Scientific and Practical Cyber Security Journal (SPCSJ) 9(3): 115 – 120 ISSN 2587-4667
Scientific Cyber Security Association (SCSA)

A DUAL-ARCHITECTURE HASHING FRAMEWORK WITH INTERLEAVED BLOCK MIXING FOR ENHANCED SECURITY AND ENTROPY

Vedant Pillai[1]
[1]Department of Computer Engineering, K J Somaiya Institute of Technology, India

ABSTRACT. Hash functions are used to ensure integrity and confidentiality of sensitive data. They are crucial to ensure that even if sensitive data is leaked, it would be in a non-readable and non-exploitable format. From a security perspective, hash functions can be divided into 2 types - Cryptographic Hash Functions and Non-Cryptographic Hash Functions. Cryptographic hash functions are primarily used in applications that demand security such as digital signatures, certificate verification and password storage. Over the years, many cryptographic hash functions like MD5, SHA-1, SHA-256, BLAKE2 and RIPEMD-160 have been introduced to provide enhanced security. MD5 (Message Digest 5), once a widely used hashing algorithm, is now deprecated due to the presence of collision and preimage attacks. Despite its deprecation, it is still used in many applications due to its simplicity and performance efficiency. Non-Cryptographic hash functions are commonly employed in applications where the primary goal is not security but speed and performance of the application. xxHash is a non-cryptographic hash function which is known for being extremely fast, working at RAM speed limits. It is widely used for non-security tasks like checksums and indexing. Thus, in this paper, a hybrid hashing algorithm is proposed which integrates the legacy compatibility of MD5 with the speed and entropy characteristics of xxHash. The proposed system can be used in applications requiring improved security while not decreasing their performance.

KEYWORDS*: Hybrid hash, Cryptographic functions, Entropy enhancement, Block mixing, MD5 replacement, Dual architecture, Secure hashing, Software Security*

1. INTRODUCTION

In today's world, hashing algorithms play a crucial role in ensuring data integrity and enabling trust in digital signatures. Hash functions are used in a wide range of applications - from verifying message authenticity and saving data integrity to storing sensitive data in a secure and irreversible format. Hashing algorithms can be categorized as Cryptographic Hash Functions and Non-Cryptographic Hash Functions. Applications that prioritize security over all other factors utilize Cryptographic Hash Functions to ensure data integrity and confidentiality. An ideal Cryptographic Hash function must produce a fixed size output, operate efficiently, resist collisions, be deterministic and avoid preimage attacks. Ideal Cryptographic Hash Functions show the Avalanche Effect, where a single bit change in input gives a completely different output. Over the years, many hash functions like MD5, SHA-1, SHA-3 and BLAKE2 have been opted by different applications due to their simplicity and compatibility.
However, the integrity of many hashing algorithms has come under scrutiny over time. One of these hash functions was MD5 (Message Digest 5), which was widely used due to its simplicity and computational efficiency. Despite its early success, it was soon deprecated after the demonstration of practical Collision Attacks, where 2 distinct inputs give the same output hash. These vulnerabilities compromise the core principle of Cryptographic Hash Functions – Collision Resistance.
In contrast, Non-Cryptographic Hash Functions like xxHash are used by applications that prioritize speed over the security of applications. It was designed mainly for extremely-fast performance, for use cases like checksum validation, file deduplication and hash tables. However, it is deemed unsuitable for applications that prioritize security over speed, as it lacks cryptographic properties required for a secure environment.

Amidst these contrasting use cases, the industry has been facing a high need for hash functions that are lightweight and secure. As the world evolves, with the prevalence of resource-constrained devices like IOT devices and mobile systems, there has emerged a great need for hashing algorithms that provide a higher bit length output (256 bits digests) while maintaining performance efficiency. One critical use case where all these requirements converge are systems that require verification of Digital Signature. Digital Signatures are used to verify the data authenticity and integrity. In such systems, a hash of the original message can be used for the signing and verification process. In such scenarios, the authenticity and integrity of the original data relies completely on the strength of the hash function. If the hash function is vulnerable to collision attacks or takes too long to compute, then the whole signature scheme can be compromised or impractical for performance-sensitive applications. Therefore, such systems must use hash functions that strike a balance between security and speed, ensuring that digital signature systems and other cryptographic systems remain secure and efficient in the modern dynamic technological landscape. The proposed system in this paper can be used as a lightweight, secure and fast hashing algorithm to ensure security in such applications.

2. LITERATURE REVIEW

One of the major keystones of cybersecurity is Cryptography, which provides essential mechanisms for data security. Within this landscape, hash functions play a vital role by generating fixed length hash values, which enables efficient data verification without a need of storing the input data. Hashing is crucial in distributed and cloud-based systems like Internet of Transportation Systems (IoTS), where massive volumes of vehicular and infrastructural data are continuously transmitted and processed.

Among the traditional hash functions, MD5 has been often used for data verification, digital signatures and authentication. It's widespread is owed to its speed and simplicity. However, MD5 has been proven to be susceptible to collision and preimage attacks, as shown in real-world demonstrations [10]. Despite its limitations and flaws, many security researchers have sought to strengthen MD5 for specific applications. Yao et al. [3] proposed the idea of making an improved salting algorithm for the traditional MD5 algorithm which would make it very difficult for an attacker to crack. Kumar and Venkaiah [6] proposed an extended version of MD5 algorithm, QGMD5, which is used as an optimal quasigroup along with two operations named as QGExp and QGComp. In addition, Kumar et al. [8] showed MD5's role in low security applications where resource optimization is essential, and Shaikh et al. [9] applied MD5 for image forgery detection.

In contrast to cryptographic hashes like MD5, xxHash[1] is an extremely fast, non-cryptographic hashing algorithm, primarily designed for fast checksums and duplicate detections. This algorithm is used the most in applications where speed is essential. Mahajan and Bedi [5] used xxHash effectively to limit duplicate content in large-scale data systems, showcasing its use for file indexing, content filtering, and lightweight integrity checks. However, due to its non-cryptographic nature, xxHash alone cannot guarantee protection against deliberate data manipulation or tampering.

Recognizing the complementary strengths of both cryptographic and non-cryptographic hashing algorithms, recent research has shown that use of dual hash mechanisms, where 2 independent hashing algorithms over a single input can give enhanced security and performance. Samundi et al. [2] investigated this approach in the Cloud based IoTS framework, where the researchers proposed the idea of using 2 different independent hashes and combining their output hashes in a similar way to double hashing in hash tables, ensuring uniqueness in positioning and enhancing security. This dual-hash design not only improves the overall strength against attacks targeting a single hash function but also enables more efficient storage and transmission of integrity metadata in resource-constrained transportation environments.

3. PROPOSED SYSTEM

The proposed system introduces a hybrid, keyless hashing algorithm by using 2 structurally and algorithmically unique hash functions: MD5 and XXH3-128(Unseeded 128-bit variant of XXH3). Due to its vulnerability to collision attacks, MD5 has been long deprecated from use. XXH3-128 is a modern, non-cryptographic hash well known for extremely fast performance and good distribution. By interleaving their outputs in a systematic manner, the proposed algorithm mitigates their individual weakness while increasing the overall strength of the algorithm. The proposed system is designed to improve entropy in the final digest output and enhance resistance against collisions.

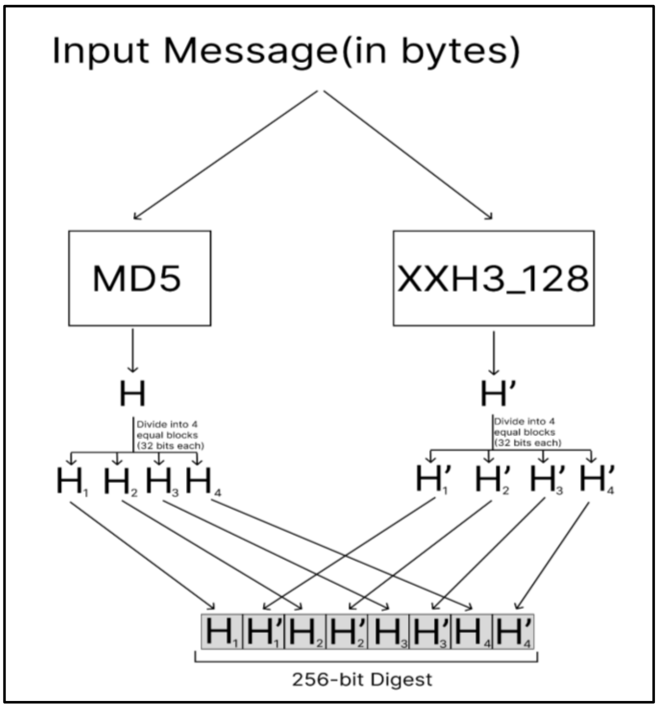

Fig.1. Algorithm Working Flowchart

Steps for the hash function according to the algorithm flowchart given in Figure 1:

1. The input data is converted into bytes using UTF-8 encoding.
2. The byte input is hashed using MD5, resulting in a 128-bit hexadecimal string. The same byte input is hashed using XXH3-128, also yielding a 128-bit hexadecimal string.
3. The 2 hash outputs are divided into 4 equal blocks (32-bits each)
4. The final hash is constructed by alternating blocks from MD5 and XXH3. That is, the first block of MD5 is followed by the first of XXH3, the second of MD5 followed by the second of XXH3, and so on.
5. These interleaved blocks are joined into a final 256-bit (64 hexadecimal characters) hash string.

The pseudocode of the hybrid hash function is shown in Figure 2.

```
function hash_input(input_string):
    input_bytes = UTF8Encode(input_string)

    md5_hash = MD5(input_bytes)
    xxh3_hash = XXH3_128(input_bytes)

    md5_parts = split_into_4_parts(md5_hash)
    xxh3_parts = split_into_4_parts(xxh3_hash)

    final_hash = []
    for i from 0 to 3:
        final_hash.append(md5_parts[i])
        final_hash.append(xxh3_parts[i])

    return concatenate(final_hash)
```

Fig.2. Pseudocode of Hybrid Hash

4. RESULTS AND ANALYSIS

To test the hashing algorithm effectiveness, the following experimental setup was used:

1. Test set: 50,000 randomly generated inputs
2. Size of each input was 512 bytes
3. Evaluation Metrics were Collision detection and bit-level uniformity
4. Programming Language used: Python

The results of the test showed that 0 collisions were found. Moreover, we performed a Bit-Level Bias Analysis. In this, each of the 256 output bits was analyzed to assess frequency of being set to 1.

1. Ideal Probability: 0.5
2. Observed Range: 0.4931 to 0.5048
3. Mean Value: Approximately 0.5

The Bit-Level Bias analysis shows us that bit frequencies closely approximate the ideal Bernoulli distribution, reflecting an unbiased and statistically uniform output, showing high entropy and low predictability.

To assess the bit-level diffusion, it was tracked that how many times each bit (from bit position 0 to 255) flipped when a single-bit change was introduced in the input. This process, known as the avalanche test, simulates how sensitive the hash output is to minimal changes to any imput. The result, visualized in Figure 3, presents a uniformly distributed frequency of bit flips, with each bit flipping approximately 500 times across 1,000 perturbation trials per bit — exhibiting high adherence to the avalanche property.

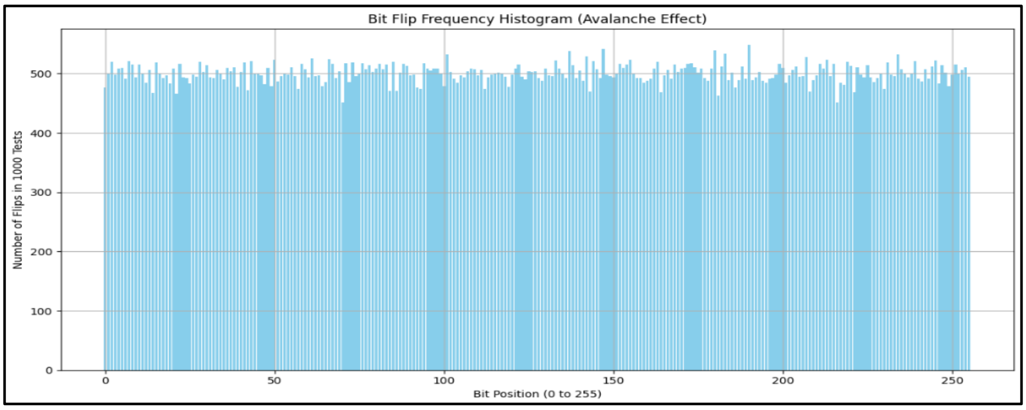

Fig.3. Bit Flip Frequency Histogram

In parallel, the entropy and uniformity of the output bits is shown in Figure 4, which plots the probability of bit flips across all 256 output positions. The observed values oscillated closely around the ideal flip probability of 0.5, with fluctuations bounded between approximately 0.46 and 0.54. Such results demonstrate that no specific bit position is more stable than others, ensuring that the final digest does not leak structural information about the original input. This near-ideal dispersion of bit flips confirms the hash function's strong diffusion properties. Together, both graphs show that the hybrid approach not only effectively spreads input entropy but also neutralizes deterministic vulnerabilities, validating the theoretical design with empirical robustness.

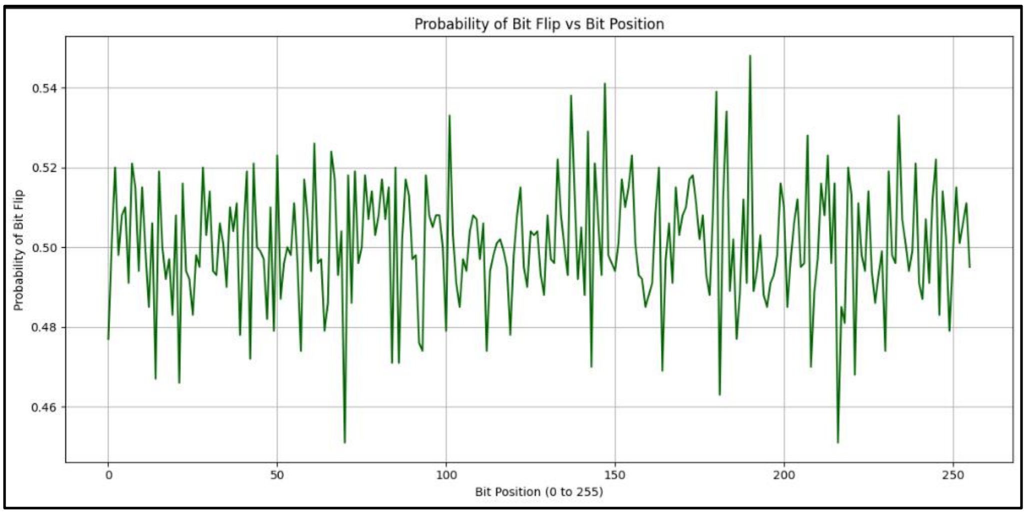

Fig.4. Bit Flip Probability v/s Bit Position Graph

The hash function was also tested against the known collision attacks of MD5. The results are shown in Figure 5. For testing purposes, we used 2 files - hello and erase, which have the same MD5 hashes [10].

Fig.5. Bit Flip Probability v/s Bit Position Graph

The result came to be that the hybrid function nullifies MD5 collisions by injecting entropy from XXH3-128. This shows that despite identical MD5 hashes, XXH3-128 yielded differing digests due to its sensitivity. The hybrid construction demonstrates complete resistance to both accidental and MD5-induced collision scenarios. This has surpassed the standalone reliability of MD5, hence validating the robustness of the interleaving strategy.

CONCLUSION

The proposed system exhibits strong empirical performance by combining MD5 and XXH3-128 into a unified, keyless 256-bit hash function. Extensive testing across 50,000 inputs revealed zero collisions, near-perfect bit balance, and high entropy dispersion, validating its effectiveness in avoiding structural biases and enhancing input sensitivity. By interleaving outputs from two fundamentally different algorithms, the system successfully mitigates MD5 collision vulnerabilities while maintaining high throughput and entropy. In essence, it strikes a balance: outperforming MD5 in every critical metric, achieving entropy metrics comparable to SHA-256 and BLAKE2, and maintaining speed comparable to them.

REFERENCES

1. "xxHash - Extremely Fast Non-cryptographic Hash Algorithm," n.d. https://xxhash.com/.
2. R, Samundi, Gunanidhi G S, and Karthikeyan R. "Cyber Security and Dual Hash Mechanism for Cloud-Based Internet of Transportation System." *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, November 22, 2023, 1519–21. https://doi.org/10.1109/iceca58529.2023.10394985.
3. Yao, Xiaosong, Ran Liu, Runan Yang, Wen Liu, You Xue, Shengbiao Cao, and Zhigang Wang. *Research and Improvement of Secure Hash Algorithm Based on MD5*. *2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, 2023. https://doi.org/10.1109/icbaie59714.2023.10281223.
4. Zhuoyu, Han, and Li Yongzhen. "Design and Implementation of Efficient Hash Functions." *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)* 1 (January 21, 2022): 1240–43. https://doi.org/10.1109/icpeca53709.2022.9719176.
5. Mahajan, Rahul, and Rajeev Bedi. "Reducing Duplicate Content Using XXHASH Algorithm." *International Journal of Science and Research (IJSR) (Vol.3, No. 7)*, January 1, 2014. https://www.ijsr.net/archive/v3i7/MDIwMTQ5NTE=.pdf.
6. Kumar, Umesh, and V. Ch. Venkaiah. "A New Modified MD5-224 Bits Hash Function and an Efficient Message Authentication Code Based on Quasigroups." In *Lecture Notes in Networks and Systems*, 1–12, 2022. https://doi.org/10.1007/978-981-16-8664-1_1.
7. K, Kavin Kumar, Nandhini V D, Kowsalya M, Ragu P J, Raghapriya N R, and Maheswaran S. "Designing and Validating an MD5 Hash Algorithm Using the UVM Verification Framework and Checkers." *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, July 6, 2023. https://doi.org/10.1109/icccnt56998.2023.10307798.
8. Kumar, K. Kranthi, E. Ramaraj, B. Srikanth, A. Srinivasa Rao, and Pbvn Prasad. "Role of MD5 Message-Digest Algorithm for Providing Security to Low-Power Devices." *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, May 25, 2022, 352–58. https://doi.org/10.1109/iciccs53718.2022.9788249.
9. Shaikh, Mohammad Shahnawaz, Aparajita Biswal, Akruti Pandwal, Nilesh Khodifad, and Bhavesh Vaghela. "Image Forgery Detection Using MD5 & Open CV." *2024 International Conference on Emerging Research in Computational Science (ICERCS)*, December 12, 2024, 1–8. https://doi.org/10.1109/icercs63125.2024.10895348.
10. "Peter Selinger: MD5 Collision Demo," n.d. https://www.mscs.dal.ca/~selinger/md5collision/.