Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.4, August 2026, pp. 3597~3610
e-ISSN: 2548-6861

Efficiency Without Security Trade-off: Statistically Validated Cryptographic Hash Selection for Ethereum Fraud Detection

Ahmad Dani [1]*, Wildanil Ghozi [2]*, Ramadhan Rakhmat Sani [3], Fauzi Adi Rafrastara [4]***
* Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro
** Information System, Faculty of Computer Science, Universitas Dian Nuswantoro
111202214296@mhs.dinus.ac.id [1], wildanil.ghozi@dsn.dinus.ac.id [2], ramadhan_rs@dsn.dinus.ac.id [3], fauziadi@dsn.dinus.ac.id [4]

Article Info

Article history:

Received 2026-06-30
Revised 2026-07-28
Accepted 2026-08-10

Keywords:

Avalanche Effect,
Cryptographic Hash Function,
Ethereum Fraud Detection,
Multi-Criteria Ranking,
Statistical Equivalence Testing.

ABSTRACT

Cryptographic hash functions form the preprocessing layer of Ethereum fraud detection pipelines, yet prior research evaluated hash performance and fraud detection as separate concerns, leaving practitioners without evidence-based guidance for selecting an algorithm that balances speed, energy efficiency, and cryptographic security. This study benchmarks SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3 on a stratified sample of 6,396 fraud-labeled transactions from a public 9,841-record Ethereum dataset, measuring execution time, throughput, energy consumption, and avalanche effect. Its novelty is, to the best of our knowledge, the first inferential validation, on real fraud-labeled data, that the four algorithms are cryptographically equivalent in diffusion strength—combining non-parametric testing (Shapiro-Wilk, Mann-Whitney U, Kruskal-Wallis) with formal two one-sided equivalence tests (TOST) and replicating the result on an independent 454,289-transaction dataset—establishing that selection can be decided on efficiency grounds alone without sacrificing security. BLAKE2s-256 achieved the lowest execution time (0.65 µs), highest throughput (≈1.54 million ops/s), and lowest energy (0.029 mJ/op), while all algorithms converged near the 50% avalanche ideal with no significant security difference ($H = 3.03$, $p = 0.387$). BLAKE2s-256 attained the highest composite score (99.98/100) and, in an end-to-end pipeline test, improved batch screening throughput by up to 11.5%, confirming it as optimal for off-chain Ethereum fraud-detection preprocessing on processors without SHA hardware acceleration.

I. INTRODUCTION

A cryptographic hash function is a one-way algorithm that maps input of arbitrary length to a fixed-size digest, such that any modification produces a substantially different output; to be secure, it must satisfy determinism, pre-image and second pre-image resistance, collision resistance, and the avalanche effect [1], [2]. These properties make hash functions fundamental to data integrity, digital signatures, and blockchain networks, where each block is cryptographically linked to its predecessor, rendering retrospective tampering detectable [3], [4].

The growing reliance on Ethereum for financial transactions has expanded the attack surface for fraud. The FBI IC3 reported cryptocurrency fraud losses of USD 5.6 billion in 2023, a 45% increase over 2022 [5]; Ethereum was the most targeted blockchain in H1 2024 (222 incidents, USD 315 million) [6] and recorded the highest number of security incidents of any chain across full-year 2024, with USD 748 million in losses [7]. This trend persisted into H1 2025, when USD 1.63 billion of the USD 2.47 billion total Web3 losses was stolen from Ethereum [8]. Efficient and reliable cryptographic preprocessing is therefore no longer optional but a systemic necessity for Ethereum security infrastructure.

Despite the effectiveness of machine learning-based Ethereum fraud detection [9], the cryptographic preprocessing layer—the hash function encoding transaction

payloads prior to classification—remains understudied, having been evaluated separately from detection accuracy [10]. Yet in high-frequency environments processing thousands of transactions per second, even microsecond-level latency differences constrain real-time screening feasibility [11].

Within an off-chain fraud-detection pipeline, the hash layer performs three architectural functions upstream of classification. First, integrity binding: the digest of the canonical transaction payload provides a tamper-evident fingerprint, allowing records passed between ingestion, feature-extraction, and classification services to be verified without re-transmitting the payload [3]. Second, deduplication: digest comparison detects repeated payloads at constant lookup cost, preventing redundant classification work. Third, privacy-preserving indexing: digests serve as fixed-length join and cache keys across monitoring components without exposing raw transaction attributes. Because every transaction entering the detection system passes through this layer before classification [11], its per-operation latency bounds the ingestion capacity of the entire pipeline—which motivates the benchmarking focus of this study.

Relevant prior studies fall into three streams. The first benchmarks hash algorithms: Sevin and Osman Mohammed [12] showed BLAKE2s-256 outperforming SHA-2 and SHA-3 families, corroborated by Sorescu et al. [13] across IoT platforms, while Radhakrishnan et al. [14] validated that TDP-based energy estimates correlate monotonically with execution time. The second concerns Ethereum fraud detection: Aziz et al. [9] proposed an LGBM classifier on the same public dataset used in this study, while Umer et al. [15], Ashfaq et al. [11] and Abdiwi [10] identified preprocessing latency as a key bottleneck constraining real-time classification. The third addresses diffusion and multi-criteria evaluation: Upadhyay et al. [1] and Vaughn and Borowczak [16] confirmed that SHA-2, SHA-3, and BLAKE family algorithms converge near the 50% Strict Avalanche Criterion (SAC), Purimahua et al. [17] reported BLAKE3 hashing 3.2× faster than SHA-512 with 65% lower energy in Ethereum smart contract benchmarks, and Thakur et al. [18] and Chen [19] demonstrated that weighted multi-criteria ranking provides more actionable guidance than single-metric comparison.

Although these streams have advanced independently, to the best of the authors' knowledge no prior study has inferentially validated the cryptographic equivalence of competing hash algorithms on real fraud-labeled Ethereum data as the basis for algorithm selection. The closest study, Purimahua et al. [17], targets smart-contract gas optimization rather than fraud-detection preprocessing, evaluates only a SHA-512–BLAKE3 pairing that excludes BLAKE2s-256 and SHA3-256, and reports descriptive averages without inferential testing. Consequently, no existing work establishes—on real fraud-labeled data and with statistical rigor—whether hash algorithm selection for this layer can be decided on efficiency grounds alone without compromising security, leaving practitioners without evidence-based guidance.

This study addresses that gap by benchmarking SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3 on a publicly available, fraud-labeled Ethereum dataset (9,841 records) across five dimensions—execution time, peak isolation latency, throughput, theoretical energy consumption, and avalanche effect—validated with Shapiro-Wilk, Mann-Whitney U, Kruskal-Wallis, and formal equivalence (TOST) tests, extended with output-distribution uniformity and empirical collision analysis, and integrated into a weighted composite score whose robustness is verified through weight-sensitivity analysis. It is guided by three research questions. RQ1: Which algorithm achieves the best execution time, throughput, and energy efficiency on real fraud-labeled Ethereum payloads? RQ2: Do execution time and avalanche behavior differ significantly between Normal and Fraud classes and across the four algorithms under non-parametric testing, and does any observed security difference reach practical significance? RQ3: Which algorithm is optimal under a weighted multi-criteria ranking reflecting the operational priorities of fraud-detection preprocessing?

This study makes three contributions. First, and as its primary novelty, it provides, to the best of our knowledge, the first inferential validation—combining Shapiro-Wilk, Mann-Whitney U, and Kruskal-Wallis testing with formal two one-sided equivalence tests (TOST) on a stratified sample of 6,396 real fraud-labeled Ethereum transactions, and replicating the result on an independent 454,289-transaction dataset—that the four algorithms are cryptographically equivalent in diffusion strength, establishing that algorithm selection for this layer can be decided on efficiency grounds alone without sacrificing security. This conclusion is unattainable by any prior stream in isolation, as it simultaneously requires measuring avalanche behavior, statistically validating its equivalence, and weighing it against performance criteria on real data. Second, it establishes a statistical validation protocol distinguishing statistically significant from practically meaningful differences, offering a replicable template for cryptographic benchmarking. Third, by integrating all metrics into a weighted composite score, it delivers dataset-grounded decision support enabling practitioners to select an algorithm without independent benchmarking.

II. METHOD

This chapter describes the empirical-experimental benchmarking methodology used to evaluate four cryptographic hash algorithms — SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3 — within an off-chain preprocessing pipeline for Ethereum fraud detection. Prior studies have evaluated hash function performance in

isolation [12], [13] or focused on machine learning models with limited attention to the cryptographic preprocessing layer [9], [10]. This study addresses that gap by simultaneously evaluating computational performance, energy efficiency, and cryptographic diffusion strength via the Strict Avalanche Criterion (SAC), alongside a multi-criteria weighted ranking scheme to provide actionable algorithm selection guidance.

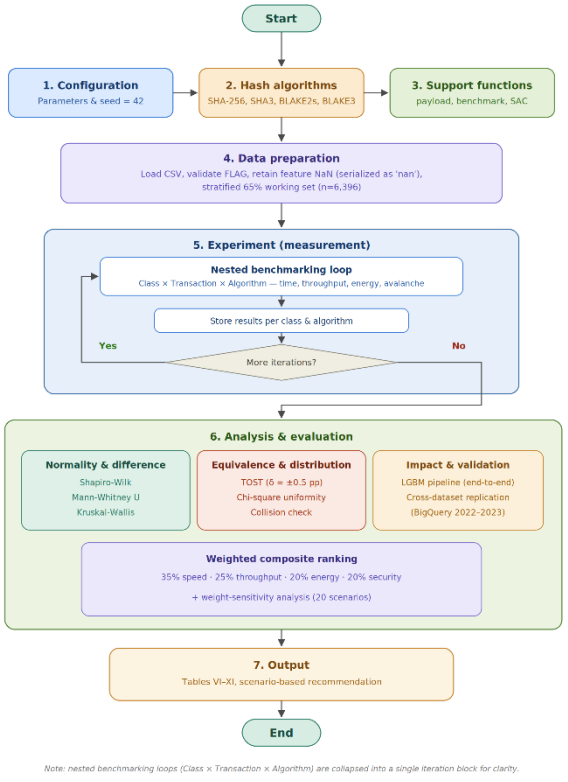

Figure 1. Experiment Process and Systematic Workflow

A. Operational Research Stages

The experimental workflow of this study follows a systematic multi-stage process as illustrated in Figure 1.

1) Dataset Acquisition and Preparation

The Ethereum transaction dataset used in this study is a publicly available fraud-labeled dataset comprising 9,841 transaction records described by 51 features, including a binary label (FLAG) distinguishing normal transactions (FLAG = 0) from fraudulent ones (FLAG = 1); for machine-learning-based Ethereum fraud detection. Records with missing FLAG values were removed and the column was cast to integer type to support subsequent operations, yielding the retained set used for sampling. No feature-level imputation was applied. Of the nine payload features selected for hashing (Section II-A-3), eight contained no missing values in the retained records; the Total ERC20 tnxs feature contained 541 missing values within the working set, which were serialized as the literal string ‘nan’ in the pipe-delimited payload—preserving payload determinism and structural consistency without imputation.

2) Stratified Sampling

A 65% stratified subsample of the retained dataset was drawn using scikit-learn's train_test_split with stratification on the FLAG column and a fixed random seed (random_state = 42) to ensure full reproducibility. This preserved the original class distribution and yielded a working set of 6,396 transactions (4,980 normal; 1,416 fraudulent). The 65:35 split ratio was selected to retain a sufficiently large fraud sample (1,416 records) for statistically meaningful comparison given the approximately 22% minority-class prevalence, while still providing a representative majority-class subset for benchmarking. This approach mitigates the biased representation that arises from the underrepresented fraud class [9]. The insensitivity of the findings to this particular sample composition is further verified through cross-dataset replication in Section III-G.

3) Payload Construction

Four cryptographic hash functions were evaluated: SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3. For each transaction record, nine features — *Address*, *Avg min between sent tnx*, *Avg min between received tnx*, *Time Diff between first and last (Mins)*, *Sent tnx*, *Received Tnx*, *avg val sent*, *total ether balance*, and *Total ERC20 tnxs* — were concatenated into a pipe-delimited string payload. These nine features were deliberately selected based on their established relevance to temporal activity patterns, value flows, and token interaction dynamics in Ethereum fraud detection literature [9], [20], while excluding redundant or highly sparse ERC20 sub-features to maintain payload consistency across records. The Address field was included to maximize payload entropy across records.

4) Performance Benchmarking

Each hash function was preceded by 1,000 warm-up iterations to eliminate cold-start artifacts [3], followed by five measurement cycles of 1,000 iterations each. The minimum mean cycle time was recorded as best latency. Energy consumption was estimated as $E = T_{best} \times P_{TDP} \times 1{,}000$, where P_{TDP} = 45 W (Intel Core i7-10750H), and throughput was derived as the reciprocal of best latency [21]. All algorithms were evaluated under identical single-threaded conditions to isolate intrinsic per-operation cost. This deliberately conservative setting reflects the per-transaction hashing pattern of preprocessing pipelines, where individual payloads are small—well below BLAKE3's 1,024-byte chunk parallelization threshold—so thread-level parallelism yields no benefit in this workload [22].

5) Avalanche Effect Measurement

Cryptographic strength was assessed via the Strict Avalanche Criterion (SAC). For each sampled payload, one character was mutated (ASCII +1 mod 128) and the normalized Hamming distance between original and mutated digests was averaged over 50 trials, with an ideal score near 50% [23]. Evaluation was applied probabilistically at a rate of *AE_LIMIT / N* (AE_LIMIT = 500), yielding approximately 500 evaluated transactions.

6) Statistical Analysis

The Shapiro-Wilk normality test confirmed non-normal distributions ($p < 0.05$) across all algorithm-class combinations, justifying the Mann-Whitney U test for Normal vs. Fraud comparisons [24]. Effect sizes were quantified via rank-biserial correlation (r) and classified as negligible ($r < 0.10$), small ($0.10 \leq r < 0.30$), medium ($0.30 \leq r < 0.50$), or large ($r \geq 0.50$). The Kruskal-Wallis H test assessed global differences across all four algorithms at $\alpha = 0.05$. Because a non-significant omnibus test cannot by itself establish equivalence, cryptographic equivalence was additionally tested formally using the two one-sided tests (TOST) procedure with an equivalence margin of ±0.5 percentage points—one percent of the 50% SAC ideal and more than five times the largest observed deviation. Pairwise TOST was applied to all six algorithm pairs, and one-sample TOST assessed each algorithm's equivalence to the theoretical 50% ideal.

7) Multi-Criteria Ranking

A composite ranking score was computed using a weighted multi-criteria scheme: computational speed (35%), throughput (25%), energy efficiency (20%), and avalanche strength (20%). The weight distribution was calibrated to reflect the operational priorities of an Ethereum fraud detection preprocessing pipeline: speed and throughput were assigned the highest combined weight (60%) because processing latency directly constrains the feasibility of real-time fraud screening in high-volume blockchain environments [12], [13]. Energy efficiency was weighted at 20% in recognition of the sustainability requirements of continuous monitoring systems [14], [25]. Avalanche strength received an equal 20% weight, reflecting its role as a necessary cryptographic baseline rather than a primary performance differentiator in preprocessing contexts [1], [16]. The robustness of the resulting ranking to this particular weight configuration is verified through a sensitivity analysis reported in Section III-E.

B. Hardware and Software

The hardware and software specifications used in this study are described to provide a clear overview of the experimental environment, as these factors may influence the baseline performance of the hashing algorithms. The hardware platform consists of a laptop equipped with an Intel Core i7-10750H processor operating at a base frequency of 2.60 GHz, 24 GB of RAM, and a 64-bit architecture. The system is further supported by an NVIDIA GeForce GTX 1650 Ti graphics processing unit. This configuration offers adequate computational capacity to conduct hashing experiments and accurately measure execution performance.

On the software side, the experiments were performed on Microsoft Windows 11 Home Single Language. The automated benchmarking pipeline was implemented in Python 3.10.7, utilizing pandas for data manipulation, hashlib for the native SHA-256, SHA3-256 (FIPS 202), and BLAKE2s-256 implementations, and the blake3 library for BLAKE3. Stratified sampling was performed with scikit-learn, statistical testing (Shapiro-Wilk, Mann-Whitney U, and Kruskal-Wallis) with scipy.stats, high-resolution timing with the time module's perf_counter, and formatted result presentation with tabulate. The complete benchmarking source code, experiment scripts, and step-by-step reproduction instructions—including all fixed random seeds and measurement parameters—are publicly available at https://github.com/Paidii-flazz/hash-benchmark-ethereum-fraud.

C. Hash Algorithms

The four candidates were selected under three explicit criteria. First, digest-length parity: all produce 256-bit outputs, ensuring that timing, energy, and diffusion comparisons are not confounded by output size—this excludes SHA-512, BLAKE2b, and truncated variants. Second, standardization status and deployment relevance: SHA-256 is the incumbent in mainstream security infrastructure; SHA3-256 is the current NIST standard built on the same KECCAK permutation family that underlies Ethereum's native hashing; BLAKE2s-256 and BLAKE3 represent the strongest published software-performance frontier among cryptographically unbroken designs, so that the set jointly spans three structurally distinct constructions (Merkle-Damgård, sponge, and tree hashing). Third, security floor: algorithms with practical collision attacks (MD5, SHA-1) and non-cryptographic hashes were excluded a priori, as an integrity-bearing preprocessing layer requires collision resistance regardless of speed. Within these criteria, the four candidates cover the design space most plausibly considered by practitioners for this layer.

SHA-256, part of the SHA-2 family, is defined in the Secure Hash Standard (FIPS 180-4) by the National Institute of Standards and Technology (NIST) [2]. It processes messages in 512-bit blocks through sixty-four computation rounds using logical bitwise operations to produce the final 256-bit output. SHA3-256 belongs to the SHA-3 family, standardized by NIST in FIPS PUB 202 (2015) [26]. It is based on the KECCAK permutation algorithm and sponge construction. The algorithm absorbs input bits into a large

internal state through multiple permutation rounds and then squeezes out the fixed-size digest, providing high resistance against length-extension attacks. BLAKE2s-256 is a cryptographic hash function derived from the BLAKE algorithm, which served as a finalist in the NIST SHA-3 competition [27]. Designed to maximize performance on 8- to 32-bit platforms, it streamlines the core compression loop using addition, rotation, and XOR operations. BLAKE3 is the latest evolution of the BLAKE tree-hashing family. It divides input strings into smaller independent chunks of 1024 bytes organized via a binary Merkle tree structure [28]. This parallel architecture enables BLAKE3 to achieve significantly higher throughput on large, multi-chunk inputs processed in multi-threaded contexts [28], although this advantage diminishes for small, single-block payloads [22].

D. Evaluation Metrics

To comprehensively assess the performance and cryptographic quality of the four hash functions (SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3), five key metrics were employed. These metrics were derived directly from the standardized benchmarking procedure illustrated in the experiment workflow (Figure 1) and implemented in the Python evaluation script. The selected metrics cover computational efficiency, stability, throughput capacity, energy footprint, and diffusion strength, providing a balanced multidimensional evaluation that directly operationalizes the three research questions: computational metrics address RQ1, class-level and cross-algorithm comparisons address RQ2, and their integration into the weighted composite score addresses RQ3.

1) Execution Time: Execution time represents the core computational latency of each hash function when processing a transaction payload. For every payload, the best latency was recorded after a warm-up phase of 1,000 iterations, followed by five measurement cycles of 1,000 iterations each. The average execution time per class is computed as:

$$\bar{T} = \frac{1}{K}\sum_{i=1}^{K} T_{best,i} \quad (1)$$

In this formulation, K represents the total number of transactions in the respective class (Normal or Fraud), and T_best,i is the minimum latency recorded for the i-th transaction. Standard deviation and median are reported to capture variability and robustness against system noise [12]; this reporting practice follows systematic benchmarking approaches established in cryptographic primitive evaluation [18].

2) Peak Isolation Latency: Peak isolation latency isolates the intrinsic performance of the algorithm by selecting the minimum execution time across measurement cycles, effectively minimizing the impact of operating system scheduling and background processes:

$$T_{best} = \min_{j=1,\ldots,5}\left(\frac{\Delta t_j}{R}\right) \quad (2)$$

Here, R = 1,000 corresponds to the number of iterations per cycle, and Δt_j is the total elapsed time recorded in cycle j. Selecting the minimum across cycles is a standard micro-benchmarking practice for cryptographic primitives [18], and has been consistently applied in comparative cryptographic performance studies to reduce the influence of OS scheduling noise [13].

3) Throughput: Throughput quantifies the processing capacity of each hash function and is defined as the number of hash operations achievable per second under optimal conditions:

$$Throughput = \frac{1}{T_{best}} \quad (3)$$

This metric is particularly important for evaluating scalability in high-throughput blockchain applications [14]. Empirical studies on blockchain-integrated IoT systems have demonstrated that throughput is a primary determinant of system feasibility in real-world deployments, where transaction rates directly constrain the practical utility of the overall architecture [29]. Furthermore, throughput and transaction latency are inherently interdependent in distributed ledger environments: as observed in blockchain-based energy trading platforms, increasing transaction volume tends to reduce per-operation latency up to a saturation point, beyond which throughput stabilizes [30]. Throughput has thus been widely adopted as a primary performance indicator in systematic cryptographic algorithm comparisons [13].

4) Theoretical Energy Consumption: Energy consumption per hashing operation is estimated using a Thermal Design Power (TDP) proxy model. With the experimental CPU having a TDP of 45 W, the energy per operation is calculated as:

$$E = T_{best} \times 45 \times 1000 \quad (4)$$

The resulting value E is expressed in millijoules per operation (mJ/op). The TDP proxy model provides a reproducible energy estimate applicable when direct power metering is unavailable. This software-level estimation approach has been adopted in cryptographic performance studies across various computing environments, including resource-constrained devices [25]. It is acknowledged that TDP-based estimation represents a theoretical upper bound;

actual per-operation energy may be lower under partial CPU utilization or thermal throttling conditions. Crucially, however, because this constant power factor is applied uniformly across all four algorithms under identical single-threaded hardware conditions, any such systematic overestimation cancels out in cross-algorithm comparison. The metric is therefore valid for the relative ranking objective of this study, which concerns comparative rather than absolute energy efficiency.

5) Avalanche Effect: The avalanche effect evaluates the diffusion property of the hash functions in accordance with the Strict Avalanche Criterion (SAC). For each evaluated payload, a single character is randomly mutated by advancing its ASCII value modulo 128. The normalized Hamming distance between the original and mutated 256-bit outputs is measured over 50 independent trials. The avalanche score is given by [23]:

$$AE = \frac{1}{50}\sum_{i=1}^{50}\left[\frac{H(B_o, B_m)}{256}\right] \times 100\% \qquad (5)$$

The notation *H(·)* refers to the Hamming distance, a standard measure for quantifying bit-level diffusion in hash function evaluation [1]. Scores approaching 50% indicate strong avalanche behavior and good resistance to differential cryptanalysis [16]. Due to computational cost, avalanche evaluation was performed on a probabilistically sampled subset of up to 500 transactions.

6) *Output-Distribution Uniformity and Collision Analysis*: Beyond the SAC, two complementary security properties were evaluated. Output-distribution uniformity was tested with a chi-square goodness-of-fit test over the byte-value histogram of all digests produced from unique payloads (256 bins, df = 255), complemented by the monobit proportion of ones, for which the ideal value is 0.5. Collision resistance was assessed empirically by counting duplicate digests over all unique payloads, applied to both the working set and, at larger scale, the full secondary validation dataset introduced in Section III-G.

III. RESULTS AND DISCUSSION

This section presents the complete experimental results of benchmarking four cryptographic hash algorithms — SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3 — across 6,396 stratified-sampled Ethereum transaction records (4,980 Normal; 1,416 Fraud). Results are organized into six sub-sections: algorithm performance, normality testing, between-class statistical comparison, cross-algorithm global comparison, multi-criteria ranking, and a synthesized discussion.

A. Benchmarking Performance per Algorithm

Tables I and II report the aggregated benchmarking outcomes for Normal and Fraud transaction classes, respectively. Each metric was obtained after applying 1,000 warm-up iterations to eliminate cold-start artefacts, followed by five measurement cycles of 1,000 iterations each; the minimum cycle mean was recorded as the best latency [18], [21].

TABLE I
BENCHMARKING PERFORMANCE RESULTS — NORMAL TRANSACTIONS

Algorithm	Avg (µs)	Std (µs)	Throughput (ops/s)	Energy (mJ/op)	Avalanche (%)
BLAKE2s-256	0.6526	0.0653	1,542,428	0.0294	49.9959
BLAKE3	1.0061	0.0936	1,000,121	0.0453	50.0394
SHA-256	1.0855	0.0939	926,154	0.0488	50.0360
SHA3-256	1.1593	0.1018	867,370	0.0522	50.0065

TABLE II
BENCHMARKING PERFORMANCE RESULTS — FRAUD TRANSACTIONS

Algorithm	Avg (µs)	Std (µs)	Throughput (ops/s)	Energy (mJ/op)	Avalanche (%)
BLAKE2s-256	0.6565	0.0145	1,523,958	0.0295	49.9982
BLAKE3	0.9941	0.0260	1,006,688	0.0447	50.0832
SHA-256	1.0857	0.0231	921,506	0.0489	49.9962
SHA3-256	1.1613	0.0242	861,547	0.0523	50.0447

Across both transaction classes, BLAKE2s-256 consistently achieved the lowest average execution time (0.6526 µs for Normal; 0.6565 µs for Fraud), the highest throughput (approximately 1.54×10^6 ops/s), and the lowest theoretical energy consumption per operation (0.0294 mJ/op). These results are aligned with BLAKE2s design

objectives: its streamlined compression loop combining addition, XOR, and bitwise rotation on 32-bit word sizes provides a structural speed advantage over SHA-based counterparts, particularly on general-purpose CPUs without hardware acceleration [27]. Independent research on blockchain-based fraud detection systems confirms that preprocessing efficiency — including hash computation speed — is a decisive factor in determining overall detection pipeline throughput under real-time constraints [11]. A recent large-scale comparison of hash algorithms for blockchain-IoT systems similarly confirmed BLAKE2s as the fastest among SHA-2 and SHA-3 family alternatives under equivalent single-threaded conditions [12].

BLAKE3 ranked second in terms of speed (1.0061 µs Normal; 0.9941 µs Fraud). Despite its parallel Merkle-tree chunk architecture designed for multi-threaded throughput gains [28], BLAKE3 exhibited approximately 54% higher latency than BLAKE2s-256 in the single-threaded execution context of this experiment, consistent with findings from comparative analyses of hash algorithm architecture, which confirm that BLAKE3's throughput advantage is primarily realized in parallel multi-core contexts and diminishes significantly under single-threaded conditions [22]. SHA-256 and SHA3-256 occupied third and fourth positions, respectively. SHA3-256's higher latency (1.1593–1.1613 µs) reflects the computational overhead of its 1,600-bit KECCAK sponge state permutation across 24 rounds, which is substantially wider than SHA-256's 512-bit Merkle-Damgård block structure [4], [31]. This performance gap is corroborated by independent blockchain-specific benchmarking, where SHA3-256 exhibited a median computation time approximately seven times higher than SHA-256 (2,100 ns vs. 300 ns) under equivalent single-threaded conditions, while SHA-256 and BLAKE2 showed no statistically significant difference in computational efficiency [19].

Standard deviation values were consistently low across all algorithms (range: 0.014–0.104 µs), confirming high measurement stability. Execution time differences between Normal and Fraud classes remained below 0.015 µs for all algorithms, indicating that transaction class membership has negligible influence on hash computation latency. This is expected, as hash function execution time depends primarily on input length rather than content [13].

The avalanche effect scores for all four algorithms consistently converged near 50% across both classes, confirming adherence to the Strict Avalanche Criterion (SAC). Specifically, BLAKE3 registered the highest avalanche score among Fraud transactions (50.08%), while BLAKE2s-256 scored 49.9959% for Normal — both within the typical SAC-compliant range reported in dedicated avalanche evaluation studies [1], [16]. These findings are further corroborated by an independent Ethereum blockchain benchmarking study, which reported avalanche scores of 50.1% for BLAKE3 and 49.8% for SHA-512 under equivalent single-input mutation conditions [17], confirming that the convergence near the 50% SAC ideal is robust across different payload types and experimental environments. The theoretical energy estimates, derived from the TDP proxy model ($E = T_{best} \times 45\ W \times 1{,}000$), follow the same ranking as execution time, as throughput and energy are deterministic monotonic transformations of latency [14], [25].

B. Shapiro-Wilk Normality Test

Prior to inferential comparison, the Shapiro-Wilk test was applied to assess the normality of execution time distributions across all algorithm-class combinations. Table III summarizes the results.

TABLE III
SHAPIRO-WILK NORMALITY TEST RESULTS (EXECUTION TIME, TIME_US)

Algorithm	Class	Shapiro W	p-value	Conclusion
BLAKE2s-256	Normal	0.50431	< 0.001	Non-normal → Mann-Whitney justified
BLAKE2s-256	Fraud	0.69092	< 0.001	Non-normal → Mann-Whitney justified
BLAKE3	Normal	0.58394	< 0.001	Non-normal → Mann-Whitney justified
BLAKE3	Fraud	0.91140	< 0.001	Non-normal → Mann-Whitney justified
SHA-256	Normal	0.56744	< 0.001	Non-normal → Mann-Whitney justified
SHA-256	Fraud	0.68256	< 0.001	Non-normal → Mann-Whitney justified
SHA3-256	Normal	0.58027	< 0.001	Non-normal → Mann-Whitney justified
SHA3-256	Fraud	0.70788	< 0.001	Non-normal → Mann-Whitney justified

All eight algorithm-class combinations produced Shapiro W statistics substantially below 1.0 and p-values approaching zero ($p < 0.001$), confirming non-normal distributions in all cases. This pattern is consistent with typical micro-benchmarking observations of cryptographic hash functions, where execution time distributions exhibit right-skewed tails due to occasional OS-level scheduling interruptions and memory access variability across both hardware and software implementations [13]. This distributional characteristic is also consistently reported in

blockchain transaction timing studies, where execution metrics tend toward non-Gaussian distributions reflecting system-level variability [4]. The confirmation of non-normality justified the subsequent use of the Mann-Whitney U non-parametric test for between-class comparisons, rather than the parametric Student's t-test, which presupposes normality [24].

C. Mann-Whitney U Test: Normal vs Fraud

Effect sizes are quantified using rank-biserial correlation (r): negligible ($r < 0.10$), small ($0.10 \leq r < 0.30$), medium ($0.30 \leq r < 0.50$), or large ($r \geq 0.50$) [24].

TABLE IV
MANN-WHITNEY U TEST RESULTS – NORMAL VS FRAUD TRANSACTIONS (A = 0.05)

Algorithm	Metric	Med. Normal	Med. Fraud	p-value	Sig?	Effect r	Level
BLAKE2s-256	Time (μs)	0.6521	0.6613	< 0.001	✓	0.1640	Small
BLAKE2s-256	Avalanche (%)	49.9766	49.9453	0.79219	–	0.0155	Negligible
BLAKE3	Time (μs)	1.0015	0.9928	< 0.001	✓	0.0706	Negligible
BLAKE3	Avalanche (%)	50.0508	50.0625	0.66211	–	0.0291	Negligible
SHA-256	Time (μs)	1.0843	1.0882	< 0.001	✓	0.1333	Small
SHA-256	Avalanche (%)	50.0391	50.0820	0.78321	–	0.0168	Negligible
SHA3-256	Time (μs)	1.1576	1.1583	< 0.001	✓	0.0527	Negligible
SHA3-256	Avalanche (%)	49.9961	50.0312	0.48018	–	0.0472	Negligible

Consistent with the rank-based nature of the Mann-Whitney U test, the execution time and avalanche values reported in Table IV are class medians, which may differ marginally from the arithmetic means reported in Tables I and II.

For execution time, all four algorithms returned statistically significant differences between Normal and Fraud classes ($p < 0.001$). BLAKE2s-256 yielded the largest practical effect size ($r = 0.1640$, Small), followed by SHA-256 ($r = 0.1333$, Small), while SHA3-256 and BLAKE3 showed negligible effects ($r = 0.0527$ and 0.0706, respectively). Despite this statistical significance — which is partly driven by the large combined sample size ($N = 6,396$) — the absolute median latency differences remained below 0.015 μs across all algorithms. Such a difference is operationally negligible in fraud detection preprocessing pipelines; empirical evaluations of real-time Ethereum anomaly detection systems confirm that per-transaction decision latencies must remain in the microsecond range to maintain effective screening throughput [9], [32]. This finding is consistent with the observation that payload structural variation between Normal and Fraud transactions does not alter the computational path of the hash function, only the exact input byte stream [15].

In contrast, avalanche effect scores showed no statistically significant differences between Normal and Fraud classes for any algorithm (p-values: 0.481–0.792), with negligible effect sizes ($r < 0.05$ for all). This result confirms that the cryptographic diffusion properties of the evaluated algorithms are invariant to transaction class membership, consistent with the deterministic and class-agnostic nature of cryptographic hash functions. The statistical independence of avalanche scores across classes is also consistent with dedicated SAC evaluation studies demonstrating that SHA-256, SHA3-256, BLAKE2, and BLAKE3 family members all converge near the 50% ideal regardless of input class [1], [16].

D. Kruskal-Wallis Test: Cross-Algorithm Comparison

To assess global performance differences across all four algorithms simultaneously, the Kruskal-Wallis H test was applied to execution time, throughput, energy consumption, and avalanche effect. Table V reports the results.

TABLE V
KRUSKAL-WALLIS H TEST – CROSS ALGORITHM PERFORMANCE DIFFERENCES

Metric	H-Statistic	p-value	Sig?
Time (μs)	20,865.10	< 0.001	✓
Throughput (ops/s)	20,865.10	< 0.001	✓
Energy (mJ/op)	20,865.10	< 0.001	✓
Avalanche (%)	3.03	0.3873	–

Execution time, throughput, and energy consumption all yielded highly significant differences across the four algorithms ($H = 20{,}865.10$, $p < 0.001$ for all three metrics). The identical H-statistic across these three metrics is mathematically expected: throughput is the reciprocal of execution time ($1/T_{best}$), and energy is a linear scaling of execution time ($T_{best} \times P_{TDP} \times 1{,}000$), so both inherit the same rank ordering as execution time. The very large H value confirms that the performance separation among BLAKE2s-256, BLAKE3, SHA-256, and SHA3-256 is not attributable to random measurement variation, but reflects genuine structural differences in their hash construction designs [2], [26].

Avalanche effect did not differ significantly across algorithms ($H = 3.03$, $p = 0.387$). This outcome aligns with theoretical expectations: all four algorithms are explicitly designed to satisfy the SAC, and despite their distinct internal architectures (Merkle-Damgård, KECCAK sponge, BLAKE compression), they converge to statistically equivalent diffusion when evaluated on finite real-world payloads [1]. This finding further validates that algorithm selection for the Ethereum fraud detection preprocessing layer may legitimately be made on efficiency grounds alone without sacrificing cryptographic integrity — a conclusion directly relevant to the broader challenge of securing Ethereum-based systems, where cryptographic primitive selection significantly impacts both security robustness and system performance [33]. Because the non-significant Kruskal-Wallis result cannot by itself establish equivalence, formal TOST equivalence testing was applied (Section II-A-6). All six pairwise comparisons rejected non-equivalence within the ±0.5 percentage-point margin on both the primary working set and the secondary validation dataset introduced in Section III-G (all $p < 0.001$), and each algorithm was individually equivalent to the theoretical 50% ideal (all $p < 0.001$). Table VI reports the pairwise results. The security-equivalence claim is therefore established by formal equivalence testing rather than inferred from a non-significant difference test.

TABLE VI
TOST EQUIVALENCE TEST RESULTS – PAIRWISE AVALANCHE COMPARISON ($\delta = \pm 0.5$ PP)

Pair	Δ primary (pp)	p (TOST)	Δ secondary (pp)	p (TOST)
BLAKE2s-256 vs BLAKE3	−0.0572	<0.001	−0.0188	<0.001
BLAKE2s-256 vs SHA-256	−0.0415	<0.001	+0.0004	<0.001
BLAKE2s-256 vs SHA3-256	−0.0297	<0.001	−0.0321	<0.001
BLAKE3 vs SHA-256	+0.0157	<0.001	+0.0192	<0.001
BLAKE3 vs SHA3-256	+0.0275	<0.001	−0.0133	<0.001
SHA-256 vs SHA3-256	+0.0118	<0.001	−0.0325	<0.001

The extended security evaluation (Section II-D-6) confirms distributional soundness. All four algorithms passed byte-level chi-square uniformity on the primary working set ($p = 0.30$–0.85), with monobit proportions within ±0.001 of the ideal 0.5, and produced zero collisions among all unique payloads. On the secondary dataset, a single nominally significant chi-square value for SHA3-256 ($p = 0.0032$) was identified as sampling variability: re-testing over the full 454,289-payload population yielded $p = 0.346$, and zero collisions were observed even at that scale. Table VII summarizes the results.

TABLE VII
OUTPUT-DISTRIBUTION UNIFORMITY AND EMPIRICAL COLLISION RESULSTS

Algorithm	χ^2 primary (p)	χ^2 secondary (p)	Monobit (bit-1)	Collisions
SHA-256	255.4 (0.482)	260.0 (0.402)	0.4994–0.4999	0
SHA3-256	249.0 (0.595)	320.9 (0.003)*	0.5002–0.5009	0
BLAKE2s-256	231.9 (0.848)	255.6 (0.478)	0.5000–0.5001	0
BLAKE3	266.3 (0.300)	271.0 (0.235)	0.4996–0.5002	0

Collisions counted over all unique payloads in each 6,396-transaction sample; zero collisions also observed over the full 454,289-payload secondary population. * Sampling variability; full-population re-test: $\chi^2 = 263.3$ ($df = 255$), $p = 0.346$.

E. Multi-Criteria Weighted Ranking

A composite weighted ranking integrates the four performance dimensions into a single actionable recommendation. The weight distribution — computational speed (35%), throughput (25%), energy efficiency (20%),

and avalanche security (20%) — was calibrated to reflect operational priorities of high-volume Ethereum fraud detection preprocessing pipelines, where latency and throughput jointly constrain real-time screening feasibility [12], [30]. Energy efficiency received a 20% weight in recognition of sustainability requirements inherent to continuous blockchain monitoring systems [14], [25], while avalanche strength was equally weighted at 20% as a mandatory cryptographic baseline rather than a primary performance differentiator [1]. Table VIII presents the normalized sub-scores and final rankings.

TABLE VIII
MULTI-CRITERIA WEIGHTED RANKING OF HASH ALGORITHMS (WEIGHTED SCORE / 100)

Rank	Algorithm	Speed (35%)	Throughput (25%)	Energy (20%)	Security (20%)	Total Score
1	BLAKE2s-256	100.00	100.00	100.00	99.90	99.98
2	BLAKE3	65.13	65.11	65.14	100.00	72.10
3	SHA-256	60.20	60.14	60.20	99.96	68.14
4	SHA3-256	56.35	56.30	56.35	99.93	65.05

BLAKE2s-256 achieved the highest composite score (99.98/100), attaining the maximum sub-scores across speed, throughput, and energy dimensions. Its marginal avalanche sub-score deficit (99.90 vs. 100.00 for BLAKE3) had a negligible impact on the final ranking, given that all algorithms cluster near the theoretical SAC ideal of 50%. BLAKE3 ranked second (72.10/100), achieving the maximum security sub-score but incurring approximately 54% higher latency than BLAKE2s-256, which penalized its speed and throughput sub-scores significantly. SHA-256 and SHA3-256 followed in third and fourth positions (68.14 and 65.05, respectively), with SHA3-256's wider internal state yielding consistently lower sub-scores across all performance dimensions.

These rankings are consistent with findings from recent multi-platform comparative benchmarking of cryptographic hash algorithms, which have similarly identified BLAKE2s as the preferred option for speed-constrained applications compared to SHA-2 and SHA-3 family candidates [13]. The throughput-sensitive nature of such preprocessing stages is further underscored by empirical Ethereum fraud detection pipelines, where classification systems must process high transaction volumes without introducing detection latency [34]. This alignment is further reinforced by studies on blockchain-based fraud detection systems, which demonstrate that hash computation latency directly constrains real-time classification throughput in preprocessing pipelines [11], [15]. Notably, an independent evaluation of hash algorithms in Proof-of-Work blockchain contexts confirmed that SHA-2 and BLAKE2 exhibit equivalent security characteristics — including hash value heterogeneity and collision metrics — while both outperform SHA-3 in computational efficiency [19]. Furthermore, direct Ethereum smart contract benchmarking demonstrated that BLAKE3 achieves 3.2× faster hashing than SHA-512 (0.108 μs vs. 0.280 μs) and 65% lower energy consumption per operation, with avalanche scores for both algorithms converging near 50% [17]. The result also aligns with empirical evaluations in blockchain environments, where BLAKE2-family algorithms have demonstrated superior throughput-to-energy ratios relative to SHA-256 under equivalent workloads [12], [29].

To verify that the composite ranking does not depend on the particular weight configuration (Section II-A-7), a sensitivity analysis perturbed each criterion weight by ±10 and ±15 percentage points (redistributing the remainder proportionally) and evaluated three extreme scenarios (uniform 25/25/25/25; security-dominant 20/15/15/50; security-extreme 10/10/10/70). BLAKE2s-256 retained rank 1 in all 20 scenarios, and the complete ordering (BLAKE2s-256 > BLAKE3 > SHA-256 > SHA3-256) was invariant throughout; a critical-threshold search shows BLAKE3 overtakes BLAKE2s-256 only when the security weight reaches 99.8%, a configuration that effectively eliminates all performance criteria. Table IX summarizes representative scenarios; the recommendation is therefore robust to any defensible weighting choice.

TABLE IX
WEIGHT-SENSITIVITY ANALYSIS OF THE COMPOSITE RANKING (SCORE AND RANK)

Scenario	BLAKE2s-256	BLAKE3	SHA-256	SHA3-256
Speed ±10/±15 pp	#1 (99.98)	#2 (70.5–73.7)	#3 (66.3–70.0)	#4 (63.1–67.1)
Throughput ±10/±15 pp	#1 (99.98)	#2 (70.7–73.5)	#3 (66.5–69.7)	#4 (63.3–66.8)
Energy ±10/±15 pp	#1 (99.98)	#2 (70.8–73.4)	#3 (66.7–69.6)	#4 (63.4–66.7)
Security ±10/±15 pp	#1 (99.97–100)	#2 (66.9–77.3)	#3 (62.2–74.1)	#4 (58.5–71.6)

Scenario	BLAKE2s-256	BLAKE3	SHA-256	SHA3-256
Uniform 25/25/25/25	#1 (99.97)	#2 (73.84)	#3 (70.12)	#4 (67.23)
Security-dominant 50%	#1 (99.95)	#2 (82.56)	#3 (80.07)	#4 (78.13)
Security-extreme 70%	#1 (99.93)	#2 (89.54)	#3 (88.03)	#4 (86.85)

F. Impact on the End-to-End Detection Pipeline

To quantify how hash selection propagates to the complete detection pipeline, the benchmarked hashing stage was integrated with an LGBM classifier trained on the working set following Aziz et al. [9] (accuracy = 0.9633, F1 = 0.9134, AUC = 0.9878 on a stratified 20% hold-out). Detection metrics are by construction identical across hash choices, since the digest serves as an integrity and deduplication layer and does not alter classifier features; hashing in fact exposed 10 duplicate payloads among the 6,396 records, empirically illustrating its deduplication role. Two operational modes were measured on the reference platform. In per-transaction screening, classifier inference dominates end-to-end latency (≈1,041 µs per transaction), bounding the hash-attributable share at 0.07–0.13% and rendering the throughput difference across hash choices negligible (959–960 tx/s). In batch (vectorized) screening—representative of high-volume monitoring—per-transaction inference cost drops to 4.4 µs, raising the hash share to 14.2–23.1% of end-to-end latency; selecting BLAKE2s-256 over SHA3-256 then improves total pipeline throughput by 11.5% (194,644 vs 174,530 tx/s), and by 10.7% over SHA-256 (Table X). These results bound the practical significance of hash selection precisely: it does not materially alter single-transaction ML screening latency, but it substantially governs total throughput in batch screening and the capacity of the ingestion layer itself, where the 1.67–1.78× differences of Section III-A apply directly.

TABLE X
END-TO-END PIPELINE IMPACT OF HASH SELECTION (BATCH SCREENING MODE)

Algorithm	Hash (µs/tx)	E2E batch (µs/tx)	Throughput (tx/s)	Hash share (%)
BLAKE2s-256	0.730	5.138	194,644	14.21
BLAKE3	0.836	5.243	190,720	15.94
SHA-256	1.279	5.687	175,846	22.49
SHA3-256	1.322	5.730	174,530	23.07

G. Cross-Dataset and Cross-Environment Validation

To assess external validity, the full benchmarking protocol was replicated on an independent secondary dataset: 454,289 raw Ethereum transactions extracted via Google BigQuery covering October 2022–March 2023, in which 22,733 transactions were fraud-labeled through 308 verified fraudulent addresses. A stratified sample of 6,396 transactions (seed = 42) was drawn under the identical protocol, with pipe-delimited payloads constructed from six transaction-level fields (mean length ≈ 201 bytes, versus ≈ 91 bytes in the primary dataset). Three findings emerge (Table XI). First, the security-equivalence result replicates fully: all algorithms again converge near the 50% SAC ideal with no statistically significant cross-algorithm difference (Kruskal-Wallis, primary: H = 3.185, p = 0.364; secondary: H = 2.919, p = 0.404), on transaction data from a different period, source, and schema. Second, two ordering properties are invariant across datasets: SHA3-256 is consistently slowest, and the BLAKE family consistently occupies the two fastest positions. Third, the top position within the BLAKE family is payload-length-dependent: BLAKE2s-256 leads on the short primary payloads (0.5634 vs 0.6576 µs), whereas BLAKE3 overtakes it on the roughly twice-longer secondary payloads (0.7438 vs 0.8647 µs). This crossover is consistent with the designs' round structures—BLAKE3's 7-round compression amortizes over multiple input blocks, while BLAKE2s's lower per-call overhead dominates on short, single-block inputs—and empirically locates the crossover between approximately 91 and 201 bytes; the recommendation of BLAKE2s-256 therefore holds specifically for the short feature-vector payloads characteristic of fraud-detection preprocessing, which is precisely the operating regime of this study. Additionally, replication in a second execution environment (a server-class Xeon exposing the SHA-NI instruction-set extension) reveals one further scope condition: with SHA-NI, hardware-accelerated SHA-256 becomes the fastest algorithm, whereas on processors without SHA extensions—including the reference i7-10750H (Comet Lake)—the software-level ranking above holds; the security-equivalence result is invariant to this hardware axis. Note that Table XI latencies reflect the core digest computation on pre-encoded byte payloads and are therefore systematically lower than the Table I–II values, which additionally include UTF-8 encoding and hexadecimal rendering; orderings are unaffected within each measurement scope.

TABLE XI
CROSS-DATASET VALIDATION – PRIMARY VS SECONDARY (2022-2023) DATASET

Dataset	Algorithm	Avg (µs)	Std (µs)	Throughput (ops/s)	Avalanche (%)
Primary	BLAKE2s-256	0.5634	0.0107	1,774,859	49.9787
(n=6,396;	BLAKE3	0.6576	0.0238	1,520,586	50.0358
H=3.185,	SHA-256	0.9600	0.0173	1,041,669	50.0201
p=0.364)	SHA3-256	1.0459	0.0233	956,157	50.0083
Secondary	BLAKE3	0.7438	0.0215	1,344,506	50.0022
(n=6,396;	BLAKE2s-256	0.8647	0.0158	1,156,419	49.9835
H=2.919,	SHA-256	1.2094	0.0276	826,855	49.9831
p=0.404)	SHA3-256	1.3661	0.0197	732,002	50.0156

H. Discussion

The results of this study yield three principal findings, each corresponding to one research question. First, addressing RQ1**,** algorithm choice has a statistically confirmed and practically meaningful impact on preprocessing throughput. BLAKE2s-256 demonstrated a throughput advantage of approximately 1.78× over SHA3-256 and 1.67× over SHA-256 under real Ethereum transaction workloads — differences that directly bound system feasibility in high-frequency fraud monitoring deployments [9], [15]. This aligns with prior Ethereum smart contract benchmarking reporting a comparable throughput differential between BLAKE3 and SHA-512 [17].

Second, addressing RQ2**,** cryptographic security is effectively equivalent across all four algorithms, with no deviation exceeding 0.09 percentage points from the 50% SAC ideal, confirmed at the population level by the Kruskal-Wallis test ($H = 3.03$, $p = 0.387$), established formally by TOST equivalence testing on both datasets (Table VI) together with output-distribution uniformity and zero empirical collisions (Table VII), and corroborated by independent entropy-based hash evaluation [19]. This implies that algorithm selection may legitimately be decided on performance grounds alone, without incurring any cryptographic security trade-off [33], [34].

Third, also under RQ2, execution time differences between Normal and Fraud classes remained below 0.015 µs across all algorithms, confirming that the preprocessing layer introduces no class-dependent latency bias — an important property for avoiding timing side-channels in production systems [35], with the observed statistical significance ($p < 0.001$) attributable primarily to sample size rather than practical effect [24].

In answer to RQ3, BLAKE2s-256 is the optimal cryptographic hash algorithm for off-chain Ethereum fraud detection preprocessing, achieving the highest composite score (99.98/100). Notably, the practical implication of the second finding — that algorithm selection can be decided on performance grounds alone without a security trade-off — could not have been established by any of the three prior research streams in isolation, as it requires simultaneously measuring diffusion behavior, validating its equivalence inferentially, and weighing it against performance criteria on real fraud-labeled data. Key limitations include the use of a TDP-based energy proxy rather than direct hardware measurement [18], [36], and the restriction to single-threaded execution — future work should evaluate BLAKE3 under multi-threaded conditions [17], [22], [28], and extend benchmarking to a complete end-to-end fraud detection pipeline [10], [19], [37].

Practically, these findings translate into scenario-conditional guidance. For throughput-constrained deployments on commodity CPUs without SHA hardware extensions—including the mobile-class and pre-Ice-Lake processors typical of edge and mid-range deployments—BLAKE2s-256 is recommended for short feature-vector payloads (up to roughly 100–150 bytes, the regime of this study). Once payloads grow beyond the empirically located crossover (between approximately 91 and 201 bytes; Section III-G), or where multi-core parallelism is available for batch reprocessing, BLAKE3 becomes the faster choice. On processors exposing SHA-NI acceleration (Intel Ice Lake and later, AMD Zen family, and most cloud vCPUs), hardware-accelerated SHA-256 becomes the fastest option and additionally satisfies FIPS 180-4 compliance, making it the default for regulated environments. SHA3-256 is indicated where sponge-based standard diversity or structural resistance to length-extension is an explicit requirement, accepting the measured throughput penalty. Because security is statistically equivalent across all candidates, these recommendations rest purely on operational grounds without a cryptographic trade-off.

IV. CONCLUSION

This study benchmarked SHA-256, SHA3-256, BLAKE2s-256, and BLAKE3 within an *off-chain* Ethereum fraud-detection preprocessing pipeline to determine which algorithm best balances computational efficiency and cryptographic strength for real-time deployment, and to establish—inferentially and on real fraud-labeled data—

whether that choice can be made without compromising security.

Three principal conclusions emerge, each answering one research question. First, in answer to RQ1, the choice of hash algorithm materially shapes preprocessing throughput: BLAKE2s-256 achieved the lowest execution time and highest throughput, with statistically significant and practically meaningful separation across the four algorithms. Second, in answer to RQ2, cryptographic security is effectively equivalent across all algorithms—avalanche scores converge near the ideal 50% SAC regardless of algorithm or transaction class, with no statistically significant cross-algorithm difference ($H = 3.03$, $p = 0.387$), a conclusion strengthened by formal TOST equivalence testing and replicated on an independent 454,289-transaction dataset—while execution-time latency is invariant to transaction class, confirming the preprocessing layer as an unbiased stage free of class-dependent timing behavior. Third, in answer to RQ3, BLAKE2s-256 is identified as the optimal algorithm for this layer on processors without SHA hardware acceleration—which include the reference platform of this study and the commodity and edge devices typical of distributed monitoring—attaining the highest weighted multi-criteria composite score (99.98/100), a ranking shown to be robust across all tested weight configurations.

The principal novelty of this study is the first inferential demonstration, on real fraud-labeled Ethereum transactions, that competing hash algorithms are cryptographically equivalent in diffusion strength—formally confirmed through equivalence (TOST) testing and reinforced by output-distribution uniformity and empirical collision analysis on two independent datasets—establishing that algorithm selection for the fraud-detection preprocessing layer can legitimately be decided on efficiency grounds alone without sacrificing security. This conclusion could not have been reached by any of the three prior research streams in isolation, as it requires simultaneously measuring diffusion behavior, validating its equivalence statistically, and weighing it against performance criteria on real data. Complementing this, the study delivers a reproducible, multi-criteria benchmarking framework—with the full experimental configuration and benchmarking parameters detailed in Section II, and the complete source code, experiment scripts, and reproduction instructions publicly available—that gives practitioners dataset-grounded guidance for selecting cryptographic hash algorithms in high-throughput blockchain security systems.

This study is limited by its reliance on a TDP-based theoretical energy proxy rather than direct hardware power measurement, and by its restriction to single-threaded CPU execution, which may understate the throughput potential of BLAKE3's parallel architecture. Future work should validate these findings through direct power metering and multi-threaded or GPU-accelerated execution, extend evaluation to resource-constrained ARM and embedded IoT devices, and scale the end-to-end pipeline integration reported here to production-scale streaming transaction volumes under sustained load. Overall, this study provides a reproducible empirical foundation for selecting cryptographic hash algorithms in high-throughput Ethereum fraud-detection systems.

REFERENCES

[1] D. Upadhyay, N. Gaikwad, M. Zaman, and S. Sampalli, "Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications," *IEEE Access*, vol. 10, pp. 112472–112486, 2022, doi: 10.1109/ACCESS.2022.3215778.

[2] National Institute of Standards and Technology (US), "Secure hash standard," National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 180-4, 2015. doi: 10.6028/NIST.FIPS.180-4.

[3] O. Kuznetsov, A. Rusnak, A. Yezhov, D. Kanonik, K. Kuznetsova, and S. Karashchuk, "Enhanced Security and Efficiency in Blockchain with Aggregated Zero-Knowledge Proof Mechanisms," *IEEE Access*, pp. 1–1, 2024, doi: 10.1109/ACCESS.2024.3384705.

[4] M. Ul Hassan, M. H. Rehmani, and J. Chen, "Anomaly Detection in Blockchain Networks: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 289–318, 2023, doi: 10.1109/COMST.2022.3205643.

[5] "2023 Cryptocurrency Fraud Report Released," Federal Bureau of Investigation. Accessed: Jun. 30, 2026. [Online]. Available: https://www.fbi.gov/news/stories/2023-cryptocurrency-fraud-report-released

[6] J. Coker, "Over $1bn in Cryptocurrency Lost to Web3 Cyber Incidents in 2024," Infosecurity Magazine. Accessed: Jun. 30, 2026. [Online]. Available: https://www.infosecurity-magazine.com/news/crypto-lost-web3-cyber-incidents/

[7] "Hack3d: The Web3 Security Report 2024 - CertiK." Accessed: Jul. 06, 2026. [Online]. Available: https://www.certik.com/blog/hack3d-the-web3-security-report-2024

[8] "Hack3d: The Web3 Security Quarterly Report - Q2 + H1 2025 - CertiK." Accessed: Jul. 06, 2026. [Online]. Available: https://www.certik.com/blog/hack3d-the-web3-security-quarterly-report-q2-h1-2025

[9] R. M. Aziz, M. F. Baluch, S. Patel, and A. H. Ganie, "LGBM: a machine learning approach for Ethereum fraud detection," *Int. j. inf. tecnol.*, vol. 14, no. 7, pp. 3321–3331, Dec. 2022, doi: 10.1007/s41870-022-00864-6.

[10] F. G. Abdiwi, "Detection of Digital Currency Fraud through a Distributed Database Approach and Machine Learning Model," *TEM Journal*, pp. 3025–3039, Dec. 2024, doi: 10.18421/TEM134-37.

[11] T. Ashfaq *et al.*, "A Machine Learning and Blockchain Based Efficient Fraud Detection Mechanism," *Sensors*, vol. 22, no. 19, p. 7162, Sep. 2022, doi: 10.3390/s22197162.

[12] A. Sevin and A. A. Osman Mohammed, "Comparative Study of Blockchain Hashing Algorithms with a Proposal for HashLEA," *Applied Sciences*, vol. 14, no. 24, p. 11967, Dec. 2024, doi: 10.3390/app142411967.

[13] T.-G. Sorescu, V.-M. Chiriac, M.-A. Stoica, C.-R. Comsa, I.-G. Soroaga, and A. Contac, "Comparative Performance Analysis of Lightweight Cryptographic Algorithms on Resource-Constrained IoT Platforms," *Sensors*, vol. 25, no. 18, p. 5887, Sep. 2025, doi: 10.3390/s25185887.

[14] I. Radhakrishnan, S. Jadon, and P. B. Honnavalli, "Efficiency and Security Evaluation of Lightweight Cryptographic Algorithms for

Resource-Constrained IoT Devices," *Sensors*, vol. 24, no. 12, p. 4008, Jun. 2024, doi: 10.3390/s24124008.

[15] Q. Umer, J.-W. Li, M. R. Ashraf, R. N. Bashir, and H. Ghous, "Ensemble Deep Learning-Based Prediction of Fraudulent Cryptocurrency Transactions," *IEEE Access*, vol. 11, pp. 95213–95224, 2023, doi: 10.1109/ACCESS.2023.3310576.

[16] R. Vaughn and M. Borowczak, "Strict Avalanche Criterion of SHA-256 and Sub-Function-Removed Variants," *Cryptography*, vol. 8, no. 3, p. 40, Sep. 2024, doi: 10.3390/cryptography8030040.

[17] I. D. A. Purimahua, S. A. Wibowo, and Y. Purwanto, "BLAKE3 and SHA-512 Combination Performances Analysis in Blockchain Mechanism and Optimization Smart Contract on D'Apps," in *2025 IEEE International Conference on Networking, Intelligent Systems, and IoT (ICONS-IoT)*, Bandung, Indonesia: IEEE, Aug. 2025, pp. 158–163. doi: 10.1109/ICONS-IoT65216.2025.11211127.

[18] S. Thakur, S. Banik, and F. Regazzoni, "Energy Analysis of Cryptographic Algorithms in Server Environment," in *Proceedings of the 2024 on Cloud Computing Security Workshop*, Salt Lake City UT USA: ACM, Nov. 2024, pp. 3–14. doi: 10.1145/3689938.3694775.

[19] A. C. H. Chen, "Report on Hash Algorithm Performance ─ A Case Study of Cryptocurrency Exchanges Based on Blockchain System," in *2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI)*, Tirunelveli, India: IEEE, Nov. 2024, pp. 30–35. doi: 10.1109/ICDICI62993.2024.10810832.

[20] R. M. Aziz, M. F. Baluch, S. Patel, and P. Kumar, "A Machine Learning based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes," *Karbala International Journal of Modern Science*, vol. 8, no. 2, pp. 139–151, May 2022, doi: 10.33640/2405-609X.3229.

[21] M. Horro, L.-N. Pouchet, G. RodriDguez, and J. Tourino, "MARTA: Multi-configuration Assembly pRofiler and Toolkit for performance Analysis," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Singapore, Singapore: IEEE, May 2022, pp. 79–89. doi: 10.1109/ISPASS55109.2022.00008.

[22] M. Al-Zubaidie, "Implication of Lightweight and Robust Hash Function to Support Key Exchange in Health Sensor Networks," *Symmetry*, vol. 15, no. 1, p. 152, Jan. 2023, doi: 10.3390/sym15010152.

[23] E. M. Horra, A. M. Beyene, and S. Y. Techan, "Enhanced Avalanche Effect Analysis Algorithm Considering both Single and Double Key Pair RSA Algorithms," Mar. 20, 2024, *In Review*. doi: 10.21203/rs.3.rs-4113962/v1.

[24] D. Chicco, A. Sichenze, and G. Jurman, "A simple guide to the use of Student's t-test, Mann-Whitney U test, Chi-squared test, and Kruskal-Wallis test in biostatistics," *BioData Mining*, vol. 18, no. 1, p. 56, Aug. 2025, doi: 10.1186/s13040-025-00465-6.

[25] N. G. Zinabu, Y. W. Marye, K. K. Tune, and S. A. Demilew, "Comprehensive Analysis of Lightweight Cryptographic Algorithms for Battery-Limited Internet of Things Devices," *International Journal of Distributed Sensor Networks*, vol. 2025, no. 1, p. 9639728, Jan. 2025, doi: 10.1155/dsn/9639728.

[26] National Institute of Standards and Technology (US), "SHA-3 standard : permutation-based hash and extendable-output functions," National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 202, 2015. doi: 10.6028/NIST.FIPS.202.

[27] H. L. Pham, T. H. Tran, V. T. Duong Le, and N. Yasuhiko, "Flexible and Scalable BLAKE/BLAKE2 Coprocessor for Blockchain-Based IoT Applications," *IEEE Des. Test*, vol. 40, no. 5, pp. 15–25, Oct. 2023, doi: 10.1109/MDAT.2023.3276936.

[28] X. Zou, Y. Peng, T. Li, L. Kong, Z. Pang, and L. Zhang, "Accelerating Blake3 in RISC-V," in *2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT)*, Xiamen, China: IEEE, Aug. 2023, pp. 296–301. doi: 10.1109/CCPQT60491.2023.00057.

[29] N. M. Chacko, V. G. Narendra, M. Balachandra, and S. Rathinam, "Exploring IoT-Blockchain Integration in Agriculture: An Experimental Study," *IEEE Access*, vol. 11, pp. 130439–130450, 2023, doi: 10.1109/ACCESS.2023.3334726.

[30] M. O. Okoye, J. Yang, J. Cui, A. Hussain, van-H. Bui, and D. Espin-Sarzosa, "Optimizing the Transaction Latency in the Blockchain-Integrated Energy-Trading Platform in the Standalone Renewable Distributed Generation Arena," *IEEE Access*, vol. 12, pp. 111970–111981, 2024, doi: 10.1109/ACCESS.2024.3414966.

[31] A. Dolmeta, M. Martina, and G. Masera, "Comparative Study of Keccak SHA-3 Implementations," *Cryptography*, vol. 7, no. 4, p. 60, Nov. 2023, doi: 10.3390/cryptography7040060.

[32] Y. K. Sanjalawe and S. R. Al-E'mari, "Abnormal Transactions Detection in the Ethereum Network Using Semi-Supervised Generative Adversarial Networks," *IEEE Access*, vol. 11, pp. 98516–98531, 2023, doi: 10.1109/ACCESS.2023.3313630.

[33] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, vol. 10, pp. 6605–6621, 2022, doi: 10.1109/ACCESS.2021.3140091.

[34] T. H. Pranto, K. T. A. Md. Hasib, T. Rahman, A. B. Haque, A. K. M. N. Islam, and R. M. Rahman, "Blockchain and Machine Learning for Fraud Detection: A Privacy-Preserving and Adaptive Incentive Based Approach," *IEEE Access*, vol. 10, pp. 87115–87134, 2022, doi: 10.1109/ACCESS.2022.3198956.

[35] J. Jancar *et al.*, "'They're not that hard to mitigate': What Cryptographic Library Developers Think About Timing Attacks," in *2022 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2022, pp. 632–649. doi: 10.1109/SP46214.2022.9833713.

[36] G. Raffin and D. Trystram, "Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 36, no. 1, pp. 96–107, Jan. 2025, doi: 10.1109/TPDS.2024.3492336.

[37] M. Krichen, M. Lahami, and Q. A. Al-Haija, "Formal Methods for the Verification of Smart Contracts: A Review," in *2022 15th International Conference on Security of Information and Networks (SIN)*, Sousse, Tunisia: IEEE, Nov. 2022, pp. 01–08. doi: 10.1109/SIN56466.2022.9970534.