Design and Analysis of a Cryptographic Hash Function Incorporating Parallel Confusion and a Multi-Compression Architecture

Yijun Yang
Guangdong Hong Kong Macao Greater Bay Area Artificial Intelligence Research Institute, Shenzhen Polytechnic University
Linlin Wang
Shenzhen Polytechnic University
Meileng Yuan
Shenzhen Polytechnic University
Bin Li
Shenzhen Polytechnic University
Zhuolin Zhong
Shenzhen Polytechnic University
Xiaohu Yan
yanxiaohuszpu@163.com
Shenzhen Polytechnic University

Research Article

Keywords: cryptographic hash function, parallel confusion, multi-compression, collision-resistance

Posted Date: September 5th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4884979/v1

Additional Declarations: No competing interests reported.

Design and Analysis of a Cryptographic Hash Function Incorporating Parallel Confusion and a Multi-Compression Architecture

Yijun Yang[1,3], Linlin Wang[3], Meileng Yuan[3],Bin Li[3], Zhuolin Zhong[3], Xiaohu Yan[2,3]

1 Guangdong Hong Kong Macao Greater Bay Area Artificial Intelligence Research Institute, Shenzhen Polytechnic University, Shenzhen, Guangdong, 518060, China

2 School of Undergraduate Education, Shenzhen Polytechnic University, Shenzhen, Guangdong, 518055, China

3 School of Artificial Intelligence, Shenzhen Polytechnic University, Shenzhen, Guangdong, 518055, China.

** Corresponding Author Email: yanxiaohuszpu@163.com*

Abstract: The cryptographic hash function stands as a cornerstone among the trio of essential cryptographic algorithms that are ubiquitously utilized across blockchain technology, digital signature applications, cloud storage solutions, and numerous other domains. Currently, a series of MD4-inspired hash functions, including RIPEMD, RIPEMD128, MD5, and SHA-1, have been critically evaluated and deemed insufficient in terms of security[10-13], thereby emphasizing the paramount importance of heightened vigilance towards safeguarding the integrity of cryptographic hash functions. Notably, the preponderance of prevalent hash functions relies heavily on inefficient serial architectures, posing limitations in terms of performance and scalability. To address these shortcomings, this paper introduces a groundbreaking cryptographic hash function, predicated on a parallel confusion and multi-compression structure (PCMCH). This innovative methodology innovatively fills the input data through a parallel confusion compression mechanism, concurrently executing multi-faceted confusion compression on each message block. Furthermore, it expedites message diffusion by meticulously tuning adaptable permutation parameters, enhancing both the speed and efficacy of the process. The exhaustive experimental analysis conducted underscores the exceptional security characteristics of the proposed hash function, including irregularity, the avalanche effect, high information entropy, and robust collision resistance. Moreover, its performance

surpasses that of existing parallel hash functions, marking it as a promising contender that offers superior efficiency and security, thereby presenting a viable alternative for applications requiring heightened cryptographic safeguards.

Keywords: cryptographic hash function, parallel confusion, multi-compression, collision-resistance

1 Introduction

1.1 Background

Cryptographic algorithms occupy a pivotal position in the realm of information security, universally acknowledged as a cornerstone of data protection. These algorithms encompass three principal classifications: symmetric encryption systems, which operate on the premise of a singular, shared key for both encrypting and decrypting data; public key encryption systems, distinguished by their employment of a key pair—one dedicated to encryption and the other to decryption, fostering secure communications; and cryptographic hash functions, a unique breed that safeguards data through an irreversible, keyless process.

Hash functions are indispensable security mechanisms that have found ubiquitous application in diverse fields including digital signatures, blockchain technology, the construction of cryptographic primitives, and the verification of data integrity. They perform a transformative feat, condensing variable-sized data into a concise, fixed-length hash value—a digital fingerprint that uniquely identifies the data (be it documents, images, or messages) and ensures its authenticity and unalterability.

1.2 Motivation and objectives

Hash functions are paramount in ensuring the inviolability of data integrity. In the context of cloud audit schemes, their employment necessitates vigilance against a range of threats, including collision vulnerabilities [2,3,10], linear reconstruction attacks [20], and forgery attempts [32]. The quest for a highly efficient hash function is imperative, as real-time verification stands as a cornerstone for the majority of

cloud-centric applications. As data volumes escalate, the need for instantaneous authentication of cloud-resident data intensifies, catering to the performance demands of diverse applications encompassing cryptographic primitive construction, integrity validation, and blockchain technology. Blockchain technology, initially conceived for transaction verification in Bitcoin, represents a continually expanding chain of records, known as blocks, intricately linked and fortified by hash functions. This immutable ledger ensures that once data is recorded within a block, it cannot be retroactively altered without simultaneously modifying every subsequent block, thereby underscoring the paramount importance of hash function security. Each transaction within the blockchain is uniquely identified through the application of a double SHA256 hash function. This process challenges miners to solve a cryptographic puzzle, involving the computation of a double SHA256 hash digest $H = \mathrm{SHA256}\big(SHA256(H' + a + n)\big)$, where H must possess a predefined number of leading zeros. Successfully accomplishing this task rewards the miner, fostering a secure and decentralized transaction verification mechanism. The string 'n' serves as a randomized 'nonce' value, integral to ensuring uniqueness and security. At the forefront of the current blockchain iteration lies 'H^', the block solution that anchors the chain's integrity. Meanwhile, 'a' represents the input for the intricate cryptographic challenge that safeguards the system. The irreversibility and collision resistance of the hash function employed within the blockchain are paramount; without them, an attacker might exploit SHA256's vulnerabilities to falsify inputs, manipulate leading zeros, and execute Bitcoin transactions without accountability, subsequently regenerating blocks to erase incriminating history. Blockchain, armed with a robust hash function, has permeated numerous industries, revolutionizing operations. Qschou.com, a crowdfunding platform dedicated to the well-being of critically ill patients, embraces blockchain technology to streamline processes and foster unparalleled transparency. Donors, empowered by blockchain, can trace their contributions, assuring that every penny is directed towards its intended cause. Notably, industry giants such as IBM and Maersk have commercialized and scaled

blockchain solutions, catering to a global audience eager to harness their potential. This has culminated in the establishment of a shared, immutable ledger, which not only underpins emerging digital currencies but also serves as a formidable tool for managing intricate, interconnected ecosystems. Over the past 3-5 years, blockchain data storage has emerged as a formidable disruptor, challenging the status quo of centralized cloud services. Storj, a beta-stage cloud storage system, represents the cutting edge of this transformation, leveraging blockchain's decentralized nature to bolster security and diminish reliance on traditional models. Users can now monetize their unused storage capacity, akin to the Airbnb model, fostering the creation of innovative marketplaces. In this evolving landscape, the significance of digital identity verification, underpinned by a secure hash function, cannot be overstated. All blockchain applications must adhere to rigorous standards, ensuring that identities are verified with unwavering accuracy and security. As the adoption of blockchain technology accelerates, so too must the development of robust, future-proof security measures to safeguard against emerging threats.

However, in the realm of security, prevailing hash functions exhibit vulnerabilities to message extension attacks and secondary collision attacks, posing significant challenges. Furthermore, as message lengths escalate, enhancing the efficiency of serial hash functions becomes increasingly arduous. In the context of cloud storage and blockchain, crafting a hash function that harmoniously balances security and efficiency stands as a formidable undertaking, thereby rendering the compilation of secure hash algorithms both imperative and highly sought-after, igniting a surge of research enthusiasm.

1.3 Related work

The primary objectives of research on hash functions are to enhance their collision resistance, randomness, and avalanches. Investigations into the security of hash functions can be categorized into three distinct areas:

Foremost among considerations lies the paramount importance of ensuring the security of contemporary hash functions. Over the past two decades, the relentless

scrutiny of these functions has intensified, with a prevailing emphasis on safeguarding against collision attacks that target prevalent hash functions. Remarkable strides have been made in this realm, unveiling vulnerabilities in nearly all hash functions belonging to the MD and SHA families (barring SHA3), which are now known to be susceptible to such attacks. Historically, hash functions were instrumental in the generation of digital signatures and message verification codes, adhering to the tripartite security criteria of irreversibility, weak collision resistance, and strong collision resistance. Rivest's seminal work in introducing the efficient MD4 hash function marked the cornerstone for both the MD and SHA series, fulfilling these criteria and fostering widespread adoption[1]. Nevertheless, the algorithm's Achilles' heel was exposed by Boer[2], leading to its successful compromise by Dobbertin[3], necessitating the development of MD5 as a fortified successor[4]. Yet, even MD5 proved vulnerable to collisions uncovered by Dobbertin through a modular differential attack[5], prompting the invention of RIPEMD128 and RIPEMD160 by Dobbertin himself, essentially refined variants of MD5[6]. In 1995, the NSA ventured into the fray with the introduction of SHA0, though its own design flaws necessitated the swift release of SHA1[8] in the same year and SHA2[9] in 2001. Despite documented collision instances for MD5, it continues to hold sway in numerous security protocols. This status quo was shaken in 2005 when Wang leveraged innovative analysis techniques to mount collision attacks on an array of hash functions, including MD4, RIPEMD, MD5, SHA0, and SHA1[10-13], spurring a reassessment of the MD and SHA series' security posture. Since then, the relentless pursuit of vulnerabilities in popular hash functions has persisted[10,14-18], underscoring the pressing need for a more robust and efficient alternative. The development of such a hash function has consequently emerged as a pivotal research endeavor and a matter of utmost urgency.

Secondly, the evolution of hash iterative structures has been marked by significant advancements. The initial iterations largely adhered to a serial paradigm, encompassing structures like the seminal MD framework[19], alongside innovations like Wide-Pipe[20], Double-Pipe[20], HAIFI[21], and the groundbreaking Sponge

design[22]. Notably, the MD structure served as the cornerstone for both the MD and SHA series[19], while subsequent iterations in Ref[20-21] refined this foundation. The Sponge structure introduced a novel concept, comprising absorption and extrusion phases, which were subsequently incorporated into SHA3[22]. Over time, numerous modifications and enhancements have been proposed for these iterative structures. Examples include the Wide-Pipe-inspired GOST-R modification[23], the Double-Serial iterative structure[24], a sponge function integrated with 3D-ECM[25], an iterative block structure based on controlled alternate quantum walks[26], a provably secure keyless iteration leveraging subsets and two-variable polynomial functions[27], a keyed iteration rooted in complex quadratic mappings[28], and the innovative MDPH iteration[29]. Concurrently, researchers have vigorously pursued parallel structures to bolster hash function efficiency. These endeavors encompass a multi-compression-based parallel iteration[30-31], a shuffle-exchange network-driven parallel structure[32], Chebyshev-Halley method-based parallel hashes with variable parameters[33], a novel chaotic system-inspired parallel hash[34], a coupled mapping lattice-based parallel function[35], a tree-patterned parallel hash leveraging internal variable input lengths[36], and the innovative parallel d-pipeline cuckoo hash algorithm[37]. While these parallel structures have undoubtedly contributed to varying degrees of efficiency gains, they also pose a delicate balance between efficiency and security. The quest for optimal performance often necessitates careful consideration of the potential security implications, ensuring that efficiency enhancements do not inadvertently compromise the robustness of hash functions.

Thirdly, a vast array of researchers has diligently focused on refining hash compression functions, with the overarching goal of bolstering the security profile of contemporary hash algorithms. Ye introduced an encrypted compression framework that emphasizes replacement and diffusion mechanisms[39], while Meysam devised a dynamic chaotic compression structure, intricately tied to safety factors and functions[34]. Guesmi contributed a compression architecture tailored for the Lorenz chaotic iterative mapping network, integrating the SHA2 algorithm[40], and Kanso presented a keyed-compression function grounded in an innovative input message

structure[41]. However, critical statistical assessments have highlighted a shortcoming in these endeavors [42], revealing subpar avalanche performance across Refs [39, 34, 40-41]. In response, Karthik crafted a compression scheme leveraging polynomial functions [43], and Zhang devised a novel message extension architecture specifically for SHA2[44]. Yet, despite these advancements, both compression methods were deemed incompatible with parallel hash functions and exhibited inadequate collision resistance [42].

It is abundantly clear that existing hash functions are beset with inherent constraints. Some persist in utilizing serial iteration structures, which can prove detrimental when confronted with vast data sets. Other parallel hash functions, while enhancing efficiency, often compromise security or lack rigorous security analysis, rendering them unfit for practical deployment. Striking the elusive balance between security and efficiency thus remains a paramount challenge and a constant area of concern in the design and evolution of hash functions.

1.4 Paper organization

The remainder of this paper has been organized as follows. Section 2 presents a novel, efficient, and collision-resistant cryptographic hash function, followed by a performance analysis in Section 3. Lastly, conclusions are summarized in Section 4.

2 Proposed algorithm

In order to improve efficiency and security, this paper proposes a cryptographic hash function based on parallel confusion and multi-compression structure. As shown in **Fig.1**, after message preprocessing, PCMCH uses multi-threading technology to compress n 512-bit message blocks using the compression function C_1 and the confusion function C_2 at the same time.

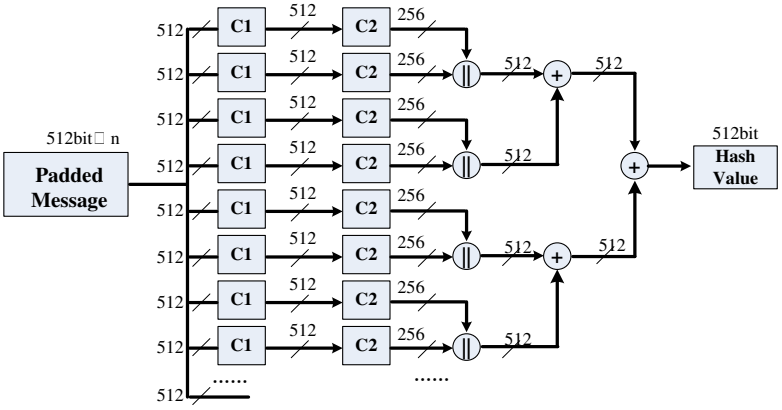

Fig.1 Schematic diagram of the main iterative framework of PCMCH

In **Fig.1**, the symbol '||' represents the message concatenation operation, for example, if $txt1 = \text{"0100"}, txt2 = \text{"1001"}$, then the result of $txt3 = txt1||txt2 = \text{"01001001"}$, the symbol '$\oplus$' represents modular addition. In terms of security, the complexity of a well-designed n-bit hash function is $2^{\frac{n}{2}}$. From the perspective of efficiency, a longer hash value will consume more storage space and computing resources, especially when hashing massive data. Therefore, the hash value of PCMCH is 512 bits.

Compression function C_1:

Step 1: Message extension

For each message block m_j, perform the following message extension operations to output 132 message words:

(1) Divide m_j into 16 32-bit message words MW_i $(i = 0, 1, \cdots, 15)$,

(2) For message *x*, define two variable parameter permutation functions:

$$\Sigma_0(x) = x \oplus ROTL_\alpha(x) \oplus ROTL_\beta(x)$$

$$\Sigma_1(x) = x \oplus ROTL_\gamma(x) \oplus ROTL_\delta(x)$$

The parameters $\alpha, \beta, \gamma, \delta$ are undetermined, $ROTL_y(x)$ is a *y*-bit loop left shift operation function. For example, if $x = \text{"0011111111111111"}, y = 2$, then the result of $ROTL_y(x) = \text{"1111111111111100"}$. In order to obtain better resistance to differential attacks described in Ref[10,24], the value of $\alpha, \beta, \gamma, \delta$ should be mutually prime to each other, therefore, the differential attack becomes increasingly hard and

difficult to continue[46].

(3) For message word MW_i $(i = 16, 17, \cdots, 131)$, the message is expanded as follows:

$$MW_i = \begin{cases} \Sigma_0\left(MW_{i-16} \oplus ROTL_5(MW_{i-3}) \oplus ROTL_1(MW_{i-9})\right) \oplus ROTL_{29}(MW_{i-4}), i = 16,17,\cdots,41 \\ \Sigma_1\left(MW_{i-16} \oplus ROTL_5(MW_{i-3}) \oplus ROTL_1(MW_{i-9})\right) \oplus ROTL_{29}(MW_{i-4}), i = 42,43,\cdots,67 \\ MW_{i-68} \oplus MW_{i-64}, i = 68,69,\cdots,131 \end{cases}$$

Step 2: Iterative compression

As soon as the number of 512-bit message blocks is established, for example, if $n = 64$, then PCMCH first define the initial link constants as:

$$\mathrm{K}_k = SHA1(k), k = 0,1,\cdots,63$$

64 constant variables K_k (the first 64 bits of the results above) are listed in **Table 1**. These constant variables are used in different Compression function C_1, which will increase the uncertainty and unpredictability of the compression function, thus enhancing the collision resistance of the algorithm.

Table 1 Specific values of K_k

356A192B7913B04C	DA4B9237BACCCDF1	77DE68DAECD823BA	1B6453892473A467
AC3478D69A3C81FA	C1DFD96EEA8CC2B6	902BA3CDA1883801	FE5DBBCEA5CE7E29
0ADE7C2CF97F75D0	B1D5781111D84F7B	17BA0791499DB908	7B52009B64FD0A2A
BD307A3EC329E10A	FA35E192121EABF3	F1ABD670358E036C	1574BDDB75C78A6F
0716D9708D321FFB	9E6A55B6B4563E65	B3F0C7F6BB763AF1	91032AD7BBCB6CF7
472B07B9FCF2C245	12C6FC06C99A4623	D435A6CDD786300D	4D134BC072212ACE
F6E1126CEDEBF23E	887309D048BEEF83	BC33EA4E26E5E1AF	0A57CB53BA59C46F
7719A1C782A1BA91	22D200F8670DBDB3	632667547E7CD3E0	CB4E5208B4CD8726
B6692EA5DF920CAD	F1F836CB4EA6EFB2	972A67C48192728A3	FC074D501302EB2B
CB7A1D775E800FD1	5B384CE32D8CDEF0	CA3512F4DFA95A03	AF3E133428B9E25C
761F22B2C1593D0B	92CFCEB39D57D914	0286DD552C9BEA9A	98FBC42FAEDC0249
FB644351560D8296	FE2EF495A1152561	827BFC458708F0B4	64E095FE763FC624
2E01E17467891F7C	E1822DB470E60D09	B7EB6C689C037217	A9334987ECE78B6F
C5B76DA3E608D34E	80E28A51CBC26FA4	8EFFEE409C625E1A	54CEB91256E8190E
9109C85A45B703F8	667BE543B02294B7	5A5B0F9B7D3F8FC8	E6C3DD630428FD54
6C1E671F9AF5B46D	511A418E72591EB7	A17554A0D2B15A66	C66C65175FECC310

Define two logical functions:

$$\mathrm{Ch}(x, y, z) = (x \cap y) \oplus (\bar{x} \cap z)$$

$$\mathrm{Maj}(x,y,z) = (x \cap y) \oplus (x \cap z) \oplus (y \cap z)$$

$\bar{x}$ is the reverse operation of x, CV_k is the intermediate link variable state at the k-th iteration, consisting of 8 registers A,B,C,D,E,F,G,H. For each message block m_j, conduct 64 rounds of iteration in the following iteration mode:

$$CV_{k+1} = C_1(CV_k, m_j), k = 0,1,\cdots,63$$

The specific iteration process can be described as follows:

$$A \leftarrow \mathrm{Maj}(A,B,C) + D + ROTL_7\left(ROTL_{12}(A) + E + ROTL_k(K_k)\right) + ROTL_{12}(A) + MW_{k+68}$$

$$B \leftarrow \mathrm{A}$$

$$C \leftarrow ROTL_9(B)$$

$$D \leftarrow \mathrm{C}$$

$$E \leftarrow \Sigma_0\left(\mathrm{Ch}(A,B,C) + H + \left(ROTL_7\left(ROTL_{12}(A) + E + ROTL_k(K_k)\right)\right) + MW_k\right)$$

$$F \leftarrow \mathrm{E}$$

$$G \leftarrow ROTL_{19}(F)$$

$$H \leftarrow \mathrm{G}$$

In this process, the input of each round of iteration CV_k comes from the output results of the previous round and message block m_j iteration compression. The initial link variable CV_0 is defined as **Table 2**:

Table 2 Initial link variable CV_0

A	6B86B273FF34FCE1	*E*	EF2D127DE37B942B
B	D4735E3A265E16EE	*F*	E7F6C011776E8DB7
C	4E07408562BEDB8B	*G*	7902699BE42C8A8E
D	4B227777D4DD1FC6	*H*	2C624232CDD22177

Confusion function C_2:

The confusion function C_2 is shown in **Fig.2**. For message *x*, define 16 fixed parameter permutation function:

$$\mathrm{E}_l(x) = ROTL_{l+1}(x) \oplus ROTL_{l+2}(x) \oplus ROTL_{l+3}(x), l = 0,1,\cdots,15$$

In order to increase uncertainty of message diffusion, four salt variables are added during the confusion process, which are defined as **Table 3**:

Table 3 Specific values of $Salt_t, 1 \leq t \leq 4$

$Salt_1 = SHA3\_{512}(t1)$	B4AB286C85BEC8D7
$Salt_2 = SHA3\_{512}(t2)$	13204F86F5C36835
$Salt_3 = SHA3\_{512}(t3)$	BA4791CA2302D51E
$Salt_4 = SHA3\_{512}(t4)$	F106AD2A9814DBA8

C_2 introduces parallel mechanism, it can process almost all message link variables in the iterative structure at the same time, which is very helpful for improving efficiency. In the process of confusion, each link variable will affect its adjacent link variable through factors such as salt variables, fixed parameter permutation function and loop left shift operation. This design makes it difficult to eliminate the differential difference between the two adjacent variables by the differential analysis method, thus the ability of the algorithm to resist differential collision is enhanced.

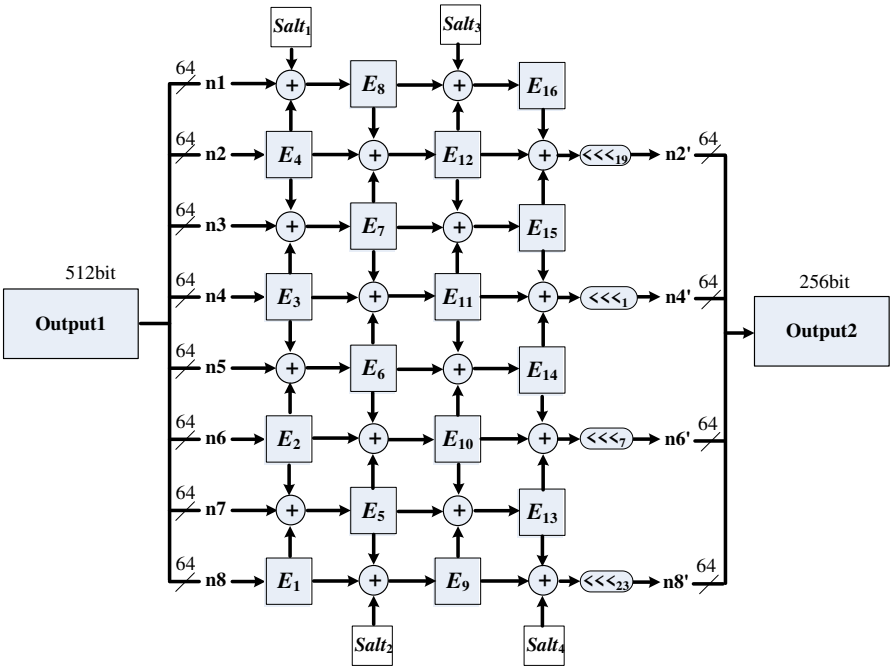

Fig.2 Schematic diagram of the confusing function iteration framework

3 Performance analysis

A secure hash function should be uniformly distributed and sensitive to small changes. This chapter will analyze and discuss the performance of the proposed PCMCH algorithm from seven aspects: random message testing, character distribution, statistical attack resistance, avalanche performance, collision resistance, information entropy and efficiency.

3.1 Random message testing

Randomly chooses a piece of text, after 6 minor changes, 7 different hash values are generated by PCMCH.

Randomly selected text: An efficient cryptographic hash function based on parallel confusion and multi-compression structure

Modification1: An efficient cryptographic hash function**,** based on parallel confusion and multi-compression structure

Modification 2: **1**An efficient cryptographic hash function based on parallel confusion and multi-compression structure

Modification 3: An efficient cryptographic hash function based on parallel confusion and mult**i c**ompression structure

Modification 4: An efficient cryptographic h**u**sh function based on parallel confusion and multi-compression structure

Modification 5: An efficient cryptographic hash function based on parallel**-**confusion and multi-compression structure

Modification 6: **a**n efficient cryptographic hash function based on parallel confusion and multi-compression structure

Table 4 shows the seven different versions of hash values and Hamming distance. **Fig.3** shows the square wave of these hash values.

Table 4 hash values and hamming distances between original and six different modified versions

Versions	Hash values	Hamming distance ($Ham(h_0, h_i)$)
original	h_0: 6e5440913226c36eb17611ec5eee380f49de5f2156e9b454 9743ca12a15ca8c551d4d1abe8deb16d5b09d3e25cbd8c67c2	/

	86dbd5748eb4f4fe7981f23610c1a4	
1	h_1: b82fd2dda8e03caae02777975c55115e43326391626dbe5 007d4e9ee7a2ce303ab26b1d721e6af28a24a15772475b598a d009b5ca15b9f81ff0d91ca3a9cb32c	240
2	h_2: aebbcfa9a27c4175639fcb1562cbf4c4010e704537aee73d 8e3593571d57f24927f85a7a9a4c636e474814820ff5e3e6799 153f8c4deaaaccafefe38c0ddf131	241
3	h_3: 0776940ca05bae0c12e61f225943f8858a66b932a6e4ee98 e0db62e31b842880051b2bb56e23e0d28c32037304d072d4b b6022a588c735d4b86816641b6295aa	255
4	h_4: 4111f6742505388e8b36ba212c7fea0c818abc470610e21f c4629e0b2ec364b7b71adac7f87a500e45a9745a13f7d170151 ca93dd935487ce0d7cbc34d5c0fb7	255
5	h_5: 9eff679bc7795e64fc613a146a20400a55769db6c1b46e40 b883d17ecad564caabafbd4f5a2e6d0655a12e4d6455a0fa552 c2f5ab5d868a2dcf44f34850b3a93	270
6	h_6: d042e44456ddce4a2d508b2d2fbf4665a33940d2f158b09 c993451183562067157d25f90f02dcb788ba9c6ac45fe504300 245daf34a3f206b7f5581bef5c0547	244

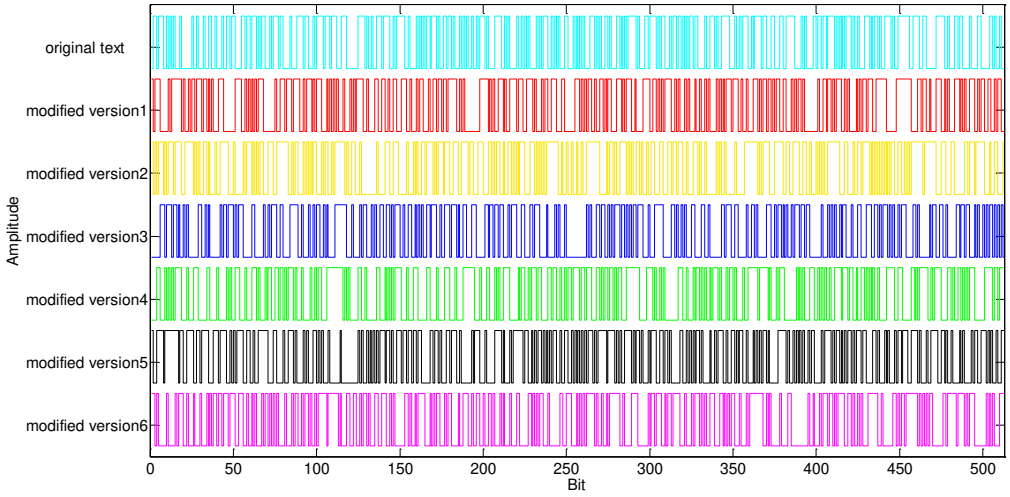

Fig.3 square wave representation of 7 hash value

In **Fig.3**, each hash value is irregular and very different from others. Hamming distance $Ham(h_0, h_i)$ is usually used to analyze the difference between two hash values. In a collision resistant hash function, it will be difficult to find two different messages satisfying $Ham(h_0, h_i) = 0$. As the number of experiments increases, the Hamming distance between two randomly generated n-bit hash values should be approximately close to $\frac{n}{2}$.

Table 4 shows that after making minor modifications to a randomly chosen text, the Hamming distance between two hash values is basically around 256. In 1 million non-repetitive experiments, no collision was found. The minimum and maximum values of Hamming's distance of these 1 million hash values are 200 and 310 respectively, indicating that the PCMCH proposed in this article can resist birthday attacks.

3.2 Character distribution

Confusion and diffusion are two basic attributes that need to be taken into account in a well-designed hash function. The confusing attribute blurs the statistical result relationship between input and output, making the statistical analysis attack more difficult; while the diffusion attribute makes the input of each bit directly affect each intermediate value of the whole iteration process.

This section evaluates the confusing attribute of PCMCH by analyzing the character distribution of different hash values obtained by 100,000 different randomly chosen inputs. PCMCH firstly converts each 512-bit hash output to 128 hexadecimal single characters, and then counts the distribution of hexadecimal single characters in all 100,000 hash outputs.

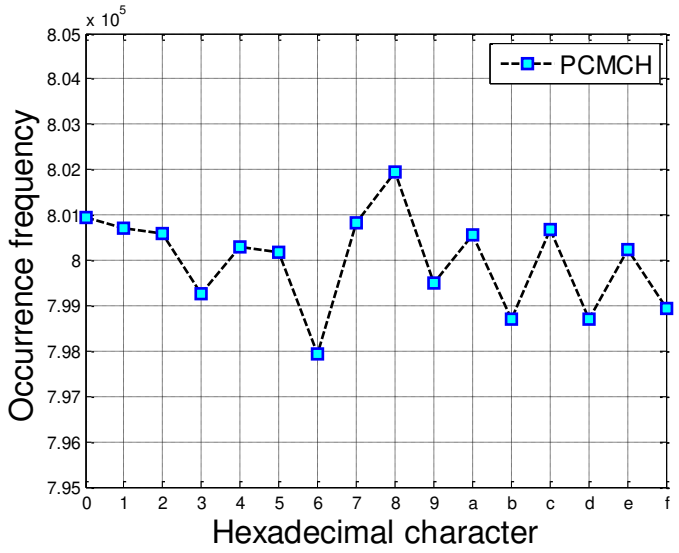

Fig.4 Distribution of hexadecimal characters in 100000 hash output

Theoretically, the hexadecimal character distribution of a hash function with good confusion property should basically obey uniform distribution[30]. In **Fig.4**, the distribution frequency of each hexadecimal character has relative errors less than 0.26%, which shows that the regularity of the corresponding relationship between the input and output of PCMCH is difficult to be analyzed.

3.3 Statistical Analysis Attack

A common attack on hash functions is statistical analysis attack. This selective plaintext attack means that a well-designed hash function should be able to generate pseudo-random and unpredictable hash values to resist statistical analysis attack. Theoretically, the probability that '0' and '1' in output hash value should remain equal, once the input is modified, the probability of each bit reversal should be 50%.

This section discuss the process of the statistical analysis attack of PCMCH as follows: randomly select an input, then randomly flip 1 bit of the input, carry out the PCMCH hash operation on these two different inputs, and then calculate the Hamming distance. After 1,000,000 different comparisons, the experimental results are shown in **Fig.5**.

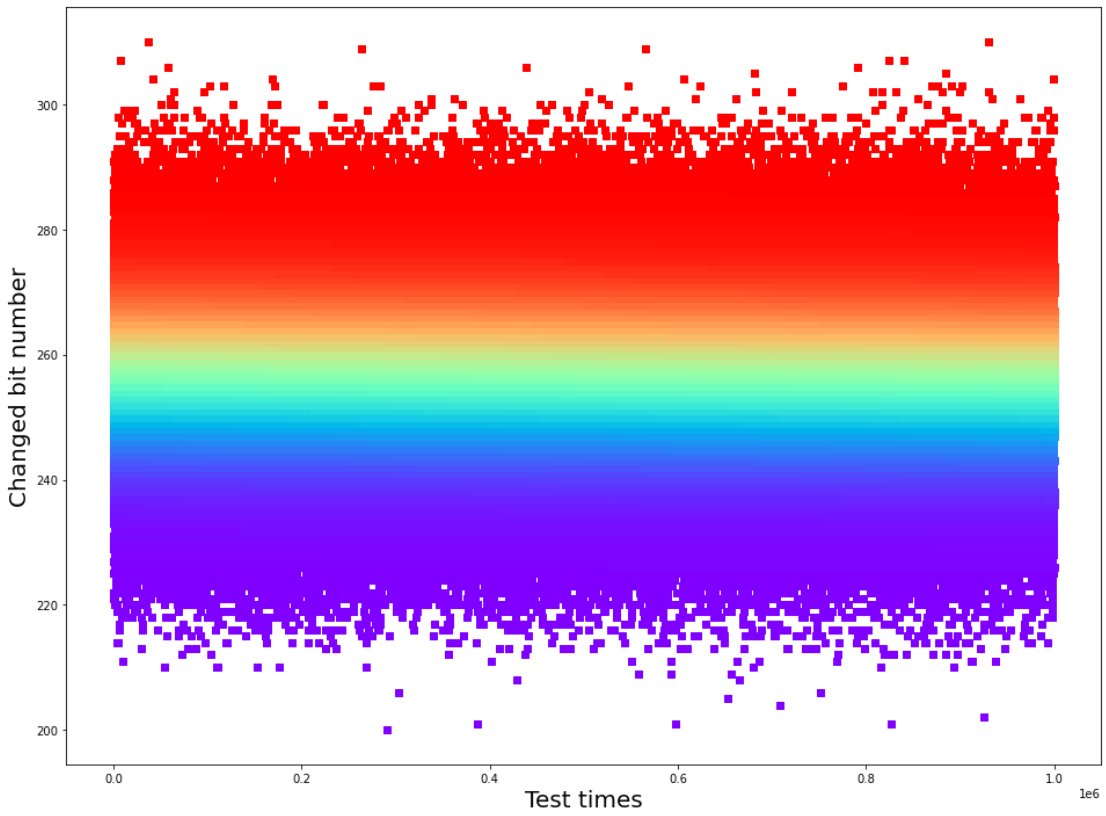

Fig.5 1,000,000 repeated flip experiments

From the statistical results in the figure, it can be seen that with PCMCH, the Hamming distance of each output hash value is concentrated around 256, and the standard deviation and variance are extremely low. If the number of test samples continues to be increased, the distribution of Hamming's distance will gradually obey the Gaussian distribution, which is the theoretical optimal random distribution model, as shown in **Fig.6**.

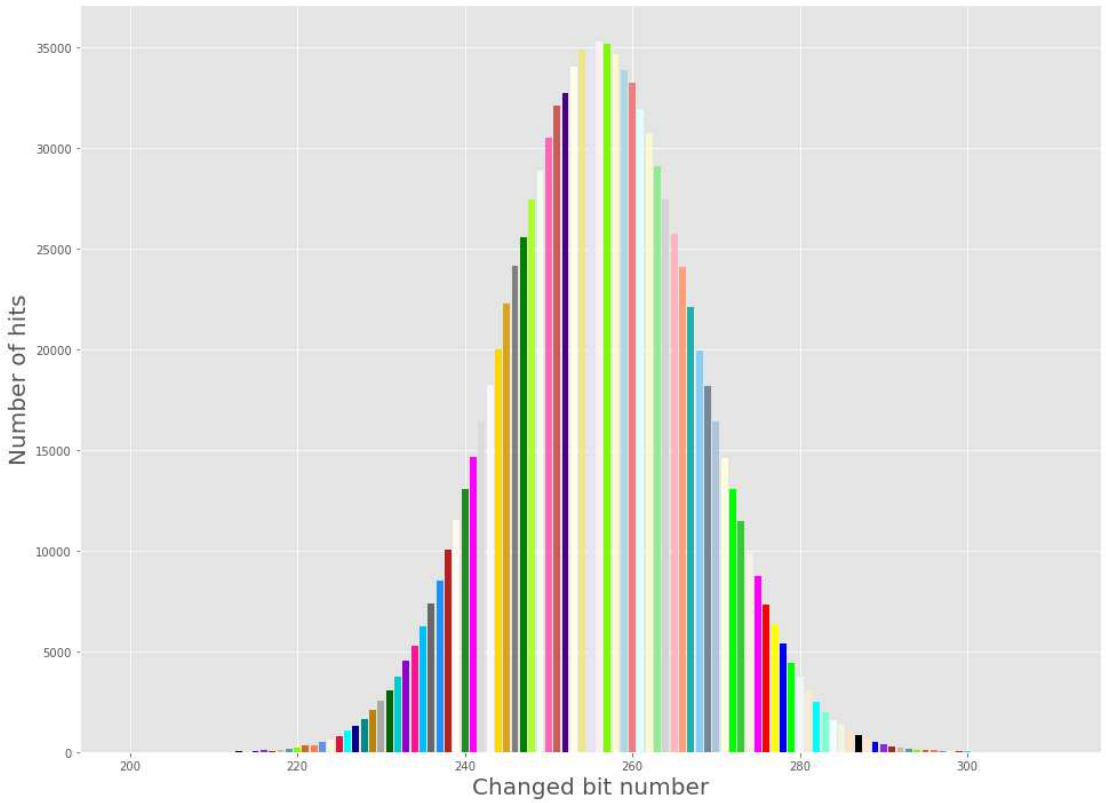

Fig.6 1,000,000 experiments on the approximate Gaussian distribution of Hamming distance

3.4 Analysis of information entropy

Information entropy is the basic concept of information theory, meticulously characterizing the inherent uncertainty surrounding various potential events emanating from an information source. Drawing profound inspiration from thermodynamics, Shannon introduced the concept of information entropy, defining it as the average quantity of information that remains after meticulously stripping away all redundancies within the information itself. This seminal proposal of information entropy revolutionized the landscape by enabling the quantitative measurement of information, a previously elusive feat.

Broadly speaking, the entropy of a hash value's information bears a direct, proportional relationship with the duration required to mount an attack against the underlying hash function through the meticulous analysis of its inherent patterns and regularities. Consequently, the irregularity or unpredictability of a hash function can be precisely quantified by assessing the value of its information entropy. The mathematical formula devised for this purpose, which calculates the information

entropy, is presented as follows:

$$\mathrm{H}(x) = \sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

In this formula, $\mathrm{H}(x)$ represents the information entropy of the message x, and $p(x_i)$ is the probability of x_i in the message.

In the compression function C_1, different circular left shift bits $\alpha, \beta, \gamma, \delta (1 \leq \alpha, \beta, \gamma, \delta \leq 31)$ may have a certain impact on the fluctuation of the information entropy value.

In view of this, the paper has carried out the following research:

For different messages, information entropy obtained from $32 \times 32 \times 32 \times 32 = 1048576$ different combinations is shown in **Fig.7**.

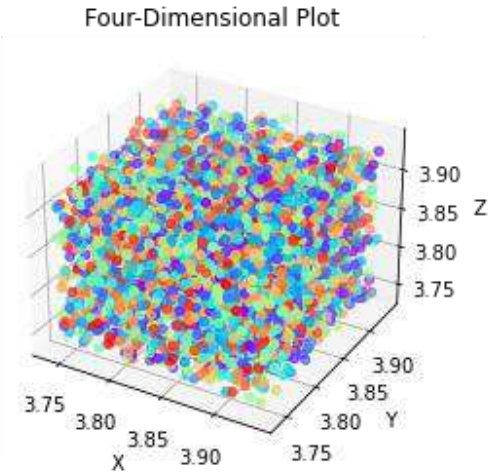

Fig.7 Information entropy under the combination of different loop left shift operations

After 10 tests, when $(\alpha, \beta, \gamma, \delta) = (23,7,12,29)$, the comprehensive value of information entropy under different inputs is the highest (about 3.9379). Therefore, the values of the different loop left shifts $\alpha, \beta, \gamma, \delta$ of PCMCH in this paper are set for this. Compared with other popular hash functions, the comparison of hash information entropy is shown in **Table 5**.

Table 5 Comparison of hash information entropy of several mainstream hash functions

Input	Algorithm	Hash value	Information entropy
01	SHA2	7b3e2f9860391685c2ff6785ab60541bb0db11a20b7e511bf020f4b3073053cc36b647d82f520c5a1c323e853f4bb110fce8210ba409fa42069ab4bf0b0d39e1	3.8777776919433204
	Ref[30]	84865a87593500aaaa29a49c382b84491eb97ac61a9264edd	3.88615148

		724aaaa81227040a557412b98841c14ed48b365f9a2f25faf7f59561d001bfa118070ec60dea8f3	2433462
	PCMCH	564e1971233e098c26d412f2d4e652742355e616fed8ba88fc9750f869aac1c29cb944175c374a7b6769989aa7a4216198ee12f53bf7827850dfe28540587a97	3.9159457082447844
2	SHA2	40b244112641dd78dd4f93b6c9190dd46e0099194d5a44257b7efad6ef9ff4683da1eda0244448cb343aa688f5d3efd7314dafe580ac0bcbf115aeca9e8dc114	3.8785632783009545
	Ref[30]	093763d6b1457e9fb44eeb346737783ad13bb739370d4a7d42d9cade8da9040b8c64e525aa0f2afde2dd5a98e763fbde69a679098603083c6a76fa2f66aa1188	3.8957685765013275
	PCMCH	cb7d4cb9b1b1fd3a28bf8db849c1d9a1473f7fd40e45e1ef965c62e7fde3a35b5b764597ad46274b10e67682f205b7ceaa8180eb3c686dfcd748f83f488af8fa	3.9327926984442344
3	SHA2	3bafbf08882a2d10133093a1b8433f50563b93c14acd05b79028eb1d127990272414509806519945014 23a66c276ae26c43b739bc65c4e16b10c3af6c202aebb	3.9001188992685 11
	Ref[30]	a6197943ca1052eee61e366a88d2dce2cef3352101c71883ea2253a1841c9b7c08c213cbb233e76829639e2b9e97bb878a91fc9bf958a3b5f1d58d8df6ac01a9	3.8888743836026514
	PCMCH	a321d8b405e3ef2604959847b36d171eebebc4a8941dc70a4784935a4fca5d5813de84dfa049f06549aa61b20848c1633ce81b675286ea8fb53db240d831c568	3.9388383702346954
d	SHA2	48fb10b15f3d44a09dc82d02b06581e0c0c69478c9fd2cf8f9093659019a1687baecdbb38c9e72b12169dc4148690f87467f9154f5931c5df665c6496cbfd5f5	3.8880265882432 63
	Ref[30]	e1336fcbc60ecdebb9e54760701f2c1a7506b5f86b39270b7ac2e13a7c2ca5cf0e10826301e2c38d26a30e657a680b8c44b4765b530e5576e3cacb6d1e1ceba2	3.8577748829788665
	PCMCH	f05210c5b4263f0ec4c3995bdab458d81d3953f354a9109520f159db1e8800bcd45b97c56dce90a1fc27ab03e0b8a9af8673747023c406299374116d6f966981	3.9461955023787993
5	SHA2	06df05371981a237d0ed11472fae7c94c9ac0eff1d05413516710d17b10a4fb6f4517bda4a695f02d0a73dd4db543b4653df28f5d09dab86f92ffb9b86d01e25	3.8942822408069 55
	Ref[30]	de648a9aa148e7735a1204055271298a0ef423a5d4097de82534013d5be0e7db0952e3eab4e7ad37f301dc06be0bdced6f0a568dcd1b94ad8e5bb622f4c02079	3.9160740446657907
	PCMCH	3c9ad55147a7144f6067327c3b82ea70e7c5426add9ceea4d07dc2902239bf9e049b88625eb65d014a7718f79354608cab0921782c643f0208983fffa3582e40	3.9576120889446664
……			

1000000	SHA2	7320d878832f79700d026817c54ea8b84cf89748a9f84622c09cea59b0ad91b247fb28a66c8f815841ef2fcbc17a694704a76d7ceb9c6ec47b654b5ac80b248a	3.8765719885198577
	Ref[30]	1de9f1e3de466780868f0dc8016cc018bdbd5df2adbba9d7c1f743a1afa3951e9e86cd74cedda3f3c5e227942cccd8aa55d59b82fbd3f878cbde686a79bfdcad	3.894188278300954
	PCMCH	4d66217d8d4cda49089312a7371784d4c62ff6be5fe4a8ae4d08ca0c5f5bfb190c7a91cb26a3d05b33153ebbdf112f985913cbadcd3f5b953eccc9cbad1509a5	3.9365729062818393

From the 1,000,000 experiments in **Table 5**, PCMCH obtained the highest information entropy in 994846 experiments. The average information entropy of SHA2, Ref[30] and PCMCH in these experiments was 3.8898577439779237, 3.890005274747175, 3.937908785579928 respectively.

3.5 Avalanche

In the realm of cryptography, the hash function is imperative to adhere to the principle of the avalanche effect, which stipulates that even the slightest alteration in the input message must elicit a sweeping, pervasive change in the resulting hash value. This stringent requirement fosters a robust confusion and diffusion mechanism, thereby enhancing security. Conversely, should a hash function exhibit suboptimal avalanche performance, it invites the risk of attackers exploiting this weakness to infer segments of the input message and correlate variables via analysis of the hash output. This latent threat ultimately paves the way for localized and, ultimately, full-fledged collision attacks against the hash function, posing a significant security challenge [45].

D_I, D_E, D_S, D_F are four important parameters used to measure the avalanche of hash functions. The specific definition of these parameters and the introduction of variables are described in Ref[45].

Using the following formulas, the avalanche stability of the hash function can be evaluated as:

$$D_I = 1 - Sa\{(i,j)|b_{ij} = 0\}/(mn)$$

$$D_E = 1 - \sum_{i=1}^{n} \left| \sum_{j=1}^{m} 2ja_{ij}/SaX - m \right| /(mn)$$

$$D_S = 1 - \sum_{i=1}^{n} \sum_{j=1}^{m} \left| 2b_{ij}/SaX - 1 \right| /(mn)$$

$$D_F = \frac{\sum_{D_{hash}(x,y)=1} D_{hash}[f(x), f(y)]}{Sa\{(x,y)|D_{hash}(x,y) = 1\}} \times \frac{SaM^2}{\sum_{x,y} D_{hash}(x,y)}$$

According to Ref[31], Sa is the number of total samples, a_{ij} is element of $n \times (m+1)$ distance matrix which indicates the number of vector change in *j*-th bit output when the *i*-th bit input changes, b_{ij} is element of $n \times m$ matrix which also indicates the number of vector change in *j*-th bit output when the *i*-th bit input changes, and $D_{hash} = \sum_{i=1}^{\frac{n}{2}} |dec(c_1) - dec(c_2)|$.

Among them, it can be estimated whether each hash output depends on any bit of the input, once a bit of input is flipped, each bit of output has a 50% probability of flipping. It is used to ensure that every bit flipped randomly, and every bit of output is independent of each other.

Table 6 Avalanche effect test

Hash funcs	Step of $D_I = 1$	Step of $D_E \geq 0.999$	Step of $D_S \geq 0.99$	Step of $D_F = 1$
MD5	30	31	33	12
SHA1	22	26	24	10
SHA2-256	23	24	22	10
Ref [33]	27	26	22	11
Ref [42]	8	11	10	7
Ref [38]	16	14	12	12
Ref [24]	12	17	13	11
Ref [30]	3	2	7	5
PCMCH	3	2	7	5

Avalanche comparisons of some popular hash functions are listed in **Table 6**. Each hash function can gradually approach or reach the theoretical optimal value after several rounds of iteration. According to **Fig.8** and **Table 6**, PCMCH can achieve

theoretical optimal value faster than other algorithms, so the algorithm has higher avalanche stability than the existing popular hash functions.

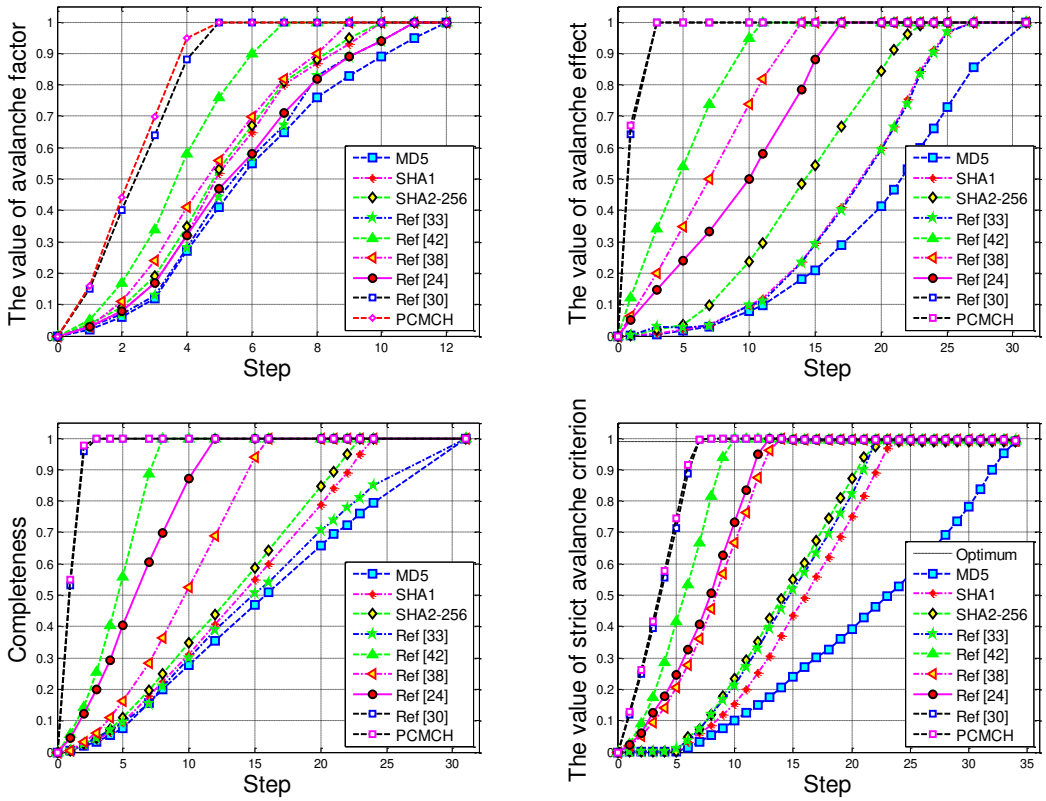

Fig. 8 Comparison of four key avalanche performance parameters

3.6 Collision resistance

For secondary collision attacks, the necessity to defend against secondary collision attacks is greater than that for collision attacks. In light of this, when a hash function is designed to resist collision attacks, it is also fortified against secondary collision attacks, as a higher level of security must be implemented to weaken its susceptibility to attacks.

For differential cryptanalysis, taking SHA1 as an example[8], the step function of SHA1 in the second round can be represented as:

$$A_{j+1} = ROTL_5(A_j) + (B_j \oplus C_j \oplus D_j) + E_j + \mathrm{W}[j] + K_2$$

$$B_{j+1} = A_j$$

$$C_{j+1} = ROTL_{30}(B_j)$$

$$D_{j+1} = C_j$$

$$E_{j+1} = D_j$$

As long as A_{j+1} can be represented by a formula containing the message word $\mathrm{W}[j]$, the differential can always be eliminated successively using the modular differential method[46].

$$\mathrm{W}[j] = A_{j+1} - ROTL_5(A_j) - (B_j \oplus C_j \oplus D_j) - E_j - K_2$$

In this formula, the differential on the right side can be inferred from the last chaining variable, and the attacker can modify the message word $\mathrm{W}[j]$ to eliminate the differential step by step. According to **Table 7**, the differential of step j can be eliminated in step $j + 6$.

Table 7 local collision

Step	A	B	C	D	E	$\mathrm{W}[j]$
j	0	0	0	0	0	1
$j+1$	1	0	0	0	0	6
$j+2$	0	1	0	0	0	1
$j+3$	0	0	31	0	0	31
$j+4$	0	0	0	31	0	31
$j+5$	0	0	0	0	31	31
$j+6$	0	0	0	0	0	0

The number in **Table 7** indicates the location where the attackers modify $\mathrm{W}[j]$. Wang successfully found a local collision and consequently obtained a collision with SHA1 with less time complexity than a birthday attack[12-13].

In this study, the main step function of PCMCH can be represented as:

$$A \leftarrow \mathrm{Maj}(A, B, C) + D + ROTL_7\big(ROTL_{12}(A) + E + ROTL_k(K_k)\big) + ROTL_{12}(A) + MW_{k+68}$$

For message word MW_i $(i = 16,17,\cdots,131)$, the message is expanded as follows:

$$MW_i = \begin{cases} \Sigma_0\big(MW_{i-16} \oplus ROTL_5(MW_{i-3}) \oplus ROTL_1(MW_{i-9})\big) \oplus ROTL_{29}(MW_{i-4}), i = 16,17,\cdots,41 \\ \Sigma_1\big(MW_{i-16} \oplus ROTL_5(MW_{i-3}) \oplus ROTL_1(MW_{i-9})\big) \oplus ROTL_{29}(MW_{i-4}), i = 42,43,\cdots,67 \\ MW_{i-68} \oplus MW_{i-64}, i = 68,69,\cdots,131 \end{cases}$$

It is extremely difficult to represent MW_i by a formula. 5 and 29 (circular left-shift bits) were relatively primed at 64. Under this condition, it becomes more difficult to eliminate the modular differential using a modular differential attack.

Collision attacks, a prevalent method of assaulting hash functions, can compromise transaction data in a variety of crucial applications, such as transactions and payments, resulting in significant financial losses.

To assess the collision resistance of PCMCH, we first define a variable denoted as N_h, which is utilized to count the number of hexadecimal character pairs of two hash values at the same position. For instance, consider two different hash values, such as "1d e9 f1 e3 de 46 67 80 86 8f 0d c8 01 6c c0 18" and "1b 3b f1 7c de 74 67 1a fa 39 51 e9 e8 6c d7 4c." If these two hash values have four identical hexadecimal pairs, then they can be recorded as $N_h = N_h + 4$.

During the test, if *x* message pairs are selected and *n* is the length of the output hash value, theoretically, the value of N_h should approach infinity.

$$N_h \approx x \times \frac{1}{256} \times \frac{n}{8} = \frac{nx}{2048}$$

To substantiate this claim, **Table 8** presents the outcomes stemming from an exhaustive replication of one million trials conducted on prevalent hash functions, alongside PCMCH. The empirical findings underscore that PCMCH demonstrates a remarkable actual variation of less than 0.014%, outperforming the benchmarked performance of all other widely employed hash functions enumerated within the table, thereby validating its superiority.

Table 8 anti-collision experimental test results of PCMCH

Name	*n*	Theoretical optimum values of N_h	Number of equal hexadecimal pair N_{eq}								Actual values of N_h
			0	1	2	3	4	5	6	≥ 7	
MD5	128	62500	938807	59421	1738	34	0	0	0	0	62999
SHA1	160	78125	924690	72529	2715	65	1	0	0	0	78158
SHA2	256	109375	896187	98415	5212	181	5	0	0	0	109402

PCMCH	512	250000	778450	195344	24128	1952	120	6	0	0	249966

3.7 Efficiency

The conventional serial architecture of hash functions encounters inefficiencies stemming from its reliance on the outcome of the preceding message block for the hash iteration calculation of each subsequent block. As the size of the message increases, the operating efficiency of this structure cannot be compared to that of a parallel structure.

Parallel hash functions are evaluated for time consumption through theoretical analysis, as shown in **Table 9**. The iterative process of hash functions involves three operations: multiplication(costs T_{mul}), addition(costs T_{add}), and shifting(costs T_{sh}). Theoretically, for a main frequency of 2.5GHz, the per time period T is approximately 0.4ns. Additive and shifting operations account for approximately one time cycle(T), whereas multiplication operations account for approximately 10 time cycles(10T). **Table 9** compares the operation numbers of several popular parallel-hash functions.

Table 9 Number of arithmetic operations of the benchmark parallel hash functions

Ref [32] (64 rounds)	Ref [35] (80 rounds)	Ref [38] (80 rounds)	Ref [30] (1 round)	PCMCH (64 rounds)
$(9T_{mul} + 15T_{add} + 1T_{sh}) \times 64$	$(5\,T_{mul} + 11T_{add} + 6.5T_{sh}) \times 80 + 1T_{mul}$	$(4\,T_{mul} + 17\,T_{add} + 9\,T_{sh}) \times 80 + 1T_{mul}$	$(34T_{mul} + 67T_{add} + 11T_{sh}) \times 8 \times 1$	$(2T_{mul} + 11T_{add} + 7T_{sh}) \times 64$

The time expenditure of Refs [32, 35, 38, and 30] and PCMCH are roughly 6784T, 5410T, 5290T, 3344T, and 2432T, respectively, as depicted in **Table 9**. Efficiency evaluations were conducted on files of different sizes, and the results are illustrated in **Fig.9**.

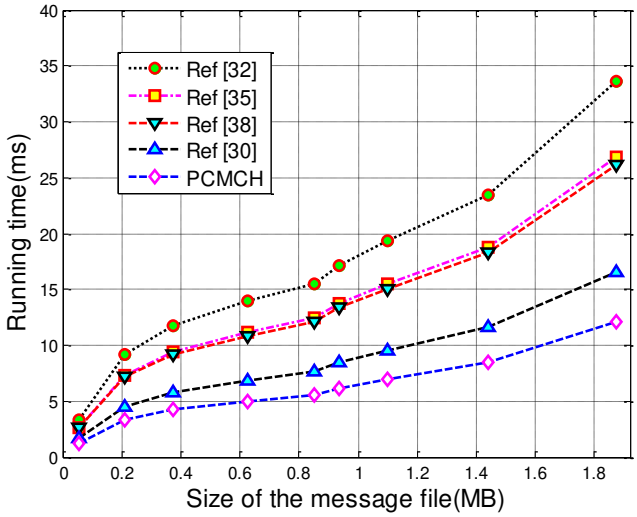

Fig.9 Comparison of the efficiency of several popular hash functions

PCMCH exhibits specifications that are comparable to those of other parallel hash functions prevalent in modern processors, while demonstrating exceptional performance when harnessed in parallel execution modes. In resource-constrained environments, PCMCH necessitates a moderate amount of code size and memory consumption. As the volume of data being hashed escalates, the operational efficiency of PCMCH and other parallel hash functions undergoes a notable enhancement, with their processing time scaling almost logarithmically. This advantage becomes particularly pronounced when tasked with hashing larger data files, where PCMCH demonstrates a remarkable ability to handle increased workloads with minimal degradation in performance.

4 Conclusion

This paper introduces an innovative parallel hash function, meticulously crafted to leverage a parallel iterative hashing architecture alongside a sophisticated message diffusion compression function, seamlessly integrated with multi-threading technology. This innovative approach aims to elevate both the security posture and operational efficiency of hashing mechanisms. At the core of this proposal lies the PCMCH parallel compression framework, which revolutionizes the way multiple message blocks are processed—concurrently, through an intricate blend of iterative and obfuscated compression techniques. By meticulously tuning the parameters of

this compression architecture, we achieve a marked enhancement in the rate of message diffusion, significantly accelerating the hashing process. Extensive experimental evaluations underscore PCMCH's remarkable sensitivity to even the slightest alterations in the input message, ensuring that the distribution of generated hash values mirrors an ideal uniform distribution. When benchmarked against prevailing hash functions, PCMCH distinguishes itself with superior performance across critical metrics such as statistical analysis resilience, information entropy, avalanche effect, collision resistance, and efficiency. These advancements not only mitigate the inefficiencies inherent in traditional serial hash functions but also address the vulnerabilities often associated with parallel hashing approaches. Consequently, PCMCH emerges as a versatile solution with immense potential for deployment across diverse applications, encompassing blockchains for enhanced transaction integrity, digital signatures for robust authentication, cloud auditing for seamless data verification, and symmetric encryption for secure data transmission.

Funding: This study was funded by Post-doctoral Later-stage Foundation Project of Shenzhen Polytechnic under Grant 6020271005K, 6020271002K, the National Natural Science Foundation of China under Grant 62102268, the Stable Supporting Program for Universities of Shenzhen under Grant 20220812102547001, Shenzhen Science and Technology Program, China (No. JCYJ20210324100813034) and the Research Foundation of Shenzhen Polytechnic University under Grants 6024310001K, 6024310002K, 6024310010K, 6024310040K, 6019310010K, 6022312044K, 6022310031K, 6024310045K, 6023240118K and 6023310030K.

CRediT authorship contribution statement:

Yijun Yang contributed to the algorithm design and performed the experiments. **Xiaohu Yan** helped design the hash function and he is a director of this work. **Linlin Wang** contributed to the study conception and design. **Meileng Yuan** contributed to data collection, **Bin Li** contributed to the first draft of the manuscript, **Zhuolin Zhong** contributed to data analysis of the manuscript. All authors read and approved the final

manuscript.

Declaration of competing interest:

The authors declared no potential conflicts of interest with respect to the research, author- ship, and/or publication of this article.

Data availability: Data that has been used is confidential.

References

1. Rivest R. L. The MD4 Message Digest Algorithm. Lecture Notes in Computer Science: volume 537 Advances in Cryptology - CRYPTO ’90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Springer: 303-311(1990)
2. Boer. B. D., Bosselaers A. An Attack on the Last Two Rounds of MD4. Lecture Notes in Computer Science: volume 576 Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Springer: 194-203(1991)
3. Dobbertin H. Cryptanalysis of MD4. Fast Software Encryption, LNCS 1039, D. , Springer-Verlag, (1996)
4. Rivest R. L. The MD5 Message-Digest Algorithm. RFC, 1321:1-21, https://www.rfc-editor.org/info/rfc1321 (1992)
5. Dobbertin H. Cryptanalysis of MD5 compress. Presented at the rump session of Eurocrypt 1996 (1996)
6. Dobbertin H. RIPEMD-160: A Strengthened Version of RIPEMD. Lecture Notes in Computer Science: volume 1039 Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings. Springer: 71-82（1996）
7. NIST. Secure Hash Standard. Federal Information Processing Standard. FIPS-180, May, 1993(1993)
8. NIST. Secure Hash Standard. Federal Information Processing Standard. FIPS-180-1, April, 1995(1995)
9. NIST. Secure Hash Standard. Federal Information Processing Standard. FIPS-180-2, August, 2002(2002)
10. Wang X., Feng D., Lai X., Yu H. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004,199 (2004)
11. Wang X., Yu H. How to Break MD5 and Other Hash Functions. Lecture Notes in Computer Science: volume 3494 Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Springer, 2005: 19-35(2005).

12. Wang X., Yu H. Efficient Collision Search Attacks on SHA-0. Lecture Notes in Computer Science: volume 3621 Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Springer, 2005: 1-16（2005）
13. Wang X., Yin Y, Yu H. Finding Collisions in the Full SHA-1. Lecture Notes in Computer Science: volume 3621 Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, 202 Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Springer, 2005: 17-36(2005)
14. Liang J., Lai X. Improved collision attack on hash function MD5. IACR Cryptol. ePrint Arch. 2005:425 (2005)
15. Sasaki Y., Naito Y., Kunihiro N., Ohta K. Improved collision attacks on MD4 and MD5. IEICE Trans. 90-A(1), 37-47(2007)
16. Stevens M. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In: Advances in Cryptology-Eurocrypt 2013, Lecture Notes in Computer Science, 7881, 245-261 (2013)
17. Li S. Y., Zhang Y., Chen K. Cryptoanalysis of an Authenticated Data Structure Scheme With Public Privacy-Preserving Auditing. IEEE Transactions on Information Forensics and Security. 16, 2564-2565 (2021)
18. Li W., Gao Z., Gu D. Security Analysis of Whirlpool Hash Function in the cloud of Things. KSII Transactions on Internet and Information Systems. 11(1), 536-551 (2017)
19. Merkel R. One way hash functions and DES. Advances in Cryptology CRYPTO 89. Lecture Notes in Computer Science, 435, 428-446 (1990)
20. Lucks S. A failure-friendly design principle for hash functions. Asiacrypt 2005, LNCS 3788, 474-494 (2005)
21. Biham E., Dunkelman O. A framework for iterative hash functions –HAIFA. Cryptology ePrint Archive: Report 2007, 278(2007)
22. Bertoni G., Daeman J., Peeters M. Sponge functions. ECRYPT Hash Workshop 2007. https://www.researchgate.net/profile/Gm-Bertoni/publication/242285874_Sponge_Functions/links/53db850d0cf2a76fb6679fd5/Sponge-Functions.pdf(2007)
23. Khushboo B., Dhananjoy D. MGR Hash Functions. Cryptologia, 43(5), 372-390 (2019)
24. Yang Y. J., Chen F., Zhang X. M., Yu J. P., Zhang P. Research on the Hash Function Structures and its Application. Wireless Personal Communications, 94(4), 2969-2985 (2017)
25. Liu H. J., Wang X. Y., Kadir A. Constructing chaos-based hash function via parallel impulse perturbation. Soft Computing, 25(16), 11077-11086(2021)

26. Li D., Ding P. P., Zhou Y. Q., Yang Y. G. Controlled alternate quantum walk-based block hash function. Quantum Information Processing, 22(10), 1-13 (2023)
27. Karthik,P., Bala, P. S. A new design paradigm for provably secure keyless hash function with subsets and two variables polynomial function. Journal of King Saud University – Computer and Information Sciences, 34(5), 1933-1949 (2019)
28. Ayubi P., Setayeshi Sa., Rahmani A. M. Chaotic Complex Hashing: A simple chaotic keyed hash function based on complex quadratic map. Chaos Solitons & Fractals, 34(5), DOI: 10.1016/j.chaos.2023.113647 (2023)
29. Guo C., Iwata T., Minematsu K. New indifferentiability security proof of MDPH hash function. IET Information Security, 16(4), 262-281 (2022)
30. Yang Y. J., Tian X., Pei P., He X. H., Zhang X. Y. Novel cryptographic hash function based on multiple compressive parallel structures. Soft Computing, 26(24): 13233-13248(2022)
31. Yang Y. J., Zhang X. Y. A Novel Hash Function Based on Multi-iterative Parallel Structure. Wireless Personal Communications, 127(4), 2979-2996 (2022)
32. Je S. T., Azman S., Amir A. Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dynamics, 81, 1067-1079 (2015)
33. Nouri M., Safarinia M., Pourmahdi P. The Parallel One-way Hash Function Based on Chebyshev-Halley Methods with Variable Parameter. International Journal of Computers Communications & Control, 9(1), 24-36(2014)
34. Meysam A., Shahram J., Narjes N. A novel keyed parallel hashing scheme based on a new chaotic system. Chaos, Solitions and Fractals, 87, 216-225 (2016)
35. Wang Y., Wong K. W., Xiao D. Parallel hash function construction based on coupled map lattices. Communications in nonlinear science and numerical simulation, 16(7), 2810-2821 (2011)
36. Kevin A., Robert R. Optimization of Tree Modes for Parallel Hash Functions: A Case Study. IEEE Transactions on Computers, 66(9), 1585-1598 (2017)
37. Salvatore P., Pedro R., Juan A. M. Parallel d-Pipeline: A Cuckoo hashing implementation for increased throughput. IEEE Transactions on Computers, 65(1), 326-331 (2016)
38. Yang Y. J., Chen F., Sun Z. W., Wang S. L., Chen J. Y. Secure and efficient parallel hash function construction and its application on cloud audit. Soft Computing, 23(18), 8907-8925 (2019)

39. Ye G., Zhao H., Chai H. Chaotic image encryption algorithm using wave-line permutation and block diffusion. Nonlinear Dynamics, 83, 2067-2077 (2016)
40. Guesmi R., Farah M., Kachouri A. A novel chaos-based image encryption using DNA sequence operation and Secure Hash Algorithm SHA-2. Nonlinear Dynamics, 83, 1123-1136 (2016)
41. Kanso A., Ghebleh M. A structure-based chaotic hashing scheme. Nonlinear Dynamics, 81, 27-40 (2015)
42. Yang Y. J., Chen F., Chen J. Y., Zhang Y., Yung K. L. A secure hash function based on feedback iterative structure. Enterprise Information Systems, 13(3), 281-302 (2019)
43. Karthik P., Shanthi P. A New Design Paradigm for Provably Secure Keyless Hash Function with Subsets and Two Variables Polynomial Function. Journal of King Saud University - Computer and Information Sciences, 34(5), 1933-1949 (2022)
44. Zhang Y., He Z., Wan M. A New Message Expansion Structure for Full Pipeline SHA-2. IEEE Transactions on Circuits and Systems, 68(4), 1553-1566 (2021)
45. Lee J., Hong D. Collision resistance of the JH hash function. IEEE Transactions of Information Theory, 58(3), 1992-1995(2012)
46. Yang Y. J., Yu J. P., Zhang Q., Meng F. Y. Improved Hash Functions for Cancelable Fingerprint Encryption Schemes. Wireless Personal Communications, 84, 643-669 (2015)