Comparative Analysis of Cryptographic Hash Functions in Blockchain Systems

Oleksandr Kuznetsov[1,2], Oleksandr Peliukh[2], Nikolay Poluyanenko[2], Serhii Bohucharskyi[2], and Ievgeniia Kolovanova[2]

[1] *Department of Political Sciences, Communication and International Relations, University of Macerata, 30/32 Via Crescimbeni, 62100 Macerata, Italy*
[2] *Department of Information and Communication Systems Security, School of Computer Sciences, V. N. Karazin Kharkiv National University, 4 Svobody sq., 61022 Kharkiv, Ukraine*

Abstract
In the burgeoning realm of digital technologies, blockchain has emerged as a revolutionary force, underpinned by the intricate fabric of cryptographic hash functions. This study provides a comprehensive evaluation of various hash functions, delineating their processing efficacies across a spectrum of input block sizes. Through rigorous empirical analysis, our research juxtaposes the performance metrics of ten distinct algorithms, thereby offering invaluable insights into their respective computational robustness. Notably, our findings underscore the complexities inherent in the selection of an appropriate hash function for blockchain applications. Beyond mere processing speeds, the selection process requires a nuanced understanding of cryptographic resilience, adaptability to emerging threats, and seamless integration with prevailing infrastructures. While our results furnish a contemporaneous benchmark, they also accentuate the imperative for incessant research and adaptation in an ever-evolving digital landscape. This investigation serves both as a reference point for current blockchain applications and a clarion call for sustained innovation in the quest for optimized cryptographic solutions. The overarching aim is to fortify the blockchain's promise by ensuring its security and performance through the judicious application of cryptographic hash functions, thereby catalyzing a more decentralized, efficient, and secure digital future.

Keywords
Blockchain, cryptographic hash functions, processing efficiency, digital evolution, decentralized systems.

1. Introduction

In the contemporary era of digital communication and data exchange, cryptographic hash functions play an indispensable role in ensuring data integrity, authentication, and security [1]. A cryptographic hash function is a mathematical algorithm that takes an input (or "message") and produces a fixed-size string of characters, which is typically a sequence of numbers and letters [2–3]. These functions are designed such that even a minute change in the input will produce a drastically different output, making them fundamental tools in the fields of cryptography, cyber-security, and, notably, blockchain technology [4].

Blockchain, a decentralized and distributed ledger system, relies heavily on cryptographic hash functions for its operation [5]. The core principle behind blockchain's security and integrity is the continuous chaining of blocks using cryptographic hashes [6–7]. Given the growing significance of blockchain in various applications, ranging from financial

CPITS-2023-II: Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2023, Kyiv, Ukraine
EMAIL: kuznetsov@karazin.ua (O. Kuznetsov); oleksandrpelyukh@gmail.com (O. Peliukh); nlfsr01@gmail.com (N. Poluyanenko); sbogucharskiy@karazin.ua (S. Bohucharskyi); e.kolovanova@karazin.ua (I. Kolovanova)
ORCID: 0000-0003-2331-6326 (O. Kuznetsov); 0000-0003-0507-0262 (O. Peliukh); 0000-0001-9386-2547 (N. Poluyanenko); 0000-0003-4971-4314 (S. Bohucharskyi); 0000-0002-0326-2394 (I. Kolovanova)

transactions to supply chain management, the efficiency, security, and performance of the hashing algorithms employed are of paramount importance [8].

Among the myriad of hash functions, SHA2 [9–10], SHABAL [11–12], SHAVITE [12–13], Keccak [14–15], and Blake [12, 16] have emerged as frontrunners in contemporary blockchain projects. Each of these algorithms offers unique characteristics in terms of security, computational requirements, and resistance against various attack vectors [17–18].

This paper embarks on a comprehensive comparative analysis of these widely adopted cryptographic hash algorithms. Through rigorous benchmarking and assessment, we aim to elucidate their respective strengths, vulnerabilities, and performance metrics. This exploration serves not only to guide current blockchain implementations but also to inform future innovations in the realm of cryptographic research and development.

2. Methodology

The comparative evaluation of cryptographic hash algorithms necessitates a robust and reproducible methodology. A structured approach ensures the reliability of results, enabling meaningful comparisons that can guide both academic research and practical implementations. In this study, we have adopted the following methods to systematically evaluate the aforementioned hash functions.

1. Hardware and Software Configuration: The testing environment was standardized using an Intel Core i9-7980 with a clock speed of 2.60 GHz. Such a high-performance platform ensures consistency in testing, eliminating potential bottlenecks that might bias results. The operating system deployed was Windows 10, which provides a stable environment with widespread adoption in both academic and commercial sectors.

2. Data Sets: To understand how these hash functions perform across varying data lengths, a range of data sizes was chosen. Specifically, we used 21 distinct data lengths starting from 2020 bytes (or 1 byte) to 220220 bytes. The exponential progression aids in capturing the potential non-linear scaling characteristics of these algorithms, providing insights into their behavior under both minimal and substantial loads.

3. Performance Metrics: The core metrics used for gauging the performance of each hash algorithm are as follows:

- Cycles/byte: This metric evaluates the number of processor cycles required to process each byte of data. A lower value indicates a more efficient algorithm in terms of computational cycles.
- MB/sec: Measuring the Megabytes processed per second, this metric offers a direct view into the throughput of the algorithm. A higher MB/sec value suggests a faster hashing capability, ideal for real-time or large-scale applications.
- KHash/sec: Representing the thousands of hashes computed per second, KHash/sec offers insights into the sheer speed of the algorithm, especially valuable for applications like mining where the hash rate is a critical parameter.

Each hashing algorithm was run multiple times for each data length to ensure statistical robustness. The results were then averaged to minimize the potential influence of outliers or sporadic system interruptions.

In summary, our methodology provides a comprehensive and systematic approach to evaluate the speed and efficiency of leading cryptographic hash algorithms, paving the way for informed decisions in blockchain implementations and other applications where hash functions play a crucial role.

3. Hash Algorithms in Blockchain Systems

The bedrock of blockchain technology, a decentralized ledger system, is cryptographic hashing. These algorithms ensure that data remains intact and unchanged, providing a layer of security and authenticity to transactions. In this section, we explore the specific hash algorithms that have found prominence in the blockchain sphere, illuminating their distinctive features and their adoption rationale in various blockchain projects.

1. **SHA-256** [9–10]:

- Description: A member of the SHA-2 (Secure Hash Algorithm 2) family, SHA-256 produces a 256-bit (32-byte) hash. It's known for its security and computational efficiency.
- Blockchain Adoption: Perhaps the most famous application of SHA-256 is in Bitcoin, the pioneering cryptocurrency [19]. The developers of Bitcoin opted for SHA-256 due to its strong collision resistance and widespread recognition as a secure algorithm.

2. **SHA-512** [9–10]**:**
   - Description: Also from the SHA-2 family, this produces a 512-bit hash, offering even more robust security, albeit at a higher computational cost.
   - Blockchain Adoption: Some blockchain platforms choose SHA-512 for added security layers in specific components where computational efficiency is not the primary concern.
3. **SHABAL-256 & SHABAL-512** [11–12]**:**
   - Description: These are cryptographic hash functions with output lengths of 256 and 512 bits respectively.
   - Blockchain Adoption: While not as ubiquitous as SHA variants, they find use in some niche blockchain applications due to their unique cryptographic properties.
4. **SHAVITE-256 & SHAVITE-512** [12–13]**:**
   - Description: SHAVITE is a family of hash functions designed to be efficient on a variety of platforms, especially hardware implementations.
   - Blockchain Adoption: Some blockchain projects that emphasize hardware-based transactions or have specific hardware considerations might lean towards SHAVITE for its efficiency in such contexts.
5. **Keccak-256 & Keccak-512** [14–15]**:**
   - Description: Keccak is the basis for the SHA-3 standard. It's recognized for its sponge construction which offers a high degree of flexibility and security.
   - Blockchain Adoption: Ethereum, a leading smart contract platform, uses Keccak-256. The choice stemmed from Keccak's distinction as a winner of the NIST hash function competition and its perceived security advantages over SHA-2.
6. **Blake-256 & Blake-512** [12–16]**:**
   - Description: BLAKE is notable for its speed and high-security levels. It's simpler in design than SHA-2, leading to potential advantages in performance.
   - Blockchain Adoption: Siacoin, a decentralized storage platform, employs Blake-256. The developers favored its blend of speed and security, particularly important for a system that frequently processes large amounts of data.

In summation, the choice of a hashing algorithm in a blockchain project is multifaceted. Factors like security, computational efficiency, specific platform considerations, and even historical trust play a role. As blockchain technology continues to evolve, so will its reliance on these cryptographic backbones, making their study and understanding even more crucial.

4. Results

4.1. Performance Analysis of SHA2-256 and SHA2-512

In our pursuit to comprehend the computational efficiency and performance of popular cryptographic hash functions, we subjected SHA2-256 and SHA2-512 to rigorous benchmarking. The tests were conducted on a 64-bit platform powered by Intel Core i9-7980 running at 2.60 GHz. The following sections offer a detailed analysis and a summary of Table 1 that amalgamates the results for both hash algorithms.

Analysis and Commentary:

- Observing the Cycles/byte for both algorithms, SHA2-256 begins with a considerably higher number of cycles for smaller input blocks. However, as the input size increases, the efficiency in terms of cycles/bytes tends to converge for both algorithms. Notably, SHA2-512 maintains a consistent trend and is marginally less efficient for larger input sizes compared to SHA2-256.
- In terms of MB/sec, a similar pattern emerges where SHA2-512 has a slightly lower throughput compared to SHA2-256, especially evident in larger input

blocks. The throughput for both algorithms dramatically increases as the input block size grows, with SHA2-512 reaching a peak of around 369.87 MB/sec at 2^{20} bytes.

- The KHash/sec metric is particularly revealing. For smaller data blocks, SHA2-256 seems to achieve higher hashing speeds. However, as the input block size increases, both algorithms experience a stark reduction in their KHash/sec. Notably, the drop in performance for SHA2-512 is more precipitous, especially after 2^6 bytes, reflecting a decrease of almost half.
- When juxtaposed, SHA2-256 generally offers better computational efficiency and higher throughput for most input sizes. However, the choice between the two would also depend on security requirements, as SHA2-512, with its larger hash size, could offer stronger cryptographic security.
- In conclusion, the performance metrics highlight the inherent trade-offs between computational efficiency and cryptographic strength. While SHA2-256 demonstrates superior performance in most scenarios, the choice to use SHA2-512 might be motivated by heightened security concerns. As with any cryptographic decision, understanding the context and requirements of the application remains paramount.

Table 1
Performance Metrics for SHA2-256 and SHA2-512

Test#	Input Block Size, bytes	SHA2-256			SHA2-512		
		Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
1	1	848.68	3.06	3060.37	945.51	2.74	2741.59
2	2	423.30	6.13	3065.65	472.36	5.49	2744.53
3	4	211.72	12.24	3059.57	236.28	10.98	2744.68
4	8	105.80	24.53	3066.73	118.07	21.94	2742.09
5	16	52.84	49.07	3066.73	59.05	43.93	2745.54
6	32	26.28	97.98	3061.86	29.48	87.89	2746.69
7	64	25.74	100.50	1570.25	14.79	175.35	2739.80
8	128	19.14	134.40	1049.99	14.36	180.85	1412.90
9	256	15.91	162.82	636.02	10.70	243.01	949.25
10	512	14.28	181.29	354,08	8,84	292,57	571,43
11	1024	13,46	192,54	188,03	7.94	327.37	319.70
12	2048	13.08	198.22	96.79	7.52	346.98	169.42
13	4096	12.88	201.49	49.19	7.24	357.63	87.31
14	8192	12.78	203.41	24.83	7.13	363.71	44.40
15	16384	12.71	203.25	12.41	7.13	366.63	22.38
16	32768	12.68	204.48	6.24	7.04	368.70	11.25
17	65536	12.67	204.88	3.13	7.03	368.70	5.63
18	131072	12.67	204.72	1.56	7.01	368.96	2.81
19	262144	12.67	205.00	0.78	7.02	369.09	1.41
20	524288	12.65	205.16	0.39	7.01	368.83	0.70
21	1048576	12.68	204.88	0.20	7.02	369.87	0.35

4.2. Performance Analysis of SHABAL-256 and SHABAL-512

Table 2 below provides a summarization of the performance metrics associated with the SHABAL-256 and SHABAL-512 cryptographic hash algorithms, specifically, when processed on a 64-bit computational platform using an Intel Core i9-7980 at 2.60 GHz.

Analysis and Commentary:

- Both SHABAL-256 and SHABAL-512 show a clear pattern where the cycles per byte decrease (indicative of improved efficiency) as the size of the input block increases. This suggests that both algorithms are designed to process larger data blocks more efficiently than smaller ones. However, SHABAL-256, generally requires slightly fewer cycles per byte compared to SHABAL-512, emphasizing its higher efficiency in processing data.

- As the input block size increases, the megabytes processed per second (MB/sec) also increase, reaching an asymptotic limit. Both algorithms have comparable throughputs, but SHABAL-256 tends to outperform SHABAL-512 for larger block sizes.
- The kilohashes per second (KHash/sec) reflect the speed at which the system can compute the hash values. The rate decreases with the size of the input block. The decrease is steeper for SHABAL-512, indicating it may require more computational resources to hash larger blocks compared to its SHABAL-256 counterpart.
- The data suggests that for tasks requiring a balance between security and efficiency, SHABAL-256 might be preferable due to its slightly better efficiency metrics. However, for tasks where the higher cryptographic strength of a 512-bit hash is necessary, the SHABAL-512, despite its slight performance penalty, would be the algorithm of choice.

In conclusion, the choice between SHABAL-256 and SHABAL-512 would be contingent on the specific application requirements and the trade-off between security and performance one is willing to accept. This detailed performance analysis offers practitioners insights to make informed decisions on the appropriate cryptographic algorithm selection for their needs.

Table 2
Performance Metrics for SHABAL-256 & SHABAL-512

Test#	Input Block Size, bytes	SHABAL-256			SHABAL-512		
		Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
1	1	1235.41	2.10	2097.95	1274.73	2.04	2039.99
2	2	617.18	4.20	2099.00	636.36	4.06	2028.43
3	4	308.82	8.39	2097.32	324.19	7.77	1941.81
4	8	154.24	16.79	2099.17	165.16	16.13	2015.87
5	16	77.19	33.53	2095.81	79.38	32.09	2005.39
6	32	38.50	67.36	2105.10	40.14	64.69	2021.47
7	64	25.00	103.79	1621.70	26.05	100.48	1569.95
8	128	14.95	173.55	1355.84	15.49	168.04	1312.82
9	256	9.91	261.69	1022.21	10.19	252.91	987.94
10	512	7.40	349.99	683.58	7.69	337.27	658.73
11	1024	6.15	420.61	410.75	6.31	416.60	406.83
12	2048	5.52	468.53	228.78	5.55	466.86	227.96
13	4096	5.21	497.43	121.44	5.23	495.08	120.87
14	8192	5.05	513.00	62.62	5.15	503.64	61.48
15	16384	4.97	514.01	31.37	5.07	511.25	31.20
16	32768	4.95	524.29	16.00	5.04	515.27	15.72
17	65536	4.94	523.24	7.98	5.00	518.84	7.92
18	131072	4.94	524.29	4.00	5.03	515.27	3.93
19	262144	4.94	525.34	2.00	5.01	523.76	2.00
20	524288	4.93	525.60	1.00	4.95	525.34	1.00
21	1048576	4.94	524.55	0.50	5.01	507.78	0.48

4.3. Performance Analysis of SHAVITE-256 and SHAVITE-512

The following section delineates a comprehensive analysis of the performance metrics associated with the SHAVITE-256 and SHAVITE-512 cryptographic hash algorithms. Both were tested on a 64-bit computational platform, specifically an Intel Core i9-7980 clocked at 2.60 GHz. Table 3 provides a consolidated overview of these findings.

Analysis and Commentary:

- A striking observation is that SHAVITE-256 is significantly more efficient in terms of cycles per byte compared to SHAVITE-512, particularly for smaller input block sizes. This suggests that SHAVITE-256 can process data more swiftly, thereby rendering it more suitable for applications that prioritize speed.
- When it comes to MB/sec, SHAVITE-256 consistently outperforms SHAVITE-512 across all input block sizes. The data delineates a trend where MB/sec increases for SHAVITE-256 as block size augments, indicating efficient data processing for larger blocks. On the other hand, SHAVITE-512 starts with a low throughput, which gradually increases with block size but never surpasses SHAVITE-256.
- The kilohashes per second (KHash/sec) provide insights into the hashing capabilities of the system. While both algorithms show a declining trend with increasing block size, SHAVITE-256 showcases a steep decline after a certain threshold, suggesting some limitations in handling larger blocks efficiently.
- Overall, SHAVITE-256 appears to be more performance-centric, delivering faster results with less computational overhead. In contrast, SHAVITE-512, albeit less efficient, might offer a more robust cryptographic strength due to its larger hash size.

In summation, the distinction between opting for SHAVITE-256 or SHAVITE-512 depends heavily on the specific application's requirements, emphasizing the trade-off between computational efficiency and cryptographic resilience. This empirical analysis provides a solid foundation for stakeholders to make an informed decision regarding the selection of an appropriate cryptographic algorithm tailored to their specific needs.

Table 3

Performance Metrics for SHAVITE-256 and SHAVITE-512

Test#	Input Block Size, bytes	SHAVITE-256			SHAVITE-512		
		Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
1	1	883.77	2.92	2915.87	2788.42	0.93	925.64
2	2	441.15	5.85	2924.41	1393.34	1.85	927.29
3	4	220.52	11.72	2928.98	687.22	3.77	942.29
4	8	110.18	23.41	2926.37	346.47	7.51	939.25
5	16	55.13	46.81	2925.71	176.91	14.73	920.45
6	32	27.43	93.88	2933.83	87.25	29.79	930.91
7	64	27.01	95.18	1487.16	43.16	59.53	930.12
8	128	20.16	126.20	985.92	42.48	59.93	468.17
9	256	16.79	152.25	594.74	32.49	80.25	313.46
10	512	15.11	168.96	330.00	27.31	96.47	188.43
11	1024	14.33	178.63	174.45	23.77	106.31	103.82
12	2048	13.92	184.15	89.92	22.78	113.36	55.35
13	4096	13.77	187.11	45.68	22.13	113.53	27.72
14	8192	13.67	188.66	23.03	22.05	119.41	14.58
15	16384	13.61	189.24	11.55	22.55	117.75	7.19
16	32768	13.58	189.79	5.79	21.18	121.87	3.72
17	65536	13.55	189.86	2.90	21.12	122.90	1.88
18	131072	13.53	189.99	1.45	21.07	121.55	0.93
19	262144	13.52	190.03	0.72	21.09	122.97	0.47
20	524288	13.52	190.10	0.36	21.12	123.19	0.23
21	1048576	13.53	189.93	0.18	21.04	123.14	0.12

4.4. Performance Analysis of KECCAK-256 and KECCAK-512

The subsequent section presents an analysis of the performance metrics associated with the KECCAK-256 and KECCAK-512 cryptographic hash algorithms. These were rigorously tested on a 64-bit computational platform utilizing an Intel Core i9-7980, clocked at 2.60 GHz. A synthesized overview is captured in Table 4.

Analysis and Commentary:

- Comparatively, KECCAK-256 and KECCAK-512 present close competition in terms of cycles per byte, especially when handling smaller input block sizes. This close margin hints at their underlying similarities in algorithmic efficiency.
- Observing the MB/sec, KECCAK-512 marginally outperforms KECCAK-256, albeit the differences are subtle. As block size increases, both algorithms exhibit an exponential growth in MB/sec, underscoring the efficiency of both KECCAK variants for processing larger blocks.
- The KHash/sec metric reveals interesting trends. Both algorithms start strong but undergo a drastic reduction in hashing speed when handling larger blocks. Notably, KECCAK-512 seems to suffer a sharper drop in KHash/sec than KECCAK-256 for the same data sizes, post the input block size of 2^{27} bytes.
- At a broad level, while KECCAK-256 seems slightly less efficient in data processing (MB/sec) compared to KECCAK-512, its steadier hashing speed (KHash/sec) might render it preferable in scenarios where consistent hash rate performance is crucial.

In essence, the choice between KECCAK-256 and KECCAK-512 hinges upon specific application demands, weighing the nuances between data processing speed and hashing consistency. This empirical scrutiny offers stakeholders a substantial base for selecting an optimal cryptographic hashing solution, that resonates with their nuanced requirements.

Table 4

Performance Metrics for KECCAK-256 and KECCAK-512

Test#	Input Block Size, bytes	KECCAK-256			KECCAK-512		
		Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
1	1	1307.78	1.97	1974.12	1269.54	2.04	2041.02
2	2	657.99	3.94	1969.97	633.65	4.09	2045.36
3	4	328.59	7.88	1970.41	316.76	8.18	2044.80
4	8	159.08	16.29	2036.23	156.76	16.44	2055.39
5	16	79.54	32.54	2034.02	77.77	33.31	2081.83
6	32	39.75	65.20	2037.56	38.83	66.58	2080.51
7	64	19.88	130.53	2039.59	19.42	133.64	2088.20
8	128	9.92	261.16	2040.35	19.11	134.52	1050.93
9	256	9.84	263.33	1028.63	18.93	137.16	535.78
10	512	9.69	267.22	521.92	18.60	139.48	272.41
11	1024	9.64	268.87	262.56	17.41	149.28	145.79
12	2048	9.57	271.09	132.37	16.71	155.00	75.68
13	4096	9.24	280.82	68.56	16.42	158.04	38.58
14	8192	9.06	285.64	34.87	16.38	158.11	19.30
15	16384	8.99	288.07	17.58	16.37	158.20	9.66
16	32768	8.95	289.34	8.83	16.49	157.30	4.80
17	65536	8.95	289.42	4.42	16.40	157.49	2.40
18	131072	8.94	289.90	2.21	16.47	158.08	1.21
19	262144	8.96	289.58	1.10	16.38	158.18	0.60
20	524288	8.95	289.50	0.55	16.39	158.28	0.30
21	1048576	8.96	289.50	0.28	16.40	158.11	0.15

4.5. Performance Analysis of BLAKE-256 and BLAKE-512

This section elucidates the performance metrics associated with the BLAKE-256 and BLAKE-512 cryptographic hash algorithms. Testing was conducted on a 64-bit computational platform powered by an Intel Core i9-7980 with a clock speed of 2.60 GHz. For clarity and brevity, a consolidated representation of the results is furnished in Table 5.

Analysis and Commentary:

- When contrasting BLAKE-256 and BLAKE-512, the former showcases fewer cycles per byte across most block sizes, indicating enhanced efficiency. This suggests that, on a byte-to-byte basis, BLAKE-256 can process data more expediently than its BLAKE-512 counterpart.
- MB/sec metrics reveal that BLAKE-256 consistently surpasses BLAKE-512 in data processing rates. Especially for larger block sizes, BLAKE-256's ability to sustain high throughput is particularly noteworthy.
- The KHash/sec parameter for both algorithms depicts an exponential decline as block size increases. However, BLAKE-256 consistently delivers higher hashing rates than BLAKE-512 until a certain data size threshold, after which the hashing speeds of both variants start to converge.
- BLAKE-256 seems to outperform BLAKE-512 in terms of both data processing and hashing speed for the majority of block sizes. This renders BLAKE-256 a more favorable choice for applications where speed and efficiency are pivotal.

In conclusion, while both BLAKE variants exhibit commendable performance, BLAKE-256 appears to have a slight edge in processing and hashing capabilities over BLAKE-512 on the tested platform. Decision-makers and stakeholders are advised to evaluate these empirical findings within the context of their specific application needs, optimizing for desired performance outcomes.

Table 5

Performance Metrics for BLAKE-256 and BLAKE-512

Test#	Input Block Size, bytes	BLAKE-256			BLAKE-512		
		Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
1	1	648.36	3.99	3991.69	846.69	3.06	3059.12
2	2	323.95	7.99	3993.05	423.03	6.12	3060.28
3	4	160.85	16.06	4015.69	211.61	12.24	3060.28
4	8	80.29	32.16	4020.61	104.68	24.74	3092.05
5	16	40.18	64.33	4020.61	52.26	49.58	3098.63
6	32	19.96	129.74	4054.44	26.06	99.20	3100.09
7	64	19.39	133.63	2087.93	13.06	198.41	3100.09
8	128	14.28	181.63	1419.02	12.52	206.78	1615.46
9	256	11.73	221.41	864.86	9.21	281.27	1098.71
10	512	10.45	248.71	485.77	7.63	343.35	670.60
11	1024	9.80	264.93	258.72	6.74	385.36	376.33
12	2048	9.46	273.85	133.72	6.33	410.88	200.63
13	4096	9.29	279.03	68.12	6.11	424.70	103.69
14	8192	9.20	281.50	34.36	5.99	431.51	52.67
15	16384	9.16	282.71	17.26	5.94	436.18	26.62
16	32768	9.17	282.71	8.63	5.93	436.91	13.33
17	65536	9.21	281.42	4.29	5.93	438.37	6.69
18	131072	9.21	281.65	2.15	5.91	438.92	3.35
19	262144	9.32	281.88	1.08	5.91	438.55	1.67
20	524288	9.21	281.65	0.54	5.91	439.29	0.84
21	1048576	9.22	281.12	0.27	5.92	438.00	0.42

5. Comparison of results

5.1. Comparison of Cycles per Byte

The Fig. 1 visualizes the cycles per byte for ten different hashing algorithms, as a function of input block size.

Upon examining the data presented in the Fig. 1, several observations and conclusions can be made:

- The Cycles/byte varies widely among the algorithms, demonstrating the diverse efficiency levels of these algorithms. For instance, when considering the smallest input block size of one byte, BLAKE-256 shows the highest efficiency with the least cycles, whereas SHAVITE-512 consumes the most.
- As the input block size increases, the difference in efficiency between algorithms tends to diminish. This implies that the overheads affecting the efficiency of algorithms are relatively more prominent for smaller block sizes.
- Certain algorithms, like BLAKE-256 and BLAKE-512, maintain a relatively consistent performance regardless of the input block size. On the other hand, algorithms like SHAVITE-512 show drastic improvements as the input block size increases.
- For most algorithms, efficiency in terms of Cycles/byte seems to stabilize or reach an asymptotic value beyond an input block size of 8KB (8192 bytes). This suggests that using block sizes above this might not provide significant gains in terms of computational efficiency.
- For algorithms that have both 256-bit and 512-bit versions, it is evident that the 256-bit version is generally more efficient in terms of Cycles/byte. This is expected due to the reduced computational complexity associated with smaller bit sizes.
- Among all, BLAKE-256 consistently stands out as one of the most efficient algorithms across all input block sizes. Conversely, SHAVITE-512, while initially being the least efficient for smaller block sizes, sees significant improvement as block size increases.
- While cycles per byte is an important metric for computational efficiency, it’s essential to consider other factors like security, implementation complexity, and compatibility when selecting a hashing algorithm for real-world applications.

In conclusion, the choice of a hashing algorithm should not solely rely on cycles per byte but should be a holistic decision based on various performance metrics and specific application requirements. However, from a pure efficiency perspective, BLAKE-256 exhibits excellent performance across a wide range of input block sizes.

This analysis provides valuable insights for developers and organizations looking to optimize their cryptographic operations, ensuring both security and performance are maintained at optimal levels.

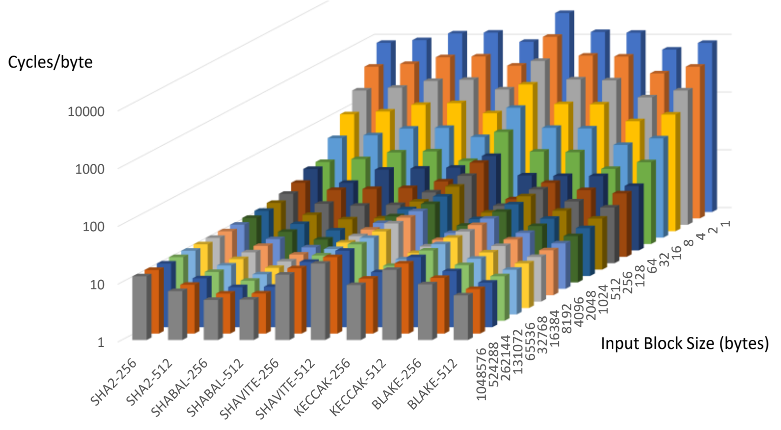

Figure 1: Comparison of Cycles per Byte across Multiple Hashing Algorithms

5.2. Comparison of Hashing Algorithms by Throughput

Fig. 2 visualizes the performance of 10 cryptographic hashing algorithms across varying input block sizes, measured in MB/sec.

The results indicate a non-uniform performance landscape across the studied algorithms and input block sizes. Several key observations can be deduced:

- Almost all algorithms show increased throughput as the input block size increases. This trend likely represents the amortization of initialization and finalization costs over larger data blocks. However, there's a plateauing effect observed in most algorithms, indicating an approach towards their maximum throughput capacity.
- By the largest input block size (1048576 bytes), the BLAKE-512 algorithm emerged as the leading performer with 438 MB/sec, closely followed by SHABAL-256 and SHABAL-512. This implies that in scenarios requiring high-throughput hashing of larger data chunks, these algorithms would be preferable.
- KECCAK-256 exhibits a remarkably stable growth in throughput as the block size increases, without any significant dips or spikes. This consistent performance might make it a preferable choice in environments where input sizes can vary significantly, ensuring predictable hashing rates.
- For the smallest input block size (1 byte), several algorithms show significantly reduced throughput, most notably SHAVITE-512. This is an important consideration for applications dealing with a large number of small-sized data blocks.
- Interestingly, for many algorithms, their 512-bit variants (like SHA2-512 and SHAVITE-512) don't necessarily double the throughput compared to their 256-bit counterparts. This suggests that the performance does not linearly scale with the bit length, and other internal algorithmic factors play a significant role.
- Despite its performance with smaller block sizes, SHABAL-512 showed a significant jump in throughput once the block size exceeded 64 bytes, overtaking many competitors. This behavior underscores the importance of benchmarking cryptographic algorithms over a diverse range of input sizes.

In conclusion, while raw throughput is an essential metric, selecting a hashing algorithm for practical applications should consider other factors like security guarantees, implementation complexity, and specific use-case requirements. The diverse performance landscape observed underscores the importance of nuanced, application-specific benchmarking before settling on a cryptographic primitive.

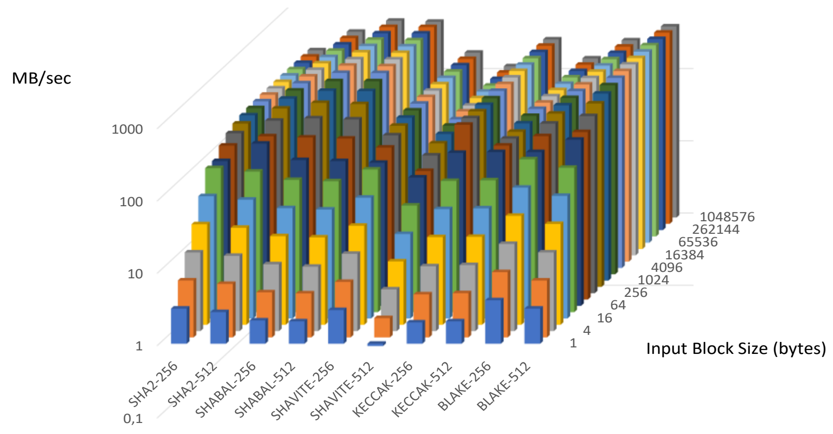

Figure 2: Comparison of Hashing Algorithms by Throughput (MB/sec)

5.3. Performance Comparison by KHash/sec Criterion

In this rigorous performance assessment, we aim to dissect and analyze the throughput of various cryptographic hash functions, delineating their performance in kilohashes per second (KHash/sec). This metric offers a comprehensive understanding of how swiftly each algorithm can process input data, rendering it a quintessential element in evaluating and benchmarking cryptographic protocols (Fig. 3).

Several key observations can be deduced:

- From a cursory glance at the dataset, it's palpable that the KHash/sec for each algorithm plummets as the input block size burgeons. This is a quintessential behavior as a larger input block demands greater computational power and time, thereby reducing the KHash/sec.
- The SHA2-256 and SHA2-512 demonstrate a fascinating interplay. Initially, SHA2-256 marginally outperforms its 512 counterpart for smaller block sizes. However, as we approach larger block sizes, SHA2-512 emerges as the preeminent variant, even doubling the throughput of SHA2-256 in some tests.
- The performance degradation is particularly pronounced for SHABAL-256 and SHABAL-512. For minuscule block sizes, their throughput is analogous, but as the block size escalates, SHABAL-512's performance slightly trumps SHABAL-256, showcasing its optimized efficiency for handling larger data chunks.
- SHAVITE-256 consistently supersedes SHAVITE-512 across all block sizes. This could be attributed to the inherent architectural decisions made during its design, emphasizing speed for 256-bit processing.
- KECCAK-256 and KECCAK-512 both experience an intriguing surge around 64 bytes, but KECCAK-256's advantage wanes post this point, with its performance plummeting precipitously. In contrast, KECCAK-512 maintains a more consistent (albeit declining) trajectory, highlighting its robustness against variable input sizes.
- Of particular note is BLAKE-256 and BLAKE-512's relatively stable performance, with the latter consistently outshining the former as block sizes enlarge. This consistency might be emblematic of BLAKE's resilience and adaptability across diverse data sizes.
- For smaller block sizes, BLAKE-256 reigns supreme, but as we transition to heftier blocks, the crown oscillates between BLAKE-512, SHABAL-512, and SHA2-512.

Cryptographic hash function performance is contingent upon a myriad of factors, including design principles, input block size, and implementation nuances. While this empirical analysis provides a granular breakdown of their respective efficacies, practitioners must weigh these results against their specific use-case scenarios and constraints. In realms where speed is of the essence, selecting an algorithm with superior KHash/sec is paramount. Conversely, in more security-sensitive domains, one might prioritize cryptographic strength and resistance to attacks over sheer speed.

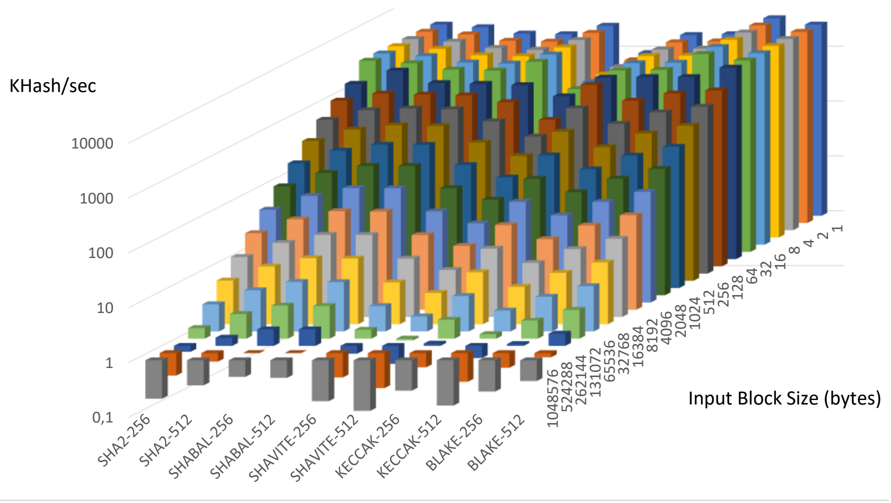

Figure 3: Comparative performance of cryptographic hash functions across different input block sizes in terms of KHash/sec

6. Discussion

In the rapidly evolving landscape of digital technologies, the performance of cryptographic hash functions takes on heightened importance. The results outlined in the previous section present a comprehensive understanding of how varying cryptographic algorithms fare in terms of processing speeds across diverse input block sizes. To contextualize these findings, it's essential to dive into their implications, especially within the ambit of blockchain systems.

1. The Imperative of Hash Functions in Blockchain. At the core of blockchain technology lies the unassailable need for data integrity, authentication, and security. Cryptographic hash functions are the lynchpin that holds this edifice together. Every transaction or data entry within a block is hashed, and this hash is woven into the very fabric of the blockchain, ensuring both its veracity and its inviolability. The choice of a hash function, therefore, isn't a trivial one; it can profoundly shape the efficacy, scalability, and security of a blockchain system.
2. Throughput versus Security. While our study primarily focused on performance in terms of KHash/sec, it is crucial to remember that in the blockchain realm, speed isn't the sole desideratum. A balance must be struck between processing speed and cryptographic robustness. For instance, while BLAKE-256 showcased stellar performance for smaller block sizes, it's imperative to assess its resilience against potential cryptographic attacks, especially in a blockchain setting where a compromised hash function can have cataclysmic consequences.
3. Scalability Concerns. As blockchain systems burgeon in size and complexity, they grapple with scalability issues. Here, the hash function's performance for larger block sizes becomes especially pertinent. Functions like BLAKE-512, which demonstrate consistent throughput across escalating block sizes, might be particularly suited for burgeoning blockchain networks, ensuring that transaction processing remains brisk even as the system grows.
4. Adaptability and Future-Proofing. The landscape of digital threats is in constant flux. As cryptographic methodologies evolve, so do the techniques employed by malicious actors. Hence, it's pivotal for blockchain systems not just to adopt algorithms that are robust today but ones that can stand the test of time. KECCAK's consistent trajectory, for instance, might hint at its future adaptability.
5. Beyond Performance: Other Considerations. While our focus has been squarely on KHash/sec, blockchain architects must also consider other facets like power consumption, ease of implementation, and compatibility with existing systems. A hash function might be blazingly fast, but if it demands exorbitant computational power, it might not be tenable for energy-conscious implementations.

In summation, while the performance metrics provided in our study furnish invaluable insights into the prowess of various cryptographic hash functions, they are but one piece of the puzzle. For blockchain systems, the choice of a hash function is a nuanced decision, interweaving concerns of speed, security, scalability, and sustainability [20–22]. As blockchain continues to reshape industries and redefine paradigms, ensuring its bedrock—the cryptographic hash function—is optimally chosen becomes not just desirable but indispensable

7. Conclusion

The relentless advance of technology has positioned blockchain as a frontrunner in the reshaping of our digital landscape. At the heart of this transformative technology lie cryptographic hash functions, serving as sentinels guarding the sanctity and security of data transactions. Our comprehensive study has thrown into sharp relief the comparative merits and demerits of various hash functions, particularly in the context of their processing speeds across diverse input block sizes.

While our empirical data provides a significant point of reference for determining the efficiency of these algorithms, it also underscores a broader, more intricate narrative. This narrative pivots on the need for a delicate balance between performance and security, scalability and adaptability, innovation and sustainability. The nuances of choosing the right hash function for a specific application—especially one as pivotal as blockchain—extend well beyond mere speed metrics. They touch upon foundational concerns like cryptographic resilience, adaptability to emerging threats, and harmonious integration with existing infrastructures.

It is also paramount to acknowledge that the digital milieu is in constant flux, with both challenges and innovations emerging at an unprecedented pace. As such, while our findings provide a robust benchmark for the current scenario, they also serve as a testament to the need for continuous research and evolution in this domain.

In the annals of technological evolution, blockchain stands out as a harbinger of both promise and complexity. Ensuring its unassailable security and optimal performance will invariably hinge on the judicious choice and implementation of cryptographic hash functions. The hope is that our research, with its analytical rigor and insights, will inform and inspire future endeavors in this critical juncture of blockchain technology, propelling it toward its fullest potential and shaping a more secure, efficient, and decentralized digital future.

8. Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101007820—TRUST. This publication reflects only the author's view and the REA is not responsible for any use that may be made of the information it contains.

References

[1] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press (2018). doi: 10.1201/9780429466 335.

[2] S. Rubinstein-Salzedo, Cryptography, Springer International Publishing (2018). doi: 10.1007/978-3-319-94818-8.

[3] R. Klima, N. Sigmon, Cryptology : Classical and Modern, Chapman and Hall/CRC (2018). doi: 10.1201/9781315 170664.

[4] 11 The Hash Function, Cryptography and Network Security, River Publishers, (2022) 197–204.

[5] A. Bruen, M. Forcinito, J. McQuillan, Blockchain and Bitcoin, Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century 26 (2021) 549–560. doi: 10.1002/9781119582397.ch26.

[6] A. Bessalov, et al., Modeling CSIKE Algorithm on Non-Cyclic Edwards Curves, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3288 (2022) 1–10.

[7] A. Bessalov, et al., Multifunctional CRS Encryption Scheme on Isogenies of Non-Supersingular Edwards Curves, in: Workshop on Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 (2023) 12–25.

[8] A. Tiwari, Chapter 14—Cryptography in Blockchain, Distributed Computing to Blockchain, (2023) 251–265. doi: 10.1016/B978-0-323-96146-2.00011-5.

[9] Secure Hash Standard (SHS), Natl. Ins. Stand. Technol. U.S. Department of Commerce (2015). doi: 10.6028/ NIST.FIPS.180-4.

[10] T. Hansen, D. Eastlake 3rd, US Secure Hash Algorithms (SHA and HMAC-SHA), RFC (2006). doi: 10.17487/RFC4634.

[11] Shabal, (2009). URL: https://web.archive.org/web/2009120 9170807/http://www.shabal.com/

[12] I.T.L. Computer Security Division, SHA-3 Project (2017). URL: https://csrc.nist.gov /projects/hash-functions/sha-3-project

[13] O. Dunkelman, E. Biham, The SHAvite-3—A New Hash Function, Symmetric Cryptography (2009) 1–39. doi: 10.4230/DagSemProc.09031.18.

[14] NIST, NIST Releases SHA-3 Cryptographic Hash Standard (2015). URL: https://www.nist.gov/news-events/news/2015/08/nist-releases-sha -3-cryptographic-hash-standard

[15] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Natl. Ins. Stand. Technol. U.S. Department of Commerce (2015). doi:10.6028/NIST.FIPS.202.

[16] BLAKE2. URL: https://www.blake2.net/

[17] V. Sokolov, P. Skladannyi, H. Hulak, Stability Verification of Self-Organized Wireless Networks with Block Encryption, in: 5th International Workshop on Computer Modeling and Intelligent Systems, vol. 3137 (2022) 227–237.

[18] V. Grechaninov, et al., Decentralized Access Demarcation System Construction in Situational Center Network, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3188, no. 2 (2022) 197–206.

[19] B. Bebeshko, et al., Application of Game Theory, Fuzzy Logic and Neural Networks for Assessing Risks and Forecasting Rates of Digital Currency, Journal of Theoretical and Applied Information Technology 100(24) (2022) 7390–7404.

[20] A. Kuznetsov, et al., Performance Analysis of Cryptographic Hash Functions Suitable for Use in Blockchain, Int. J. Comput. Netw. Inf. Secur. 13 (2021) 1–15. doi:10.5815/IJCNIS.2021.02.01.

[21] A. Kuznetsov, et al., Performance of Hash Algorithms on GPUs for Use in Blockchain, IEEE Int. Conf. Adv. Trends Inf. Theory (2019) 166–170. doi: 10.1109/ATIT49449.2019.9030442.

[22] A. Kuznetsov, et al., Statistical Testing of Blockchain Hash Algorithms, in: International Workshop on Conflict Management in Global Information Networks vol. 2588 (2019).