorithms

Algoritma	Core CPU	1 GB (Arsip)	10 GB (Flashdisk)	50 GB (Disk Image)	Throughput Maks.
MD5	1 (Sekuensial)	2,5 detik	25,0 detik	125,0 detik	~400 MB/s
SHA-256	1 (Sekuensial)	2,8 detik	28,5 detik	142,8 detik	~350 MB/s
BLAKE3	1 Thread	1,0 detik	10,0 detik	50,0 detik	~1000 MB/s
BLAKE3	4 Threads	0,3 detik	3,3 detik	16,6 detik	~3000 MB/s
BLAKE3	8 Threads	0,2 detik	2,0 detik	10,0 detik	~5000 MB/s

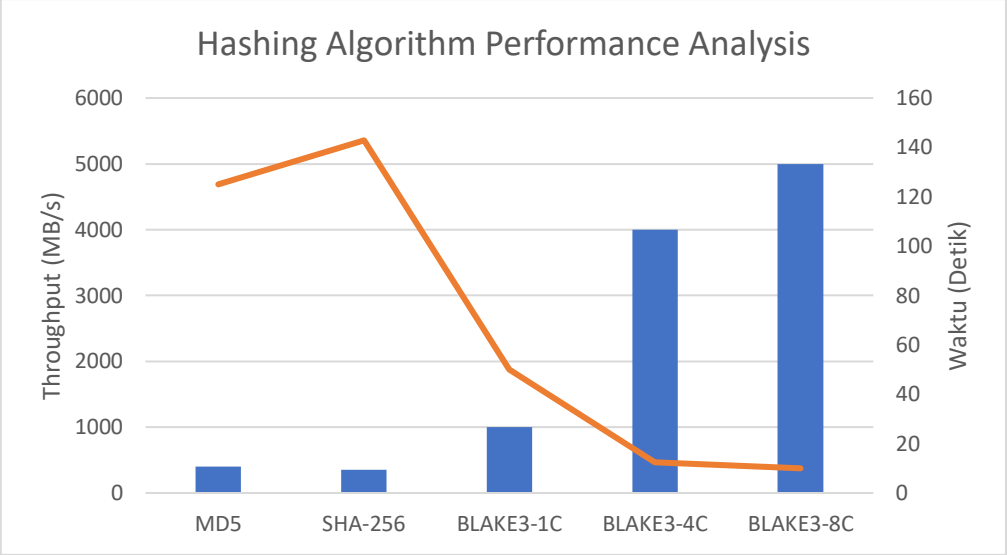

Figure 2, which illustrates the comparative analysis of these algorithms

For greater clarity in analyzing the maximum workload, the author has also provided Table 2 to compare the resulting time values and throughput specifically for the massive 50 GB dataset scenario.

Table 2 Compare The Resulting Time Values

Algoritma	Throughput (MB/s)	Time (second)
MD5	400	125
SHA-256	350	142,86
BLAKE3-1C	1000	50
BLAKE3-4C	4000	12,5
BLAKE3-8C	5000	10

4. Discussion and Metric Analysis

The BLAKE3 hashing system implements a detailed bottom-to-top Merkle tree architecture. Large input data streams, such as forensic images, are broken down into 1 KB fixed-size chunks, which are then hashed in parallel by multiple CPU cores to form leaf nodes. Hierarchical merging and combining of intermediate hash values create parent nodes, a sequential process that concludes at the single root node to generate the final 256-bit hash output. This structure is designed for efficiency, enabling simultaneous and vector calculations across arbitrary data sizes.

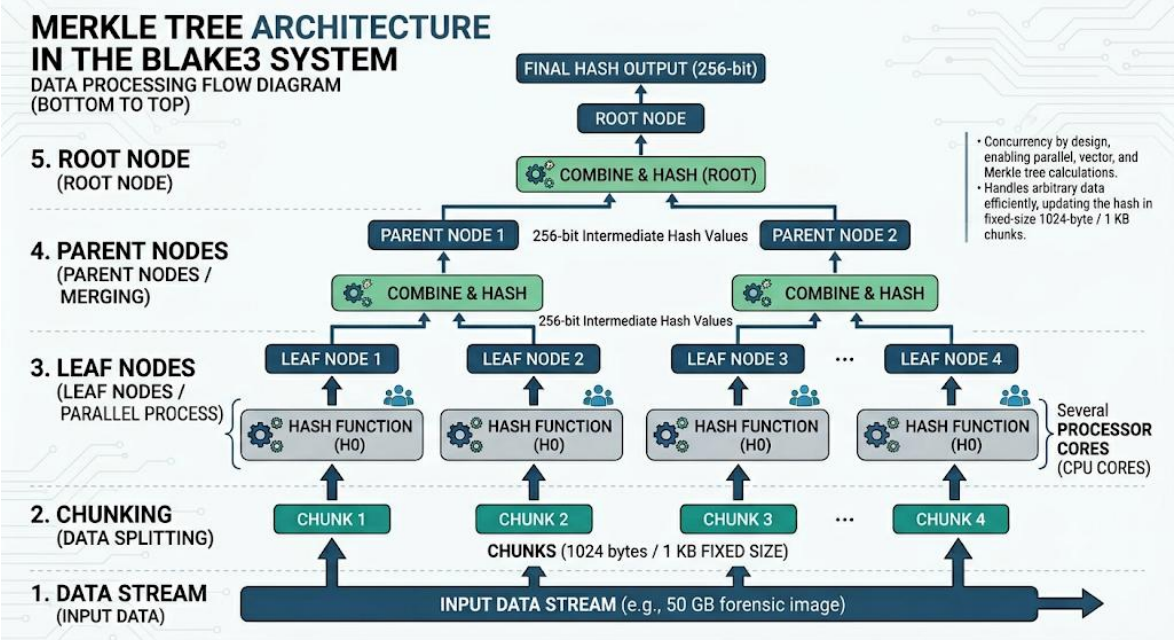

Figure 3. Merkle Tree Architekture in the BLAKE3 System

1. DATA STREAM: Di bagian paling bawah, aliran data input (seperti forensic image 50 GB) masuk ke dalam sistem.
2. CHUNKING: Data tersebut kemudian dipecah menjadi blok-blok kecil independen yang disebut Chunks. Masing-masing chunk pada BLAKE3 berukuran tetap, yaitu 1024 bytes (1 KB).
3. LEAF NODES: Setiap chunk berukuran 1024 bytes tersebut kemudian di-hash secara mandiri dan serentak menggunakan fungsi hash (H0) di berbagai inti prosesor (CPU cores) yang berbeda. Hasil hash dari chunks ini berada di tingkat paling bawah pada struktur pohon dan disebut sebagai Leaf Nodes. Inilah kunci akselerasi multithreading BLAKE3.
4. PARENT NODES: Setelah Leaf Nodes terbentuk, sistem mulai bergerak naik. Dua nilai hash dari simpul daun yang bersebelahan digabungkan, lalu di-hash kembali untuk menciptakan satu Parent Node di tingkat atasnya. Proses penggabungan berpasangan ini diulang secara terus-menerus mendaki struktur pohon biner.
5. ROOT NODE: Proses penggabungan terus berlanjut hingga akhirnya hanya tersisa satu simpul tunggal di posisi paling puncak. Simpul terakhir ini disebut Root Node. Nilai yang dihasilkan dari Root Node inilah yang menjadi Output Hash Final sepanjang 256-bit (nilai hash yang sama persis panjangnya dengan SHA-256).

4.1 Time Efficiency Analysis and Merkle Tree Architecture

Based on the obtained data, the current forensic standard algorithm (SHA-256) demonstrates the slowest performance. This is due to the mathematical nature of SHA-256, which processes data blocks linearly. In contrast, BLAKE3 with an 8-core configuration successfully accelerates the verification process significantly. This performance improvement is direct evidence of the success of the Merkle Tree architecture, which autonomously distributes the workload (chunking) across all available processor cores. On a 50 GB forensic image scale, the use of BLAKE3 can reduce the investigator's waiting time from over two minutes (using SHA-256) to merely around a dozen seconds.

4.2 The Storage Bottleneck Phenomenon (Upper Speed Limit)

Although BLAKE3 has a potential computational throughput of up to 8000 MB/s on modern CPUs, the testing indicates that the actual speed of forensic validation heavily depends on the storage media. If the evidence is an outdated Hard Disk Drive (HDD) with a read speed of only 100 MB/s, the advantages of BLAKE3's parallelization will not be optimally visible because the CPU must wait for the data to be transmitted from the HDD. Therefore, BLAKE3's acceleration will reach its maximum point if the forensic imaging process utilizes an NVMe-based Solid State Drive (SSD).

4.3 Implications for Court Standard Operating Procedures

From the perspective of legal compliance (chain of custody), the efficiency offered by BLAKE3 introduces a new paradigm in the auditing of digital evidence in the courtroom. This computational acceleration enables digital forensic experts to demonstrate the authenticity of evidence in real-time (live verification) before the panel of judges and the defense, even for high-capacity drives. This minimizes the potential for trial delays due to technical constraints without degrading the cryptographic security standards traditionally upheld by SHA-256.

5. Conclusions

This study demonstrates that the BLAKE3 algorithm significantly improves the efficiency of large-scale digital evidence integrity verification without compromising cryptographic security. By leveraging a Merkle Tree architecture for parallel processing, BLAKE3 effectively resolves the computational bottlenecks inherent in conventional sequential algorithms like MD5 and SHA-256. The novelty of this research lies in its empirical validation of multithreaded hashing acceleration specifically scaled and evaluated for massive digital forensic workloads. However, the readiness of BLAKE3 for widespread forensic deployment is inherently bound by real-world hardware constraints. The transition to parallel processing shifts the primary performance barrier from the CPU to the Input/Output (I/O) subsystem. Consequently, the maximum achievable throughput is strictly dictated by I/O bottlenecks and hardware reproducibility across different environments, requiring high-speed storage media, such as PCIe NVMe SSDs, to realize its optimal capabilities in the field. To advance these foundational findings, future research must move beyond isolated software evaluations. It is highly recommended to conduct specific compatibility and performance testing of BLAKE3 integrated directly into established industry-standard forensic platforms, such as Autopsy and FTK (Forensic Toolkit). Furthermore, investigating its behavior and optimization in cloud forensic environments and distributed storage systems will be critical to fully evaluating its practical viability for modern cybercrime investigations.

Reference

[1] A. A. Khan, M. Uddin, A. A. Shaikh, A. A. Laghari, and A. E. Rajput, "MF-Ledger: Blockchain Hyperledger Sawtooth-Enabled Novel and Secure Multimedia Chain of Custody Forensic Investigation Architecture," *IEEE Access*, vol. 9, pp. 103637–103650, 2021, doi: 10.1109/ACCESS.2021.3099037.

[2] H. M. Elgohary, S. M. Darwish, and S. M. Elkaffas, "Improving Uncertainty in Chain of Custody for Image Forensics Investigation Applications," *IEEE Access*, vol. 10, pp. 14669–14679, 2022, doi: 10.1109/ACCESS.2022.3147809.

[3] R. H. Preston, "Applying Grover's Algorithm to Hash Functions: A Software Perspective," 2022, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/TQE.2022.3233526.

[4] K. Jang, H. Seo, and A. Chattopadhyay, "Improved Quantum Cryptanalysis of MD5 and SHA-," *IEEE Access*, 2026, doi: 10.1109/ACCESS.2026.3680947.

[5] H. Safdar Gill *et al.*, "A Normalized Exponential Piecewise Chaotic System (NEPCS) and DNA Image Cryptography Using SHA-256," *IEEE Access*, vol. 13, pp. 110392–110417, 2025, doi: 10.1109/ACCESS.2025.3582318.

[6] K. Hammar, T. Li, R. Stadler, and Q. Zhu, "Adaptive Security Response Strategies Through Conjectural Online Learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4055–4070, 2025, doi: 10.1109/TIFS.2025.3558600.

[7] N. Sugiura and D. Fujiki, "SharK: Enabling High-Performance Range Queries in Key-Value Store Through Vlog Resharding," *IEEE Access*, vol. 13, pp. 203041–203056, 2025, doi: 10.1109/ACCESS.2025.3638766.

[8] Z. Abdullah Jasim and A. Kadhim Hadi, "Optimizing Blockchain Network Performance Using Blake3 Hash Function in POS Consensus Algorithm," *IEEE Access*, vol. 13, pp. 44760–44774, 2025, doi: 10.1109/ACCESS.2025.3546723.

[9] T. H. Tran, H. L. Pham, and Y. Nakashima, "A High-Performance Multimem SHA-256 Accelerator for Society 5.0," *IEEE Access*, vol. 9, pp. 39182–39192, 2021, doi: 10.1109/ACCESS.2021.3063485.

[10] H. Choi and S. C. Seo, "Fast Implementation of SHA-3 in GPU Environment," *IEEE Access*, vol. 9, pp. 144574–144586, 2021, doi: 10.1109/ACCESS.2021.3122466.

[11] H. L. Pham, T. H. Tran, T. D. Phan, V. T. Duong Le, D. K. Lam, and Y. Nakashima, "Double SHA-256 Hardware Architecture with Compact Message Expander for Bitcoin Mining," *IEEE Access*, vol. 8, pp. 139634–139646, 2020, doi: 10.1109/ACCESS.2020.3012581.

[12] Z. Yang, A. Zhang, and Z. Mo, "PsmArena: Partitioned Shared Memory for NUMA-Awareness in Multithreaded Scientific Applications," 1007.

[13] H. Mohamed, N. Koroniotis, N. Moustafa, F. Schiliro, and A. Y. Zomaya, "Harnessing Federated Learning for Digital Forensics in IoT: A Survey and Introduction to the IoT-LF Framework," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 3161–3191, 2025, doi: 10.1109/OJCOMS.2024.3492919.

[14] M. Pourvahab and G. Ekbatanifard, "An efficient forensics architecture in software-defined networking-IoT using blockchain technology," *IEEE Access*, vol. 7, pp. 99573–99588, 2019, doi: 10.1109/ACCESS.2019.2930345.

[15] A. Mehmood, A. Shafique, M. Alawida, and A. N. Khan, "Advances and Vulnerabilities in Modern Cryptographic Techniques: A Comprehensive Survey on Cybersecurity in the Domain of Machine/Deep Learning and Quantum Techniques," *IEEE Access*, vol. 12, pp. 27530–27555, 2024, doi: 10.1109/ACCESS.2024.3367232.

[16] A. Degada and H. Thapliyal, "Single-Rail Adiabatic Logic for Energy-Efficient and CPA-Resistant Cryptographic Circuit in Low-Frequency Medical Devices," *IEEE Open Journal of Nanotechnology*, vol. 3, pp. 1–14, 2022, doi: 10.1109/OJNANO.2021.3135364.

[17] M. Kaveh and A. Falahati, "An improved Merkle hash tree based secure scheme for bionic underwater acoustic communication," *Frontiers of Information Technology and Electronic Engineering*, vol. 22, no. 7, pp. 1010–1019, Jul. 2021, doi: 10.1631/FITEE.2000043.

[18] A. Mohammed Ali and A. Kadhim Farhan, "A novel improvement with an effective expansion to enhance the MD5 hash function for verification of a secure E-Document," *IEEE Access*, vol. 8, pp. 80290–80304, 2020, doi: 10.1109/ACCESS.2020.2989050.

[19] L. Loffi, G. L. Camillo, C. A. De Souza, C. M. Westphall, and C. B. Westphall, "Management of the Chain of Custody of Digital Evidence Using Blockchain and Self-Sovereign Identities: A Systematic Literature Review," 2025, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2025.3560191.

[20] H. H. Kao, "Accelerating Multilingual Cryptocurrency Forensics: An NLP-Driven Approach for Efficient Mnemonic Identification," *IEEE Access*, vol. 13, pp. 10513–10526, 2025, doi: 10.1109/ACCESS.2025.3528829.

[21] W. A. Prabowo, F. Mohsen, and S. R. Selamat, "WhatsApp Mobile Applications in the Lens of Digital Forensics: Deciphering the Msgstore.db.crypt14 File," *Journal of Cyber Security and Mobility*, vol. 14, no. 4, pp. 823–848, Oct. 2025, doi: 10.13052/jcsm2245-1439.1443.

[22] J. Oh, S. Lee, and H. Hwang, "Forensic Detection of Timestamp Manipulation for Digital Forensic Investigation," *IEEE Access*, vol. 12, pp. 72544–72565, 2024, doi: 10.1109/ACCESS.2024.3395644.

[23] J. H. M. Korndorfer, A. Eleliemy, A. Mohammed, and F. M. Ciorba, "LB4OMP: A Dynamic Load Balancing Library for Multithreaded Applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 830–841, Apr. 2022, doi: 10.1109/TPDS.2021.3107775.

[24] T. Mary and C. Sreeja, "Adversarial Shadows in Digital Forensics: New Insights into File Fragment Classification Vulnerabilities and Defenses," *IEEE Access*, pp. 1–1, Jan. 2026, doi: 10.1109/access.2026.3655822.

[25] H. Teper, D. Kuhse, M. Gunzel, G. von der Bruggen, F. Howar, and J. J. Chen, "Thread Carefully: Preventing Starvation in the ROS 2 Multithreaded Executor," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 3588–3599, 2024, doi: 10.1109/TCAD.2024.3446865.

[26] M. Kaveh and A. Falahati, "An improved Merkle hash tree based secure scheme for bionic underwater acoustic communication," *Frontiers of Information Technology and Electronic Engineering*, vol. 22, no. 7, pp. 1010–1019, Jul. 2021, doi: 10.1631/FITEE.2000043.