International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Novel Multi-Level Hashing Algorithm to Enhance Internet of Things Devices' and Networks' Security

Teba Mohammed Ghazi Sami[1], Subhi R. M. Zeebaree[2], Sarkar Hasan Ahmed[3]

Submitted: 25/06/2023 **Revised**: 07/08/2023 **Accepted**: 29/08/2023

Abstract: The Internet of Things (IoT) has emerged as a significant technological advancement in recent years. Its development has garnered global attention from many organisations, industries and researchers. The IoT is responsible for collecting and processing data from remote areas, significantly enhancing productivity for distributed systems and individuals. Secure hash algorithms (SHAs) are critical to ensuring enhanced security levels within IoT ecosystems. These algorithms generate fixed-size hash values from input data, making them useful for various security applications. This paper proposes a new multi-level hashing algorithm (MLHA) that could overcome challenges and significantly impact IoT security. The proposed MLHA was carefully considered and managed to effectively enhance IoT security without compromising the performance and efficiency of IoT devices. The new hashing algorithm is suitable for all IoT devices, ranging from small devices with limited processing power fuelled by batteries to larger devices with constant electrical power and vast resources. This study's main objective is to develop a new hashing algorithm specifically designed for IoT devices. This research strives to enhance IoT devices' data protection and security measures using improved hashing algorithms. This objective will be achieved by examining various IoT devices, analysing current algorithms and developing strategies that effectively balance efficiency and security. Furthermore, the suggested algorithm presents a novel methodology for enhancing data security in IoT devices. The algorithms examined in this paper are from the SHA family, which are based on bitwise operations. However, these algorithms were modified to enhance their efficiency and scalability for IoT devices. The proposed algorithm comprises eight levels, each accommodating a particular IoT device. The first level is characterised by its basic nature and is specifically tailored for devices with limited resources, including low RAM and CPU and battery capacity. As the process of iterative evolution progresses, each level demonstrates adaptability to the capabilities of various IoT devices. Moreover, the algorithm progresses towards its higher levels, which are characterised by complex equations, functions, output lengths, more words, increased arithmetic and logical operations and iterations. These advanced levels of the proposed algorithm were designed to cater to larger, more complex IoT devices than the initial-stage devices.

Index Terms: *Hashing algorithm, Internet of Things (IoT), power consumption, security, secure hash algorithm (SHA) family.*

1. Introduction

The Industrial Internet of Things (IIoT) is a specialised subset of the broader Internet of Things (IoT) concept. It involves the integration of advanced sensors, devices, machines and software applications within industrial environments to enhance operational efficiency, optimise processes and enable data-driven decision-making. IIoT focuses on digitising and connecting the various elements of industries such as manufacturing, energy, transportation and agriculture. It aims to leverage the power of connectivity and data analysis to transform traditional industries into more interconnected, automated and intelligent ecosystems.

Computer Science Dept, Faculty of Science, University of Zakho, Duhok, Iraq
teba.sami@uoz.edu.krd
Energy Eng. Dept, Technical College of Engineering, Duhok Polytechnic University, Duhok, Iraq
subhi.rafeeq@dpu.edu.krd
Network Dept, Technical College of Engineering and Informatics, Sulaimani Polytechnic University, Sulaimani, Iraq
sarkar.ahmed@spu.edu.iq

Key characteristics and components of IIoT include the following:

- Connected devices: IIoT involves deploying sensors, actuators and other smart devices across industrial settings to collect real-time data. These devices can range from machinery in a factory to equipment in a power plant.
- Data collection and analysis: The data collected from these devices is sent to centralised platforms or cloud-based systems for analysis. Advanced analytics tools are used to process the data and derive meaningful insights.
- Automation: IIoT enables automation of various processes, reducing the need for manual intervention and streamlining operations. This can lead to increased efficiency, reduced errors and faster response times.
- Remote monitoring and control: Industrial processes and equipment can be monitored and controlled remotely through IIoT. This is particularly valuable

for scenarios where physical access may be challenging or dangerous.

- Predictive maintenance: IIoT enables predictive maintenance by continuously monitoring equipment conditions. It can predict when maintenance is required by analysing data patterns, minimising downtime and optimising maintenance schedules.
- Energy efficiency: IIoT can optimise energy consumption by monitoring energy usage patterns and suggesting waste minimisation strategies.
- Supply chain optimisation: IIoT can be applied to supply chain management. It can be used for tracking the movement and condition of goods throughout the supply chain, enhancing visibility and efficiency.
- Security and privacy: Due to the sensitive nature of industrial operations, IIoT strongly emphasises security measures to protect data, devices and systems from cyber threats.
- Interoperability: As different devices and systems are often sourced from various manufacturers, ensuring interoperability and seamless communication between these components is a critical aspect of IIoT.
- Real-time decision-making: IIoT provides real-time insights that enable quick and informed decision-making, helping industries respond promptly to changing conditions.
- Overall, IIoT has the potential to revolutionise industries by driving innovation, optimising processes and increasing productivity. It introduces a new level of connectivity and intelligence to industrial environments, enabling organisations to achieve greater operational efficiency and competitiveness [1][2][3][4].
- Real-time data access is a crucial aspect of IoT systems, particularly in cloud-based environments where data from various connected devices is collected, processed and analysed.
- Securing real-time data access involves several key considerations:
- Low-latency communication: Low-latency communication protocols are essential for securing real-time data access. For example, Message Queuing Telemetry Transport and Constrained Application Protocol are protocols commonly used in IoT environments for their simple and efficient messaging capabilities.
- Data ingestion and processing: Cloud platforms designed for IoT often include data ingestion pipelines that can process and store incoming data streams in real-time. These platforms can handle the large volumes of data that IoT devices and sensors generate.
- Scalability: The ability to quickly scale resources up or down based on demand is essential to accommodate varying data loads while maintaining real-time responsiveness.
- Edge computing: Edge computing can reduce latency and enhance real-time capabilities. This involves processing data closer to the source (i.e. at the network’s edge) before sending relevant information to the cloud for further analysis.
- Push mechanisms: Cloud-based IoT platforms often use push mechanisms to deliver real-time data updates to applications or dashboards. WebSockets or server-sent events (SSE) are technologies that enable servers to relay data to connected clients without the need for frequent polling.
- Event-driven architecture: Designing cloud-based IoT applications with event-driven architecture enables actions to be triggered in real-time based on specific events or conditions detected in the data stream.
- Streaming analytics: Implementing streaming analytics tools enables real-time data processing, enabling immediate insights to be obtained and decisions to be made based on the analysed information.
- Caching and in-memory databases: Utilising in-memory databases or caching mechanisms can improve data retrieval speed, especially for frequently accessed or time-sensitive information.
- Load balancing: Load balancing across distributed cloud resources ensures that data processing and retrieval tasks are evenly distributed. This prevents bottlenecks that could impact real-time access.
- Monitoring and alerts: Real-time monitoring tools can provide insights into the performance and health of the IoT system, allowing administrators to proactively address issues that might impact data access.
- Security: While ensuring real-time access, it is essential to maintain robust security measures that protect the data and systems from potential breaches or unauthorised access.
- Quality of service: Prioritising certain data streams over others based on importance or urgency can help

to ensure that critical information is accessed and processed in real-time.

In summary, achieving real-time data access in a cloud-based IoT environment involves a combination of efficient communication protocols, data processing pipelines, edge computing, event-driven architectures and scalable infrastructure. By optimising these aspects, organisations can harness the power of real-time data to obtain timely insights and make informed decisions. The IoT industry is experiencing rapid expansion. This is primarily due to cost-effective manufacturing and the wide range of available IoT devices. Additionally, the widespread adoption of cloud technology increases the potential of IoT applications by harnessing the combined computational power of the cloud and the physical capabilities of IoT systems. According to projections made by previous statistics, the number of IoT devices is expected to rise to 73 billion by 2025 [5][6][7].

IoT security is paramount due to the interconnected nature of the devices and the sensitive data they manage. Consequently, the impact of emerging technologies on IoT security introduces challenges and potential solutions. A comprehensive overview is presented below [11][12][13][14][15][16][17].

Current IoT security challenges:

- Device vulnerabilities: Many IoT devices have limited computational resources, leading to vulnerabilities that attackers can exploit.
- Data privacy: IoT devices often collect and transmit sensitive personal data. Therefore, inadequate data protection can lead to privacy violations.
- Lack of standardisation: IoT lacks standardised security measures. This results in inconsistent implementation across devices and platforms.
- Interoperability issues: Different devices and platforms may struggle to communicate securely due to compatibility and interoperability issues.
- Inadequate updates: Manufacturers often neglect to provide timely security updates and patches for IoT devices, leaving them vulnerable.
- The impact of emerging technologies: 5G connectivity offers higher speeds and lower latency, which can improve security by enabling real-time threat detection and responses.
- Edge computing: Edge computing can reduce data transmission to central servers. This improves security by limiting exposure to potential attacks during data transfer.
- Blockchain: The blockchain's dispersed and immutable nature can enhance IoT security by providing secure data storage and tamper-proof audit trails.
- Artificial intelligence (AI) and machine learning: These technologies can detect anomalies in IoT device behaviour. This enables the identification of potential security breaches in real-time.
- Secure hardware: Developing secure hardware components can provide a foundation for building more secure IoT devices.
- Post-quantum cryptography: As quantum computing advances, post-quantum cryptography will be crucial for maintaining encryption integrity.
- Various security measures and solutions are discussed below:
- End-to-end encryption: Encrypting data at rest and during transmission ensures that the data remains secure even if it is intercepted.
- Device identity management: Strong authentication and identity management can prevent unauthorised devices from accessing the network.
- Security updates: Regularly updating and patching IoT devices can help mitigate known vulnerabilities.
- Security by design: Integrating security measures into the design and development process is fundamental to building resilient IoT devices.
- Network segmentation: Isolating IoT devices from critical systems through network segmentation reduces the potential impact of a breach.
- Intrusion detection systems (IDSs): An IDS monitors network traffic, identifying suspicious activities and unauthorised access attempts.
- Behavioural analytics: Monitoring the typical behaviour patterns of IoT devices enables quick detection of anomalies that are indicative of a security breach.
- Regulatory compliance: Complying with regulations such as GDPR ensures the data is protected and privacy rights are maintained.
- Future directions: The continued evolution of emerging technologies will challenge and augment IoT security. Combining the capabilities of secure hardware, AI, blockchains, and edge computing can create a comprehensive security framework. In addition, industry collaboration, regulatory advancements and education on secure IoT

practices will be vital for mitigating security risks. In conclusion, securing IoT devices amidst the impact of emerging technologies is a dynamic endeavour. Therefore, addressing existing challenges while leveraging the benefits of emerging technologies to create a safer and more resilient IoT ecosystem will require continuous research, development and collaborative efforts among stakeholders [8][9][10].

- Furthermore, developing a new collision-resistant hashing algorithm is a complex and specialised task that requires a deep understanding of cryptography, security principles and algorithm design. A high-level outline of the steps to consider when embarking on such a project is described below:
- Define objectives and requirements: Clearly outline the goals of your new hashing algorithm by determining the security level required, the anticipated use cases and any specific features or properties the algorithm should possess.
- Research existing algorithms: Study the existing collision-resistant hashing algorithms such as SHA-256, SHA-3 and BLAKE2. Understanding their strengths, weaknesses and design choices will provide valuable insights into best practices and potential areas for innovation.
- Threat model and analysis: Identifying potential threats and attack vectors that your algorithm should be resistant to will help you design one that addresses specific security concerns.
- Algorithm design: Start designing the algorithm based on your research and objectives and consider elements such as compression and round function. The compression function is the core of the hashing algorithm that takes input data and produces a fixed-size output. Regarding the round function, if your algorithm employs multiple rounds, design an efficient and secure round function.
- Initialisation and finalisation: Determine how the algorithm will process the input data at the beginning and end of the hashing process.
- Nonce or salt: Consider including a nonce or salt for added security against precomputed attacks.
- Security properties: Focus on achieving collision, preimage and second preimage resistance. Mathematical proofs or analysis will be necessary to demonstrate the algorithm's security properties.
- Cryptanalysis and testing: Subject your algorithm to rigorous cryptanalysis by simulating attacks, evaluating known attack resistance and soliciting feedback from the cryptographic community. This step is crucial for identifying vulnerabilities and refining the algorithm's design.
- Implementation and testing: Implement the algorithm in a programming language of your choice, create comprehensive test cases and perform extensive testing to ensure accuracy, performance and security.
- Peer review and feedback: Share your algorithm design and implementation with the cryptographic community and request peer review, feedback and constructive criticism. Engaging with experts can help uncover potential weaknesses and provide valuable insights.
- Adjust and refine: Refine the algorithm based on the feedback received during the review process. Iteratively update the design to address concerns and improve security.
- Documentation and standardisation: Provide clear documentation that outlines the algorithm's design, rationale, security analysis and usage guidelines. If your algorithm gains traction, consider submitting it for standardisation from the relevant cryptographic organisations.
- Real-world testing: Implement your algorithm using real-world scenarios and applications and monitor its performance and security over time. Then, address any unforeseen issues that arise.
- Continuous research: Cryptography is an evolving field. Therefore, stay updated on new research, vulnerabilities and cryptographic advances. Be prepared to adapt your algorithm to new findings. It is important to remember that designing a new cryptographic algorithm is a complex endeavour with significant security implications. In addition, engage with experts in the field, adhere to best practices and prioritise security at every step of the process.
- Over time, different secure hash algorithms (SHAs) variants have emerged, such as SHA-1, SHA-2, SHA-256, SHA-348 and SHA-512. This reflects the continuous evolution of secure encryption techniques, observed through research and legal authorities like National Institute of Standards and Technology (NIST). SHAs are crucial in securing IoT devices and data. They provide a way to verify data integrity, authenticate messages and protect sensitive information from unauthorised access.
- Below are some widely used SHAs for securing IoT:

- SHA-256: SHA-256 is a widely adopted cryptographic hash function. It produces a 256-bit hash value and is considered secure for various applications, including IoT. It is used for digital signatures, data integrity checks and password hashing.
- SHA-3: SHA-3 is the latest SHA family member selected through a public competition. It offers a high level of security against certain types of attacks and is designed to be more resistant to potential vulnerabilities.
- BLAKE2: BLAKE2 is a cryptographic hash function faster than most other hash functions. This makes it suitable for resource-constrained IoT devices. It provides security similar to SHA-3 but with an improved performance.
- Whirlpool: Whirlpool is a cryptographic hash function designed for enhanced security. It offers a 512-bit hash value and is suitable for applications requiring higher security levels.
- Skein: Skein is another option from the SHA-3 competition. It is a flexible hash function that can adapt to various security requirements and is designed to be efficient on various platforms.
- SHA-1: Although it was once widely used, SHA-1 is now considered risky due to its vulnerabilities. Avoiding the use of SHA-1 in security-critical applications, including IoT, is recommended.
- When selecting an SHA to secure IoT devices, consider the following factors:
- Security level: Choose an algorithm with an appropriate security level for your IoT application. Higher bit lengths generally provide stronger security but may come with performance trade-offs.
- Performance: IoT devices often have limited computational resources. Consider algorithms that balance security and performance, especially for resource-constrained devices.
- IoT suitability: Some algorithms are more suitable for IoT due to their speed, efficiency and compatibility with resource-constrained devices.
- Availability and support: Choose algorithms that are well-vetted, standardised and supported by reputable cryptographic libraries.
- Updateability: Ensure that the chosen algorithm can be updated or replaced easily to address potential future vulnerabilities.
- Industry standards: Consider following industry best practices and standards for hash algorithm selection in IoT applications. Ultimately, securing IoT involves a multi-layered approach, including proper authentication, encryption, access control, secure communication protocols and the use of SHAs.

Notably, research efforts in this field have resulted in a new version, SHA-3. However, due to its novel structure, the strengths and weaknesses of this encryption technique are still not fully understood. Various methods employ SHA variants to address diverse security requirements, including electronic document authentication, enhancing security systems using chaotic maps, secure message hiding, image security, improved anonymity, combining hashing with genetic science to strengthen security, and enhancing security through blockchain and authentication in mobile networks. The SHA family consists of cryptographic hash functions developed by the National Security Agency (NSA) in the United States (US). These hash functions ensure data integrity, message authentication, and other security applications. The evolution of the SHA family includes several iterations that reflect advancements in security requirements and cryptographic research.

An overview of the evolution and analysis of the SHA family is presented below [18][19][20]:

- SHA-0: SHA-0 was the first version of the SHA, introduced in 1993. However, it was immediately observed to have vulnerabilities, and its use was discouraged.
- SHA-1: SHA-1, introduced in 1995, improved upon the flaws of SHA-0. It produced a 160-bit hash value and was widely adopted for digital signatures and data integrity checks. However, over the years, research revealed its vulnerabilities, and in 2005, researchers demonstrated a collision attack, which led to concerns about its long-term security. As a result, the use of SHA-1 has been reduced in many security-sensitive applications.
- SHA-2: Introduced in 2001, SHA-2 addressed the security concerns of SHA-1. It comprises several hash functions with different bit lengths, including SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256. SHA-256 and SHA-512 are the most widely used variants. SHA-2 remains secure and widely adopted, with no practical collision attacks demonstrated as of September 2021.

- SHA-3: SHA-3 was introduced during a public competition organised by the NIST to select a new SHA. The competition began in 2007 and culminated in the selection of Keccak as the basis for SHA-3 in 2012. SHA-3 offers cryptographic primitives and a novel design. It is considered more resistant to certain types of attacks and is a better alternative to SHA-2. It also supports 224-, 256-, 384- and 512-bit hash lengths.

SHA analysis and considerations include the following:

- Security levels: As computational capabilities advance, older members of the SHA family may become vulnerable to attacks. Therefore, selecting the appropriate SHA family member, based on the desired security level and the specific application requirements, is essential.
- Performance: Hash functions like SHA-256 are computationally efficient and widely used. However, SHA-3 variants present certain compromises between security and performance.
- Algorithm selection: SHA-2 and SHA-3 are currently secure. When selecting the most appropriate algorithm, one should consider factors such as available resources, performance requirements and potential future advances in cryptanalysis.
- Quantum threat: As quantum computing evolves, hash functions like SHA-256 may become more vulnerable to quantum attacks. Thus, preparing for the post-quantum era involves exploring hash functions designed to be resistant to quantum attacks.
- Standardisation and compatibility: When adopting new hash functions like SHA-3, it is important to consider factors such as library support, standardisation and compatibility with existing systems. In summary, the evolution of the SHA family reflects cryptographic research advancements and the need for more robust security. Therefore, when selecting a hash algorithm, it is crucial to assess the balance between security, performance and potential vulnerabilities while monitoring emerging cryptographic trends.

A list of research that is closest to being relevant to the recommended system, as identified by this study, is presented below:

Several studies have invented new hashing algorithms in recent years, while others have focused on other aspects of known hash functions. At the same time, several studies focused on various aspects of established hash functions. In 2022, Sakan et al. [21] presented and analysed the HBC-256 method [21]. While developing their method, they utilised the Devis–Mayer structure and analysed its avalanche effect. Then, they proposed a potential solution based on using parallelisation strategies to enhance overall performance. Notably, the researchers reduced the total number of rounds to improve their work's overall effectiveness.

They also observed that due to the unique nature of the compression function, it is possible to handle a greater number of blocks concurrently than was previously thought. In 2020, Tutueva et al. [22] developed a novel strategy that successfully countered collision attacks. This method involves using adaptive chaotic maps in a hashing algorithm. Moreover, Yu et al. [23] developed an original keyed hash function released in 2020. This function uses the Merkle–Damgard architecture with a chaotic map neural network to perform its tasks. According to the authors, this cryptographic hash function demonstrates resistance to various modern approaches, including multi-collision attacks.

In contrast to classic collision attacks, these attacks on cryptographic hash functions have received less attention in academic publications. In 2020, De Silva [24] offered an approach contrasting the previous statement. De Silva's [24] architectural design used complex individual block permutations and rotations to boost its resistance against prospective attacks at the expense of the building's overall performance. Despite this, using SHA has been observed to improve the IoT system's overall security level. However, data and firmware integrity, message authenticity and password hashing are the primary areas of concern in this context. Other challenges must also be considered and resolved, such as processing overhead, resource restrictions, algorithm selection and collision resistance. Based on the previously mentioned research, it is evident that contemporary researchers are actively studying unique approaches to developing hashing algorithms capable of effectively resisting modern attacks.

2. Background Theory

A substantial part of this study's algorithm comprises the cryptographic hash functions known as SHA-2. The NSA developed these functions in the US, and they play an essential role in this algorithm. The Merkle–Damgard structure, which uses a one-way compression function, was used in designing these functions. Moreover, this compression function, in turn, uses the Davies–Meyer structure, reflected in the use of a distinct block cypher. It is essential to remember that this block cypher's details will remain undisclosed.

The SHA-2 cryptographic hash algorithm has two separate iterations, SHA-256 and SHA-512. The computations performed by the former use 32-bit words, while the latter uses 64-bit words. Their overall structures are identical, with the primary difference being the number of rounds carried out. However, their shift values and additive constants are distinct. The NSA developed the SHA, a collection of cryptographic hash functions [25]. These hash functions were designed to receive an input (i.e. a message) and produce an output of a predetermined size (i.e. the output is known as a hash value). This hash value displays unpredictable properties and demonstrates a high degree of sensitivity to any input changes. In cryptography, the primary applications of SHAs include verifying data integrity, creating digital signatures, storing passwords and authenticating messages. This article thoroughly analyses the SHA's relevance in cryptography and explains its primary purpose. During the data transmission and storage stages, SHAs are used to ensure the maintenance of data integrity [26].

Generating a hash value for a message and including it in the transmission enables the receiver to independently calculate the hash and determine the received message's integrity, ensuring that it has not been subjected to any unauthorised modifications. On the one hand, when the hash values are identical, deducing that the data has not been changed is possible. On the other hand, if the hash values do not align, this could mean that the file has been illegally altered. Using SHAs is highly crucial to produce and certify digital signatures.

Hashing is a cryptographic method that generates a unique digital representation of a message. The sender uses this method to enable the recipient to verify the message's authenticity. The resulting hash value is cyphered using the sender's private key. The receiver may decrypt the encrypted hash using the public key that the sender has provided, and it can then be compared to the hash that the recipient has calculated based on the message they received. If the digital signatures are identical, it is reasonable to assume that the document is genuine. Combining SHAs and various salting techniques ensures that user passwords are kept in databases in a secure manner. When a user creates a new password or changes an existing one, the password goes through a process of hashing, using the SHA. Moreover, whenever a user logs in, the system uses a hashing technique to convert the password into a value that only the system knows. The resulting hash value is then compared to the previously saved one. The user is then granted access if the two hash values are similar.

Message authentication codes (often known as MACs) are created using SHAs to ensure that communications are legitimate and have not been altered. A message authentication code is a cryptographic tag attached to a message to check its integrity and validity during the transmission process. This ensures that the message has not been illegally altered or forged. Applications related to cryptography, such as SHAs, are often employed in various cryptographic protocols, including key derivation functions and pseudorandom number generators.

The following section highlights the most commonly used SHAs. SHA-1, which generates a 160-bit long hash, is the first component in the SHA family of algorithms. Its current strength has been estimated to be lower than before due to its flaws. SHA-2 is a collection of hash functions encompassing various bit lengths, such as SHA-224, SHA-256, SHA-384 and SHA-512. The SHA-256 and SHA-512 cryptographic hash algorithms are widely employed and are often considered to have strong security levels due to their size and complexity. In addition, the Keccak algorithm provides the foundation for the SHA-3 hash function family, created during a public competition. This security solution offers an alternative technique and is often recognised as one of the most secure options.

Consequently, selecting an appropriate hash algorithm is extremely important because it requires careful evaluation of the necessary security criteria, performance aspects and compatibility with existing systems. In recent years, there has been a discernible shift towards using relatively more secure hash functions, such as SHA-256 and SHA-3. This shift may have resulted from the advancements made in the cryptanalysis field and an increase in the amount of accessible computer resources [27][28][29][30]. In 2017, the SHAttered prefix collision attack was successfully implemented, proving that SHA-1 can no longer be deemed collision-resistant. This assault involved 263 computations, leading industry experts to believe that the protection offered by SHA-1 against collision attacks was no longer enough. As a result, they recommend using SHA-2 or SHA-3 as an alternative.

Blockchain is a digital ledger system distinguished by its decentralised structure, which ensures the secure recording of transactions across several computers (i.e. nodes). It was developed by Bitcoin creator, Satoshi Nakamoto. The term 'decentralisation' refers to the operational blockchain structure, which uses a dispersed network to ensure that no single user has full control of the system. The participants reach a consensus in order to validate and record transactions. The transactions in a distributed ledger system are grouped into blocks and sequentially linked, resulting in the formation of a chain that cannot be altered. Using this distributed ledger ensures openness and security in all transactions. As a result, the transactions are protected against tampering and guaranteed to maintain their integrity with the use of cryptographic security. Every block in the blockchain system contains a cryptographic hash of the previous block, creating a reliable and secure hyperlink between the blocks in the blockchain. The term ‘consensus mechanisms’ refers to the processes employed to achieve agreement on the legality of transactions, followed by the incorporation of transactions into the blockchain. Examples include proof of work (PoW) and proof of stake (PoS). ‘Smart contracts’ are contractual obligations that may be automatically executed using computer codes. These contracts are designed to automatically enforce and execute the agreed-upon conditions once certain established criteria have been met.

Public blockchains, such as Bitcoin and Ethereum, are characterised by their open access to the public. In contrast, private blockchains restrict access, enabling only authorised participants to utilise the network. Blockchain technology can be implemented within several domains. Cryptocurrencies are digital or virtual currencies that function on decentralised networks and employ encryption for additional layers of protection. Bitcoin, a digital currency that runs on blockchain infrastructure, is one of the applications that has garnered the greatest public attention. This technology makes it possible for participants to conduct transactions without the intervention of a third party, removing the need for intermediaries. Moreover, using blockchain technology in logistics and supply chain management increases visibility due to its capacity to monitor the movement of goods and validate their point of origin. Its use has significantly reduced the risk of counterfeiting and fraudulent operations Blockchain technology also has uses within the financial services industry and can potentially improve international payments, remittances and trade settlements, reducing transaction times and associated costs. Additionally, voting machines are designed to create voting records that are unchangeable and indecipherable to increase the reliability and openness of electoral processes. The real estate sector can potentially improve the effectiveness of property transactions by adopting digital solutions, including digitising land titles and automating various operations, such as title transfers and issuing mortgages. Furthermore, using blockchain technology in IoT and supply chain management systems can improve traceability. This may be accomplished by easing the process of systematically capturing data at each step and verifying its authenticity.

The energy industry aims to facilitate energy trade between associates, monitor the output of renewable energy and improve the effectiveness of grid management. Therefore, placing stringent intellectual property protection measures is crucial for conserving copyrights, patents and trademarks while limiting disputes and ensuring appropriate recognition. In addition, the digital rights management initiative aims to give creators and producers of digital works greater control over the use and distribution of their creations. The goal is to increase transparency and accountability among charitable organisations so donors can trust that their contributions will be used appropriately.

These applications are only a few examples of the wide range of possible uses for blockchain technology within various industries. As blockchain technology continues to expand, its potential to introduce revolutionary shifts across various business sectors is significant and will result in improved levels of productivity, transparency and safety [31][32][33].

It is now generally accepted that SHA-2 is a hashing method that demonstrates adequate collision resistance. As of 2022, no modifications to SHA-2 have been observed to enable conducting a collision attack more quickly and efficiently than a general one. Although various theoretical collision attacks for SHA-2 with fewer rounds have been developed, the actual applicability of these attacks to the complete SHA-2 algorithm is restricted. The early concerns expressed by academics regarding SHA-2’s potential vulnerability to differential cryptanalysis attacks generated by the algorithm’s Davis–Meyer structure motivated the development of the new SHA-3 standard. In 2012, the NIST organised a competition with the goal of establishing a new standard, and the winner of that competition developed SHA-3. After being evaluated with the other options, the Keccak function was chosen as the winning contender. Then, four researchers working under the direction of Joan Dymen set out to create SHA-3, adding an innovative member to the existing SHA family of hash functions [34].

The IoT is a significant component of the modern information technology landscape. It enables the connection of various devices to the internet for data exchange and communication, achieved through a range of technologies, such as radio frequency identification, (RFID), infrared sensors, positioning systems and laser scanners, following established protocols. As a result, the IoT facilitates intelligent identification, positioning, tracking, monitoring and management of items. Figure 1 illustrates the theoretical model of IoT. The IoT is a complex and interdisciplinary field encompassing various theoretical frameworks used to understand and analyse its dimensions.

Some theoretical frameworks commonly used to study and conceptualise the IoT are presented below:

Information-centric networking (ICN): ICN focuses on data dissemination and retrieval by naming content rather than locations. In the IoT, ICN can enhance data distribution by allowing devices to request and share information without knowing the exact location of the data source.

Cyber-physical systems (CPSs): CPS theory examines the integration of computational algorithms and physical processes. The IoT devices often bridge the gap between the virtual and physical worlds. This makes CPS theory relevant to the IoT's interactions with the physical environment.

Ubiquitous computing: Ubiquitous computing emphasises the seamless integration of technology into the environment [35]. The IoT aligns with this framework by aiming to make technology pervasive and unobtrusive, integrating devices into everyday objects and spaces.

Network theory: Network theory studies the interactions between interconnected elements. In the IoT, network theory is used to analyse device connectivity patterns, data flows and communication dynamics.

Complex systems theory: Complex systems theory explores emergent behaviours that arise from the interactions among multiple components. With its multitude of interconnected devices, the IoT exhibits properties that can be understood through this framework.

Social network theory: In social contexts, social network theory can analyse the interactions between people and IoT devices, studying influence patterns and information dissemination.

Sensor networks theory: This framework focuses on deploying and operating large-scale sensor networks. It is particularly relevant to the IoT due to the vast number of sensors collecting data in various environments.

Data analytics and big data theory: The IoT generates massive amounts of data, and theories related to data analytics, machine learning, and big data play a significant role in extracting valuable insights from IoT-generated data.

The economic theory of IoT: This framework examines the economic aspects of IoT, including cost-benefit analysis, pricing models for IoT services, and the economic impact of IoT on industries and markets.

Privacy and security theories: The IoT raises concerns about privacy and security. Theoretical frameworks such as cryptography, privacy-preserving techniques and security analysis address the challenges associated with securing IoT systems.

Service-oriented computing: This framework focuses on the provision and consumption of services within a distributed environment. In IoT, devices offer and consume services, and this theory helps manage these interactions effectively.

User-centred design and human–computer interactions: When IoT devices interact with users, theories related to user experience, usability and human–computer interactions guide the design of IoT interfaces. It is important to note that IoT is a rapidly evolving field. These theoretical frameworks often overlap and intertwine, providing a comprehensive understanding of IoT's various aspects. Moreover, researchers and practitioners may draw from multiple frameworks to address IoT's multifaceted challenges and opportunities [36][37].

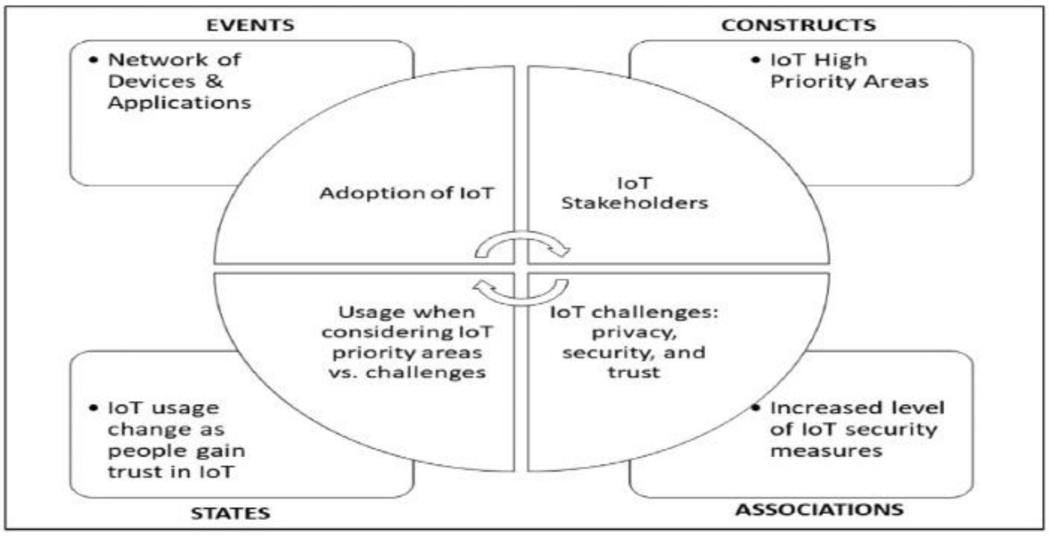

EVENTS
• Network of Devices & Applications
CONSTRUCTS
• IoT High Priority Areas
Adoption of IoT
IoT Stakeholders
Usage when considering IoT priority areas vs. challenges
IoT challenges: privacy, security, and trust
• IoT usage change as people gain trust in IoT
• Increased level of IoT security measures
STATES
ASSOCIATIONS

Fig. 1 IoT theoretical framework and conceptual model.

Fig. 1 demonstrates that human requirements are disseminated over a network under the IoT framework. Various technologies such as sensors, RFID, infrared sensors, wireless packet services, laser scanners, and other equipment are used to fulfil these requirements. Then, the outcomes of these efforts are displayed on the IoT platform.

Fig. 2 illustrates IoT's operating procedure. Perception refers to the process through which the characteristics of physical items are transformed, resulting in a cyclical sequence of data processing, transmission and control. The IoT gathers primary data from various sources. Then, the data is transformed into meaningful information, sent to terminals and integrated into the operational environment. IoT's operational core involves processing effective information according to established rules and standards and generating useable data. Subsequently, this data is disseminated to individuals through network connections. During this iterative process, individuals attain a state of centralised control and regulation over various elements and then reintegrate the acquired knowledge into the operational context [38].

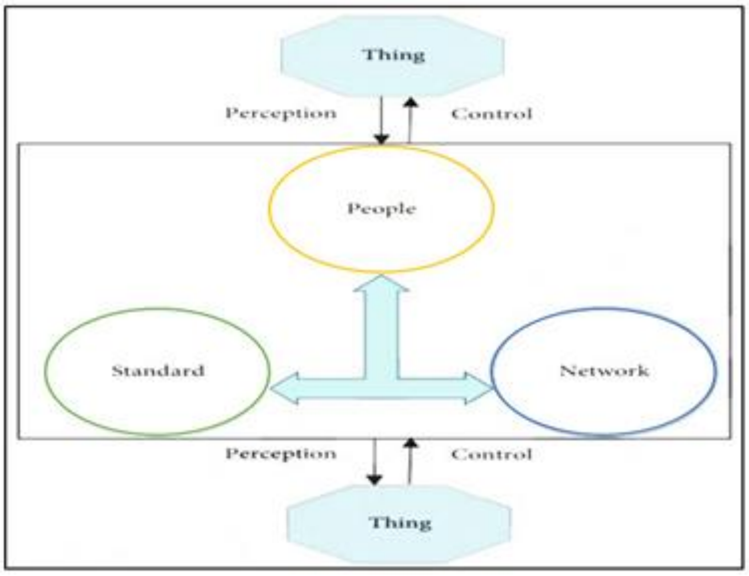


Fig. 2.The IoT operational law.

For instance, this integration can involve leveraging IoT technologies to enhance efficiency, quality and safety in infant dairy production and distribution while considering the implications of family planning policy adjustments on market demand and supply. This complex endeavour requires careful planning, collaboration and policy alignment.

Some applications for integrating IoT to enhance efficiency and quality are described below:

Smart farming: This involves utilising IoT sensors to monitor dairy farms and collect real-time data on the temperature, humidity, feed levels, and health status of the dairy animals. This enables precision farming and early disease detection, improving productivity and animal welfare.

Supply chain optimisation: This includes employing IoT devices to monitor infant dairy products' transportation, storage and distribution, ensuring optimal temperature control, minimising spoilage and improving product quality.

Quality control: IoT-enabled sensors can be deployed in dairy processing plants to monitor production parameters such as the pasteurisation temperature and product consistency, ensuring adherence to quality standards.

Regarding market dynamics and family planning, IoT can be used in the following ways:

Population trends: Altering family planning policies can influence birth rates, affecting the demand for infant dairy products. Therefore, demographic trends should be analysed, and production and marketing strategies must be adjusted accordingly.

Consumer preferences: Changes in family planning policies may impact consumer behaviours and preferences. Thus, market research must be conducted to examine the consumer's fluctuating demand patterns and help tailor product offerings.

Marketing and communication: Marketing campaigns should be aligned with family planning policies to promote the benefits of infant dairy products in the context of evolving family structures.

In terms of regulatory compliance and safety, IoT serves the following objectives:

Traceability: IoT can enhance product traceability through real-time tracking of raw materials and finished products. This is crucial for complying with food safety regulations.

Allergen control: IoT-driven monitoring can help prevent cross-contamination of allergens, ensuring that infant dairy products remain safe for consumption.

Regulatory adaptation: Staying updated with regulatory changes resulting from family planning policy adjustments can ensure that production processes and product labels remain compliant.

Concerning data security and privacy, IoT can be employed in the following ways:

IoT data protection: IoT can be used to implement robust cybersecurity measures to safeguard sensitive IoT-generated data, especially considering the privacy concerns surrounding infant-related information.

User consent: IoT can be used to ensure that individuals' data are collected with proper consent and used transparently and in compliance with data protection laws.

Additionally, IoT can facilitate a collaborative ecosystem in the following ways:

Industry collaboration: Various stakeholders, such as technology providers, dairy producers, government agencies and research institutions, can collaborate to develop and implement IoT solutions effectively.

Policy dialogue: Engaging policymakers will ensure that IoT integration aligns with family planning policies and contributes to sustainable development.

In conclusion, integrating IoT into the infant dairy industry amid family planning policy changes offers opportunities for improved efficiency, quality and market responsiveness. Stakeholders must collaborate, adapt to policy changes, prioritise data security and

align marketing strategies to cater to changing demographic dynamics. This multidimensional approach can foster a thriving infant dairy industry that is technologically advanced and socially responsible [38][39][40].

3. The New Hashing Algorithm's Methodology

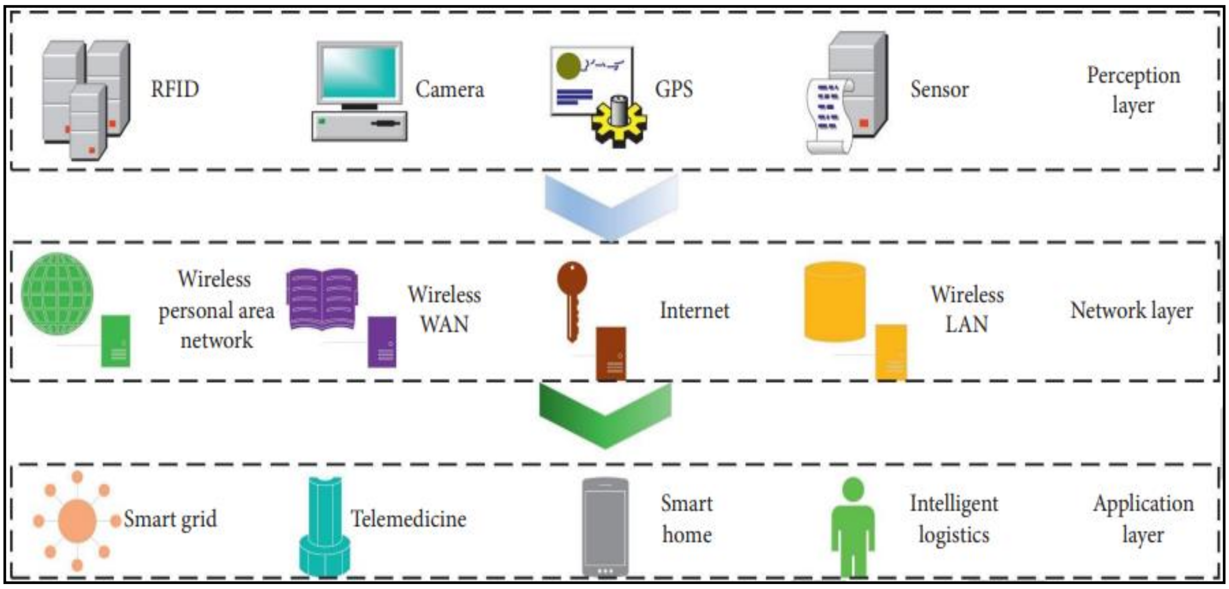


Fig. 3. IoT architecture.

IoT devices vary in size, processing capabilities and functions. The devices span a broad spectrum, ranging from small, resource-constrained gadgets with limited processing power and energy capacity to larger devices with high computing power, large memories and energy resources, often resembling supercomputers. This diversity reflects the extensive range of applications within the IoT ecosystem, where devices can serve various purposes, such as environmental sensing, healthcare monitoring and industrial automation. Developing a hashing algorithm that can effectively accommodate this broad range of devices requires carefully considering their unique constraints and demands.

This research paper focuses on creating a new hashing algorithm for IoT devices. This study aims to design an algorithm that works well across a broad range of IoT devices, since it includes small gadgets with limited power and large ones with better capabilities. By understanding the different types of IoT devices, studying existing algorithms and developing a new approach that balances efficiency and security, this research aims to improve how IoT devices handle data protection and security through enhanced hashing techniques. This study proposes an algorithm that employs a new approach to data security for IoT devices. The algorithm is derived from the SHA family based on bitwise operations but has been modified to be more efficient and scalable for IoT devices.

The proposed algorithm consists of eight levels, each designed for a specific type of IoT device. The first level is the simplest and is designed for devices with limited resources, such as RAM, CPU and battery power. This level uses a few words, has a short output length and employs simple equations and functions. The second level incorporates more words and generates longer outputs, employing more complex equations and functions. This complexity enables it to address the requirements of more advanced IoT devices with greater available resources. This iterative progression continues, with each level adapting to the capacities of different IoT devices. Eventually, the algorithm reaches its highest levels, characterised by intricate equations, functions, output lengths, more words, mathematical and logical operations and repetitions. Compared to the initial IoT gadgets, the advanced levels are designed to accommodate the demands of larger-scale and more complex IoT devices. The following table displays the properties for each proposed level.

Table 1

Properties for Each Level of the New Hashing Algorithm

Algorithm Type Characteristics	**SHAL1**	**SHAL2**	**SHAL3**	**SHAL4**	**SHAL5**	**SHAL6**	**SHAL7**	**SHAL8**
# Register	1	2	4	8	16	32	64	128
# Constant Values	1	2	4	8	16	32	64	128
# Words	20	28	32	58	68	144	170	200
Output Length (bit)	32	64	128	256	512	1024	2048	4096

One of the levels is explained in detail to clarify the creation of the new multi-level hashing algorithm (MLHA). However, the equations for each level will be presented separately to avoid the complexity of covering all eight levels in detail. For instance, the SHAL6 algorithm at the sixth level is a flow chart and pseudocode. Even though these algorithms work in similar ways, the main difference lies in their level of complexity. This complexity involves more arithmetic and logical operations, functions and iterations and increases the length of the results in each algorithm.

The first level is simple; there is only one register and constant. A straightforward arithmetic operation was used, and the output was only 32 bits long, equivalent to 8 hexadecimal numbers, as shown in Table 1. As the levels increase, gradual changes occur, increasing the robustness of the algorithm level while multiplying the bit-shuffling process. For example, in the second level, the robustness was increased using two constants and registers, and the equation was modified, allowing for longer results. This trend continues to the third, fourth and subsequent levels until the eighth. The SHAL6 algorithm will be explained in more detail in the following section. The algorithm becomes more intricate at this level than at levels lower than six. At this level, more complex arithmetic and logical operations are used, the equations and functions are more complex, and the results are longer to provide a reliable hashing algorithm based on bitwise operations. This complexity is necessary for adapting to powerful IoT devices. Therefore, it is important to strike the right balance between security and performance, as these factors are adjusted to fit different levels of computational power and security needs.

3.1 The General Statement of the SHAL6 Algorithm

The SHAL6 algorithm is a cryptographic hash function that takes an input of any length and produces a 1024-bit hash value, equivalent to 128 hexadecimal values. It is a one-way function suitable for devices with a constant power supply and tremendous computing power, including an excellent amount of RAM, CPU and storage.

SHAL6 may be used to hash a message, *M*, having a length of *l* bits, where $0 \leq l < 2^{64}$. The algorithm uses:

- A message schedule of 144 (32-bit) words.
- Thirty-two working variables of 32 bits each.
- Thirty-two hash value (32-bit) words.
- SHAL6's final result is a 1024-bit message digest.

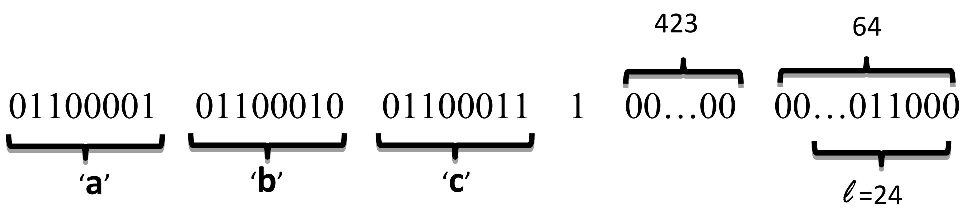
The words of the message schedule are labelled, $W_0, W_1, \ldots, W_{143}$. The 32 working variables are labeled, ***reg1, reg2, reg3, ...,*** and ***reg32***. The hash value's words are labelled, $H_0^{(i)}, H_1^{(i)}, \ldots, H_{31}^{(i)}$. These values will hold the initial hash value, $H^{(0)}$, replaced by each successive intermediate hash value, $H^{(i)}$(after each message block is processed), and ending with the final hash value, $H^{(N)}$. SHAL6 also uses seven temporary words: T_1, T_2, T_3,..., *and* T_7.

3.2 SHAL6 Preprocessing

The preprocessing stage consists of four main steps illustrated in the following subsections:

a. Pad the message, *M*

The length of the message, *M*, is l bit. Append the bit '1' to the end of the message, followed by *k* zero bits, where *k* is the smallest, non-negative solution to the equation l + 1 + *k* ≡ 448mod512. Then, append the 64-bit block equal to the number l, expressed using a binary representation. For example, the (8-bit ASCII) message '**abc**' has length 8 × 3 = 24, so the message is padded with one bit, then 448 − (24 +1) = 423 zero bits. Then, the message length becomes the 512-bit padded message.



The length of the padded message should now be a multiple of 512 bits.

b. Parse the padded message into N 512-bit message blocks, $M^{(1)}$, $M^{(2)}$, …, and $M^{(N)}$

The padded message is parsed into *N* 512-bit blocks, $M^{(1)}$, $M^{(2)}$, …, and $M^{(N)}$. Since the 512 bits of the input block may be expressed as sixteen 32-bit words, the first 32 bits of the message block *i* are denoted as $M_0^{(i)}$. The next 32 bits are $M_1^{(i)}$, and so on up to $M_{15}^{(i)}$.

c. Set the initial hash value, $H^{(0)}$

For SHAL6, the initial hash value, $H^{(0)}$, shall consist of the following 32 (32-bit) words in hex. Table 2 represents the initial values of the SHAL6 hashing algorithm. These words were obtained by taking the first 32 bits of the square roots' fractional parts of the first 32 prime numbers.

Table 2 The Initial Values of the SHAL6 Hashing Algorithm

$H_0^{(0)}$ = 6a09e667	$H_8^{(0)}$ = cbbb9d5d	$H_{16}^{(0)}$ = ae5f9156	$H_{24}^{(0)}$ = d94ebeb1
$H_1^{(0)}$= bb67ae85	$H_9^{(0)}$ = 629a292a	$H_{17}^{(0)}$ = cf6c85d3	$H_{25}^{(0)}$ = cc4a611
$H_2^{(0)}$ = 3c6ef372	$H_{10}^{(0)}$ = 9159015a	$H_{18}^{(0)}$ = 2f73477d	$H_{26}^{(0)}$ = 261dc1f2
$H_3^{(0)}$ = a54ff53a	$H_{11}^{(0)}$ = 152fecd8	$H_{19}^{(0)}$ = 6d1826ca	$H_{27}^{(0)}$ = 5815a7be
$H_4^{(0)}$ = 510e527f	$H_{12}^{(0)}$ = 67332667	$H_{20}^{(0)}$ = 8b43d457	$H_{28}^{(0)}$ = 70b7ed67
$H_5^{(0)}$ = 9b05688c	$H_{13}^{(0)}$ = 8eb44a87	$H_{21}^{(0)}$ = e360b596	$H_{29}^{(0)}$ = a1513c69
$H_6^{(0)}$ = 1f83d9ab	$H_{14}^{(0)}$ = db0c2e0d	$H_{22}^{(0)}$ = 1c456002	$H_{30}^{(0)}$ = 44f93635
$H_7^{(0)}$ = 5be0cd19	$H_{15}^{(0)}$ = 47b5481d	$H_{23}^{(0)}$ = 6f196331	$H_{31}^{(0)}$ = 720dcdfd

3.3 The SHAL6 Hash Computation

The SHAL6 hash computation uses functions and constants. SHAL6 uses seven logical functions, which operate on three to five 32-bit words represented by *v*, *w*, *x*, *y*, and *z*. The result of each function is a new 32-bit word described as follows:

Ch(x, y, z) = (~x ∧ ~ z) ⨁ (~z ∧ y)

Maj(x, y, z) = (x ∧ y) ⨁ (y ∧ x) ⨁ (x ∧ z)

$\sum_0^{\{1024\}}(x) = ROTR^{39}(x) \oplus ROTR^{33}(x) \oplus ROTR^{23}(x) \oplus ROTR^{43}(x)$

$\sum_1^{\{1024\}}(x) = ROTR^{36}(x) \oplus ROTR^{40}(x) \oplus ROTR^{35}(x) \oplus (x)$

$\sigma 0_0^{\{1024\}}(x) = ROTR^{37}(x) \oplus (x) \oplus ROTR^{45}(x) \oplus SHR^{48}(x)$

$\sigma 1_1^{\{1024\}}(x) = ROTR^{34}(x) \oplus ROTR^{40}(x) \oplus (x) \oplus SHR^{44}(x)$

Comb(v, w, x, y, z) = ((x ∧ y) ⨁ (y ∧ x) ⨁ (x ∧ z)) ⨁ ((x ∧ y) ⨁ (y ∧ x) ⨁ (x ∧ z)) ∧ ~v) ⨁ (~((x ∧ z) ⨁ (~x ∧ ~z)) ∧ w))

Where:

Ch: Choice, if x is 1, the choice will be y, and if x is 0, the choice will be z.

Maj: Majority, the greatest significance or quantity in a given set of choices or data points.

Comb: It is a short combination of *Maj* and *Ch*.

SHAL6 uses a sequence of 32 constant 32-bit words, $K_0^{\{1024\}}, K_1^{\{1024\}}, \ldots, K_{31}^{\{1024\}}$. These words represent the first 32 bits of the cube roots' fractional parts of the first 32 prime numbers. In hex, these constant words read from left to right.

Table 3 The Constant Values of the SHAL6 Hashing Algorithm

428a2f98	d807aa98	e49b69c1	983e5152
71374491	12835b01	efbe4786	a831c66d
b5c0fbcf	243185be	0fc19dc6	b00327c8
e9b5dba5	550c7dc3	240ca1cc	bf597fc7
3956c25b	72be5d74	2de92c6f	c6e00bf3
59f111f1	80deb1fe	4a7484aa	d5a79147
923f82a4	9bdc06a7	5cb0a9dc	06ca6351
ab1c5ed5	c19bf174	76f988da	14292967

After preprocessing is completed, each message block, $M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$, is processed in order, using the following steps:

For *i* = 1 to *N*:

{

a. Prepare the message schedule, {*Wt*}:

$$W_t = \{M_t^{(i)} \quad 0 \leq t \leq 15 \; \sigma_1^{\{1024\}}(W_{t-2}) + \sum_1^{\{1024\}}(W_{t-12}) + \sigma_0^{\{1024\}}(W_{t-7}) + \sum_1^{\{1024\}}(W_{t-15}) + \sigma_0^{\{1024\}}(W_{t-15}) + \sum_1^{\{1024\}}(W_{t-4}) + \sigma_0^{\{1024\}}(W_{t-8}) \; 16 \leq t \leq 143$$

b. Initialize the 32 working variables, *reg1*, *reg2*, *reg3*, … and *reg31*, with the (*i*-1)st hash value:

Table 4: The Constant Values of the SHAL6 Hashing Algorithm

Reg1 = $H_0^{(i-1)}$	Reg9 = $H_8^{(i-1)}$	Reg17 = $H_{16}^{(i-1)}$	Reg25 = $H_{24}^{(i-1)}$
Reg2 = $H_1^{(i-1)}$	Reg10 = $H_9^{(i-1)}$	Reg18 = $H_{17}^{(i-1)}$	Reg26 = $H_{25}^{(i-1)}$
Reg3 = $H_2^{(i-1)}$	Reg11 = $H_{10}^{(i-1)}$	Reg19 = $H_{18}^{(i-1)}$	Reg27 = $H_{26}^{(i-1)}$
Reg4 = $H_3^{(i-1)}$	Reg12 = $H_{11}^{(i-1)}$	Reg20 = $H_{19}^{(i-1)}$	Reg28 = $H_{27}^{(i-1)}$
Reg5 = $H_4^{(i-1)}$	Reg13 = $H_{12}^{(i-1)}$	Reg21 = $H_{20}^{(i-1)}$	Reg29 = $H_{28}^{(i-1)}$

Reg6 = $H_5^{(i-1)}$	Reg14 = $H_{13}^{(i-1)}$	Reg22 = $H_{21}^{(i-1)}$	Reg30 = $H_{29}^{(i-1)}$
Reg7 = $H_6^{(i-1)}$	Reg15 = $H_{14}^{(i-1)}$	Reg23 = $H_{22}^{(i-1)}$	Reg31 = $H_{30}^{(i-1)}$
Reg8 = $H_7^{(i-1)}$	Reg16 = $H_{15}^{(i-1)}$	Reg24 = $H_{23}^{(i-1)}$	Reg32 = $H_{31}^{(i-1)}$

c. For t = 0 to 143:

{

T1 = $Maj(reg12, reg20, reg24) + \Sigma_1^{\{1024\}}(reg6) + K_t^{\{1024\}} + W_t$

T2 = $\Sigma_1^{\{1024\}}(reg10) + \sigma_0^{\{1024\}}(reg26) + reg3 + Ch(reg29, reg2, reg17)$

T3 = $reg22 + \sigma_0^{\{1024\}}(reg21) + K_t^{\{1024\}} + Comb(reg6, reg14, reg22, reg28, reg30) + Maj(reg5, reg9, reg30)$

T4 = $\Sigma_0^{\{1024\}}(reg31) + \sigma_0^{\{1024\}}(reg4) + reg32 + K_t^{\{1024\}} + Ch(reg1, reg8, reg19)$

T5 = $Maj(reg11, reg15, reg23) + Comb(reg5, reg29, reg4, reg32, reg13) + K_t^{\{1024\}} + W_t + \Sigma_0^{\{1024\}}(reg30) + \sigma_1^{\{1024\}}(reg2)$

T6 = $K_t^{\{1024\}} + W_t + reg7 + \Sigma_1^{\{1024\}}(reg18) + \sigma_1^{\{1024\}}(reg14) + Maj(reg27, reg28, reg30)$

T7 = $reg12 + \Sigma_0^{\{1024\}}(reg19) + \sigma_1^{\{1024\}}(reg16) + \Sigma_1^{\{1024\}}(reg7) + \sigma_0^{\{1024\}}(reg14) + Maj(reg17, reg13, reg20) + K_t^{\{1024\}} + W_t$

There are 32 registers, and their calculations will be the following for each iteration:

reg(i) = reg(i−1)

Except for the following registers, the T values will be used to enhance bit shuffling:

reg32 = reg31 + T7

reg27 = reg26 + T5

reg22 = reg21 + T6

reg19 = reg18 + T3

reg15 = reg14 + T7

reg11 = reg10 + T4

reg7 = reg6 + T2

reg1 = T1 + T5 }

d. Compute the i^{th} intermediate hash value H(i):

$H_0^{(i)}$ = reg(i + 1) + $H_i^{(i-1)}$

}

Where the value of i starts from 0 to 31.

After repeating steps (a to d), a total of N times (i.e. after processing $M^{(N)}$), the resulting 1024-bit message digest (i.e. the hash result) of the message, M, is:

Hash result = $H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)} \| H_8^{(N)} \| H_9^{(N)} \| H_{10}^{(N)} \| H_{11}^{(N)} \| H_{12}^{(N)} \| H_{13}^{(N)} \|$

$H_{14}^{(N)} \| H_{15}^{(N)} \| H_{16}^{(N)} \| H_{17}^{(N)} \| H_{18}^{(N)} \| H_{19}^{(N)} \| H_{20}^{(N)} \| H_{21}^{(N)} \| H_{22}^{(N)} \| H_{23}^{(N)} \| H_{24}^{(N)} \| H_{25}^{(N)} \| H_{26}^{(N)} \| H_{27}^{(N)} \| H_{28}^{(N)} \| H_{29}^{(N)} \| H_{30}^{(N)} \| H_{31}^{(N)}$

3.4 The SHAL6 Message Pseudocode

The following is the pseudocode of the proposed SHAL6 algorithm. The hashing process is divided into phases to simplify and make it easy to understand. The first phase involves initialising the constants and registers. At this level, there are 32 constants and registers, and their values will be used for enhanced shuffling and mixing with the binary values of the input message. Then, the message preparation phase will commence, which includes converting the message to a binary value by padding it with zeros. The third phase involves preparing the message schedule, which includes creating 144 words per block message based on the equations below. Finally, the most bit shuffling occurs during compression, the heart of the hashing process. The steps are shown below in a pseudocode format:

- ***Constants:***

```
Initialise K [32], predefined constants
Initialise H [32], initial hash values
```

- ***Message preparation:***

```
messageInASCII = convertToASCII (message)
messageInBinary = convertToBinary (messageInASCII)
messageInBinary = messageInBinary + ‘1’
messageInBinary = pad (messageInBinary, (len (messageInBinary) + 64) % 512, ‘0’)
messageLength = convertToBinary (len (messageInASCII))
messageLength = pad (messageLength, 64, ‘0’)
messageInBinary = messageInBinary + messageLength
```

- ***Process the message in successive 512-bit blocks:***

For each block in messageInBinary (512-bit chunks), W [144] = initialise an array to store the message schedule.

▪ *Prepare the message schedule:*

```
For j = 0 to 15:
 W[t] = substring (block, j * 32, 32)
For j = 16 to 143:
 W[i] = sigma1 (W[i – 2]) + ROTR1 (W[i – 12]) + sigma0 (W[i – 7]) + ROTR1 (W[i – 15]) + sigma0 (W[i – 15]) + ROTR1 (W[i – 4]) + sigma0 (W[i – 8])
```

▪ *Initialise working variables:*

reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9, reg10, reg11, reg12, reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20, reg21, reg22, reg23, reg24, reg25, reg26, reg27, reg28, reg29, reg30, reg31, reg32 Copy of H.

▪ *Compression loop: (Note: sigma0 = o0, sigma1 = o1, ROTR0 = s0, ROTR1 = s1)*

```
for t = 0 to 143:
 T1 = Maj(reg12, reg20, reg24) + s1(reg6) + K[t] + W[t]
[1]    T2 = s1(reg10) + o0(reg26) + reg3 + Ch(reg29, reg2, reg17)
 T3 = reg22 + o0(reg21) + K[t] + Comb (reg6, reg14, reg22, reg28, reg30) + Maj(reg5, reg9, reg30)
 T4 = s0(reg31) + o0(reg4) + reg32 + K[t] + Ch(reg1, reg8, reg19)
 T5 = Maj(reg11, reg15, reg23) + Comb(reg5, reg29, reg4, reg32, reg13) + K[t] + W[t] + s0(reg30) + o1(reg2)
 T6 = K[t] + W[t] + reg7 + s1(reg18) + o1(reg14) + Maj(reg27, reg28, reg30)
 T7 = reg12 + s0(reg19) + o1(reg16) + s1(reg7) + o0(reg14) + Maj(reg17, reg13, reg20) + K[t] + W[t]
```

- ***Update hash values:***

Final hash:

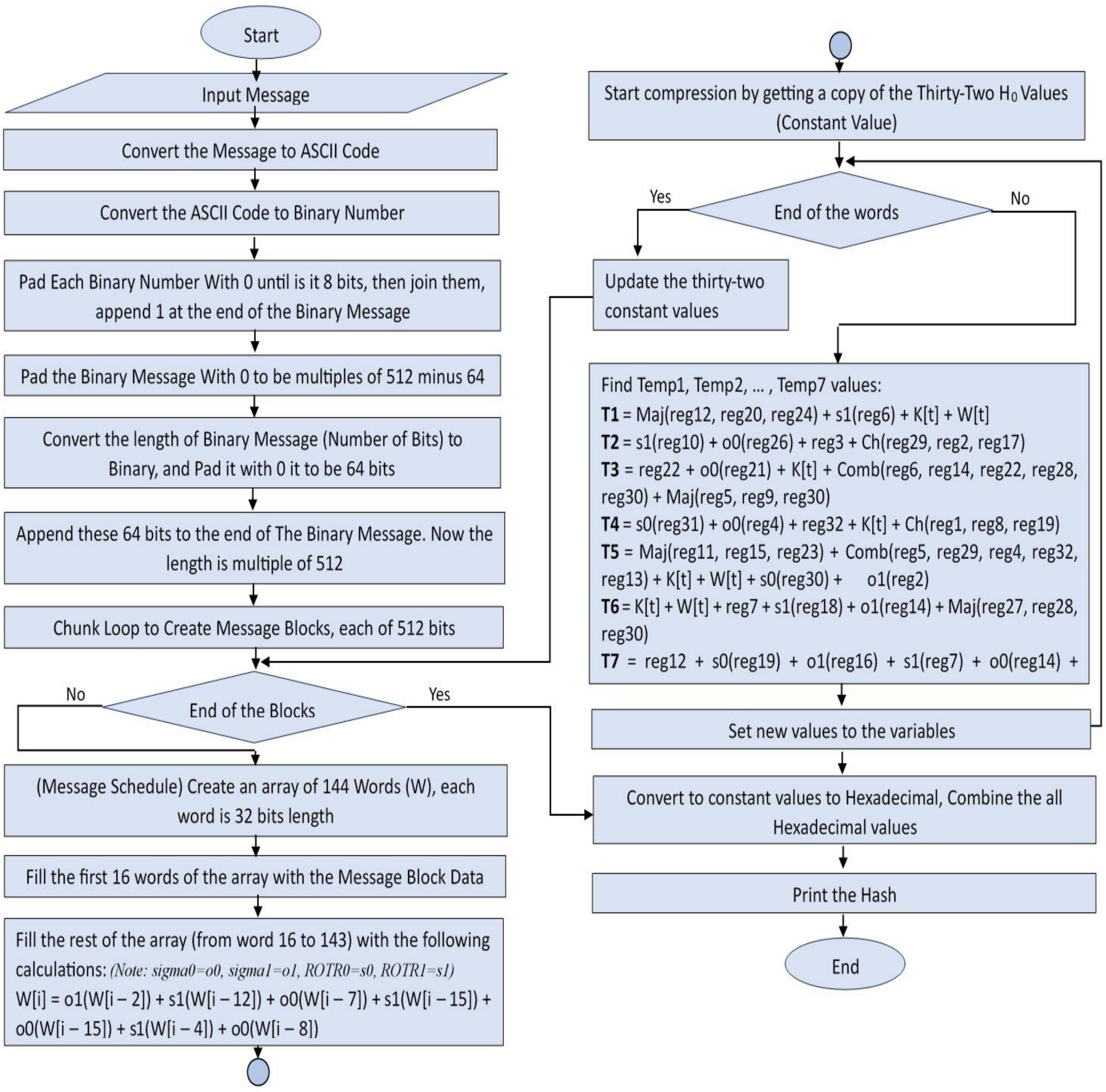
Hash result = H [0] to H [31] as hexadecimal.

- ***Return hash result***

3.5 The SHAL6 Flowchart

4. Results and Discussion

All the proposed algorithm levels have been tested, and the analysis of the results will be presented in this section. Then, the results will be compared with the SHA-256 Algorithm. We modified the proposed algorithm levels and the SHA-256 algorithm to capture the number of iterations and the executed arithmetic and logical operations to conduct the testing correctly. In addition, the same message was conferred to each algorithm, i.e. first 'abc' first and then 'ABC'. Table 5 displays the outputs of each proposed algorithm-level and the SHA-256 algorithm.



Start
Input Message
Convert the Message to ASCII Code
Convert the ASCII Code to Binary Number
Pad Each Binary Number With 0 until is it 8 bits, then join them, append 1 at the end of the Binary Message
Pad the Binary Message With 0 to be multiples of 512 minus 64
Convert the length of Binary Message (Number of Bits) to Binary, and Pad it with 0 it to be 64 bits
Append these 64 bits to the end of The Binary Message. Now the length is multiple of 512
Chunk Loop to Create Message Blocks, each of 512 bits
No
End of the Blocks
Yes
(Message Schedule) Create an array of 144 Words (W), each word is 32 bits length
Fill the first 16 words of the array with the Message Block Data
Fill the rest of the array (from word 16 to 143) with the following calculations: (Note: sigma0=o0, sigma1=o1, ROTR0=s0, ROTR1=s1)
W[i] = o1(W[i – 2]) + s1(W[i – 12]) + o0(W[i – 7]) + s1(W[i – 15]) + o0(W[i – 15]) + s1(W[i – 4]) + o0(W[i – 8])
Start compression by getting a copy of the Thirty-Two H0 Values (Constant Value)
Yes
End of the words
No
Update the thirty-two constant values
Find Temp1, Temp2, … , Temp7 values:
T1 = Maj(reg12, reg20, reg24) + s1(reg6) + K[t] + W[t]
T2 = s1(reg10) + o0(reg26) + reg3 + Ch(reg29, reg2, reg17)
T3 = reg22 + o0(reg21) + K[t] + Comb(reg6, reg14, reg22, reg28, reg30) + Maj(reg5, reg9, reg30)
T4 = s0(reg31) + o0(reg4) + reg32 + K[t] + Ch(reg1, reg8, reg19)
T5 = Maj(reg11, reg15, reg23) + Comb(reg5, reg29, reg4, reg32, reg13) + K[t] + W[t] + s0(reg30) + o1(reg2)
T6 = K[t] + W[t] + reg7 + s1(reg18) + o1(reg14) + Maj(reg27, reg28, reg30)
T7 = reg12 + s0(reg19) + o1(reg16) + s1(reg7) + o0(reg14) +
Set new values to the variables
Convert to constant values to Hexadecimal, Combine the all Hexadecimal values
Print the Hash
End

Fig. 4. Flowchart of SHAL6 of the new hashing algorithm.

Input Algo.	**abc**	**ABC**
SHA-256	**ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad**	**b5d4045c3f466fa91fe2cc6abe79232a1a57cdf104f7a26e716e0a1e2789df78**
SHAL1	d5b3455b	807e067c
SHAL2	8264a9c187f8339a	e39ebe9ce26ac2d6
SHAL3	64b5410fbd7bb403385448d71d75eae4	5ced17fd9930d8ea050a878f0a45126c
SHAL4	dfe15a6f8edf92404b5875f2235518391afe0d24f3881bf011a1e9e378bb7d3e	6a54b96bf6e6821b067ecf3ccc6f2f70dd0d67edab410e43a020d7976d5936eb
SHAL5	f217b67aacf35981859d1c8304d62656c83318688f2a169dea808b888b8b6fbe084ae13db5c10e7e2542235c2777e0052cd18fd043b0b1a323e650ff9270410e	d92f63b09d7dc68939647267748553d370c77cb6de266655807068f79d1614e69f6a82474b6b908dbd5d80da5a648b207993dcc29d1b7da16a8d228576c448a8
SHAL6	b0a73371ac4381808615a0e3210053e5bd1a2012c5030f4cb5fefab6599055728dc131bc377bf5c025f116cd46bc6ca1cf33836170dbc81ca281804aecddf62c7e11c8df52569dfd3bc7bc7aa9f58982ae179c6476b972c6861c6aa9c0e843f964000eaac3677a1ebc3e3d17fdd0fff57f9c27925f7e1e0a95f481df70fcd87c	779c5ccc5f2afe6c507eaf61c9d1b3f7af4941533f91591c91c1f7c3c1f02831576057d5c9bb04f5b43cc41e3add41fad3b718a047b09d278ea8b364797a6d2a42f6b64f3e2bf7f49a3ec6084c728aba46424c037f279f1fe924892a2277436d9a468050da9ce9f39c99b93493786f74c486ccf04b1591d94d2585d21dc82302
SHAL7	3f559919e9b8f7429bb3a3384880e1ff230dd3d76d8abdd76583133f2a016c27148c6459e7f37ed8ee663f9a9966373fba23ca348b7b445b1fba6c9e1d151c4e2ee8de835a4e2fb330805823e2326a6a6dfdbd4ce4d6d5df467db88db86978106bb090967dc833831424084931c8657c61a176a32c3022364b9cef23c6d04d3abeda8bbcbb3b772401c08aa81a379fce54512678281602d7108a31bbe0aa729a9509d57e6aefb0e710203f99c9ce81a21382a875aaf31f24017a5c5af11d7e130c0a1ce497960a2dd9ae2b072db5d6d1873519a571ef92065f885b93ab86ccb7cc473f695868fdc2363c5f6c83bef8f4e2a83e806edd913cb736b44f2b41ae1b	6d9677b39550d3adf42db402c4471eae44714f35ab20a418d61e73196f23b4abe1149e952d350f68fea9f8bbb33db04a1110ba33c9aed53a36dca5afe848afc40992ea613ba9babcc1825841bfa2e1bdbf851608890821b178a11692b291ca7b0d13ac801d82e75a98a203b0816b05616893c05f60b7ee75e77274f1930b41676f1e9e4e16cb8ccd0e8813743cb07c56d0267ea2823ace71e4de5257e34fe3cf63581659b21e88d0cb29c584a69b4360d7c00cf876ac91738f0b2132f0ddc5e614e0bfc865fe97900ae8118340c1945997e98e50c92860098838bb76a4361b10d2d8942984015671ee81065aa52e91e7d59d7dc50896fc37b921169d26edfd0e
SHAL8	e55e41eb2a49bcfcda5aaeae2aa1a9619438f0b880c9e6035f424c5a414cf697284ff95d21b038e936b65b516ef84ce2ccfcd188e53225bf0d4685bc65b1960692e11cb923959d19ac1c8ce265a295d4a8a5a4184c8aa332c8c1c275e5b8da0dbf8a283ded153f9d7e4e6ff578ba46e38af0b13eafb17876103760a26b6ca61197234f6b8a5e0f21eb085ee8d64f6cbc75adea5764bfc4628cca6e0ebaa0975098c8347fbc0ed7def8c0dcedc9cdf910269db78a144597f939605ea7d29d8ef8f5fe549c967600741f22eedc9c9e55b109730d61d64d29411ef705eb134cfc96b3f26d76d53da196bc30d00ed019f71a5ff6a92b3c03d66b266b73788444947957e5d5bf2113b35435df58f63bb530cdbbf268071e8f8b8759dbefe3e897a59eae1f5e7284b6a897d291a9e21ead265a7ffd90eba880f1a5fbf74b19068e7ba098d53586918b8462d2a45d1a4482255e421fd663d7e470024d631c16d77d184e2a60c9d5c3999b64e5cc4e498127b9eb9d507ce0c268eb198e13c2ba122f39c7ee4213030830dbfbef2ae3a4e4fb4dffb95370932c54c99944b485eef73cdc6687ada861cdb7ba1476b2527fe1f839501c101d97de605371103db7ef780a5eaf4fe19f47c2cfdb3b0a70078364416f11b28ed0661feb1a765cbf84e27b03341e2e42a8c444f83b0e060a146772398dd3528de896c4ca0eab13177a8327230888	cbabd18e03367162955a7e3ec015be7518c081a31a2ea575291ae0415d629366633cde7a5be96ca7dfa408ec3516634e46e834cba4155300176e4208ceb2761df17624b8c633e181d307165edbb3162029862cad6caacccaa3a80dced81db7e79b1429276d4c9ac2c09c4d9d084359ba15e0222988dfe92bfc0a16ba306df98f39040a93c7f1eb6f14e8a02fbc7bf5452eb6168d17c7bde0ff22e1acdfa674b9fcab218cb66693fbd0ffd79e089f38370e27be5b7e32b7dd9692b3bb16ae95b753f05b305966ee3d16cacaf778b6c474d90ce8b367dd0f2e0d2a4b4adf47cc3291b78a5a706ae26b2aa16547a20498787a046bd2bde099c656a39acfb1bed3f163fc095a298e355359a8fdeca1c7f370fdd13f4773b1e19ff4ce530e0d3dec8271045857e8d2384714079a25baab96f201c64956c2a3ce1e4059a3b42c188a6577d08308641082acac2426b4ac63223067f13b7c275565591bbc4ab3c7109d0360e1a095d0e313e24b7491085c8891398a0f15cac7f1709a30f686e043dcc007029e5f67f63e8ed13fb860caed1db7e0a39cdc8435945572f37ba37475bdf86cfa371a6b3eb6cd096447302fa1a9ef8185ce9ae70f497912a64baf94aa8b9f7995d8427ab7d5bda07be7d9c85041a708eeb701378dd9e7ec21d27e0ed6d9fb1e091aa190222a0ed364172d0b3ceec97e668c95ecc813ed6451e47bef3805749e

Table 5: The Output of Each Proposed Algorithm-Level and the SHA-256 Algorithm

Table 5 displays the range of outputs for each algorithm level, spanning from 8 to 128 hexadecimal. The least advanced level caters to IoT devices with highly constrained specifications and minimal processing capability. Conversely, the most advanced level suits devices endowed with substantial processing power. It is important to note that the security diminishes as the levels decrease due to reduced bit shuffling. This is because the count of executed arithmetic and logical operations was constrained. Table 6 shows the important properties captured for each algorithm level and compares them to SHA-256 when the message input is ‘abc’.

Table 6: The Important Captured Properties of Each Proposed Algorithm-Level and the SHA-256 Output

Algorithm Type Characteristics	**SHA-256**	**SHAL1**	**SHAL2**	**SHAL3**	**SHAL4**	**SHAL5**	**SHAL6**	**SHAL7**	**SHAL8**
# Register	8	1	2	4	8	16	32	64	128
# Constant Values	64	1	2	4	8	16	32	64	128
# Words	64	20	28	32	58	68	144	170	200
Output Length (bit)	256	32	64	128	256	512	1024	2048	4096
# Iterations	64	20	28	32	58	68	144	170	200
# Arithmetic Operations	600	105	154	228	540	1192	6416	10986	17920
# Logical Operations	1696	64	504	736	1942	3420	26688	47650	85208

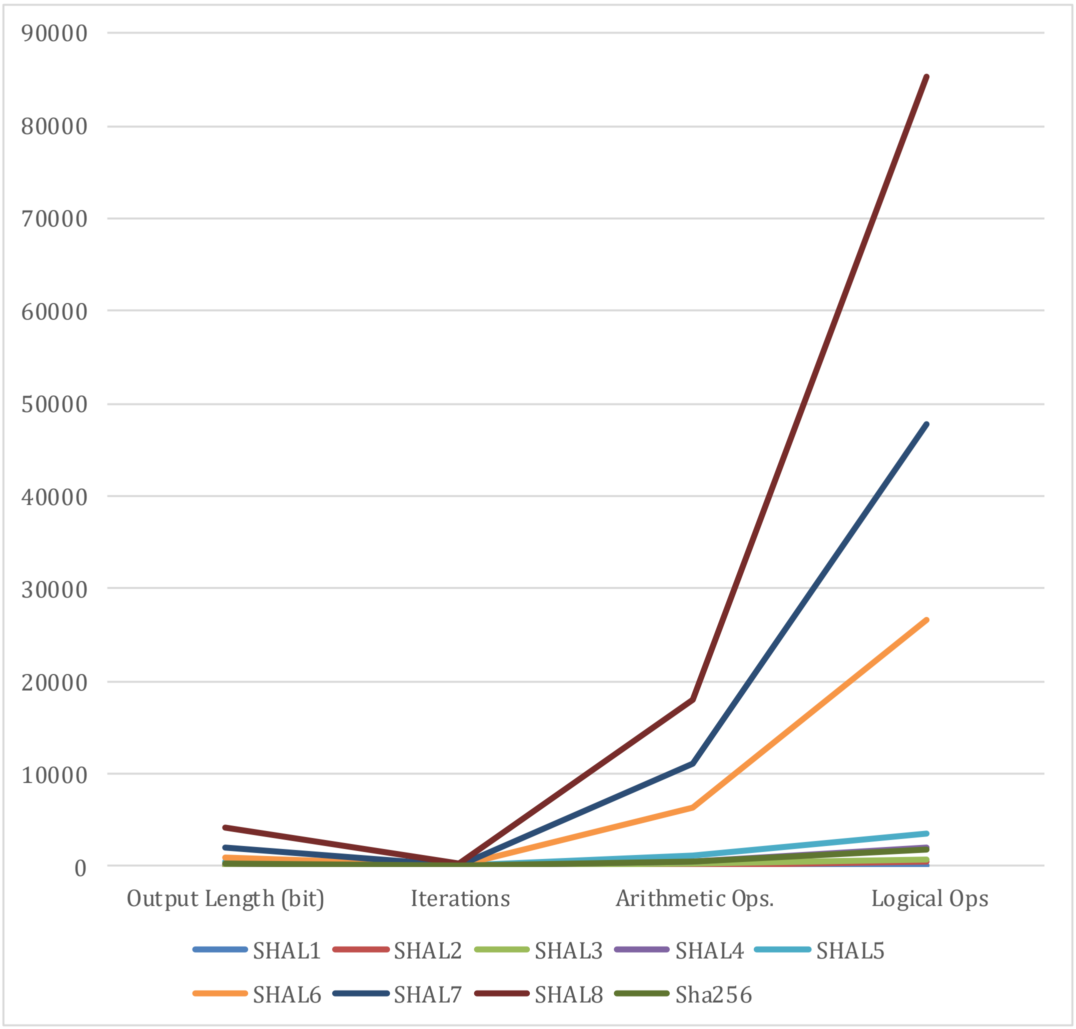


Fig. 5. Line chart showing the operations, output lengths and number of iterations of the proposed hashing algorithm.

Overall, the number of arithmetic and logical operations and output lengths of each algorithm level indicates the robustness of the algorithm level. For example, the total executed arithmetic and logical operations for level 1 is only 169. In contrast, level 8 reached 103,128 operations. This suggests that level 1 only requires 0.163% of the processing power compared to level 8 of the proposed algorithm. Conversely, SHA-256 requires 2,296 operations to hash the same message. This suggests that the SHA-256 algorithm only requires 2.226% of the computing power compared to level 8 of the proposed algorithm. Figure 5 displays the number of operations, output length, and iterations required at the different levels.

5. Conclusion

In this study, a new MLHA is proposed, designed and implemented. The proposed algorithm consists of eight levels, starting with SHAL1 for IoT devices with limited resources and progressing to SHAL8 for devices with high power consumption properties. The main outcomes observed for the proposed MLHA are related to the increasing number of characteristics at the various levels. These characteristics include the number of registers, constant values, words, output length (bit), iterations and arithmetic and logical operations. Based on the results of this study, it was observed that the proposed MLHA outperforms the existing, commonly used SHA-256 algorithm. Therefore, the algorithm proposed in this paper can be regarded as the most advanced due to its flexibility and compatibility with any IoT device regardless of its resource and power consumption requirements.

References

[1] M. Sathyan, ‘Industry 4.0: Industrial internet of things (IIOT)’, *Advances in computers*, vol. 117. no. 1, pp. 129–164, Elsevier, 2020.

[2] M. Sudip et al., ‘Industrial Internet of Things for safety management applications: A survey’, *IEEE Access*, vol. 10, pp. 83415–83439, 2022.

[3] L. David et al., ‘The industrial internet of things networking framework’, *Industrial IoT Consortium* (2021).

[4] K. Halim et al., ‘IoT, IIoT, and cyber-physical systems integration’, *Emergence of Cyber Physical System and IoT in Smart Automation and Robotics: Computer Engineering in Automation*. Cham: Springer International Publishing, 2021, pp. 31–50.

[5] A .A. Y. Fahad and A. Popa, ‘LMAAS-IoT: Lightweight multi-factor authentication and authorization scheme for real-time data access in IoT cloud-based environment’, *J. Netw. Comput. Appl.,* vol. 192, p. 103177, 2021.

[6] E. Abderrahmane and K. Maalmi, ‘A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment’, *J. Big Data,* vol. 6, pp. 1–25, 2019.

[7] H. Yosra et al., ‘Big data and IoT-based applications in smart environments: A systematic review’, *Comput. Sci. Rev.,* vol. 39, p. 100318.

[8] Williams, Phillip, et al. ‘A survey on security in internet of things with a focus on the impact of emerging technologies’, *Internet of Things,* vol. 19, p. 100564, 2022.

[9] T. Lo’ai et al., ‘IoT privacy and security: Challenges and solutions’, *Applied Sciences,* vol. 10, no. 12, p. 4102, 2020.

[10] S. E. R. Kumar and E. A. Dash, ‘Unveiling the shadows: Exploring the security challenges of the Internet of Things (IoT)’, 2023.

[11] L. V. Cherckesova et al., ‘Developing a new collision-resistant hashing algorithm’, *Mathematics,* vol. 10, no. 15, p. 2769, 2022.

[12] A. Abouchouar and F. Omary, ‘New implementation of the abstract design for non-iterative hash functions with 1024 digest length NIHF-1024’, *Int. J. Artif. Intell.,* vol. 11, no. 4, p. 1213, 2022.

[13] M. Shah, ‘Secure hash algorithms for securing IoT’, Conference secretariat. p. 63. 2021.

[14] T. Usman et al., ‘A critical cybersecurity analysis and future research directions for the Internet of Things: A comprehensive review’, *Sensors,* vol. 23, no. 8, p. 4117, 2023.

[15] E. T. Michailidis and D. Vouyioukas, ‘A review on software-based and hardware-based authentication mechanisms for the Internet of Drones’, *Drones,* vol. 6, no. 2, p. 41, 2022.

[16] P. Monika and H. J. Kaur, ‘Comparative analysis of secured hash algorithms for blockchain technology and Internet of Things’, *Int J Adv Comput Sci Appl,* vol. 12, no. 3, 2021.

[17] R. Vidya and K. V. Prema, ‘Comparative study of lightweight hashing functions for resource constrained devices of IoT’, presented at the 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), IEEE, City, State, Country, Month days, 2019.

[18] U. I. Khan et al., ‘Evolution and analysis of secured hash algorithm (SHA) family’, *Malays. J. Comput. Sci.,* vol. 35, no. 3, pp. 179–200, 2022.

[19] T. M. G. Sami, S. R. M. Zeebaree and S. H. Ahmed, ‘A comprehensive review of hashing algorithm optimization for IoT devices’, *Int. J. Intell. Syst. Appl. Eng,* vol. 11, no. 6s, pp. 205–231, 2023.

[20] G. Sandip et al., ‘A journey from md5 to sha-3’, *Trends in Communication, Cloud, and Big Data: Proceedings of 3rd National Conference on CCB, 2018,* City, Singapore: Springer, 2020.

[21] K. Sakan, S. Nyssanbayeva, N. Kapalova, K. Algazy, A. Khompysh and D. Dyusenbayev, ‘Development and analysis of the new hashing

algorithm based on block cipher’, *Eastern-European J. Enterp. Technol.*, vol. 2, pp. 60–73, 2022.

[22] A. V. Tutueva, A. I. Karimov, L. Moysis, C. Volos, D. N. Butusov, ‘Construction of one-way hash functions with increased key space using adaptive chaotic maps’, *Chaos Solit. Fractals,* vol. 141, p. 110344, 2020

[23] Y. Wang, L. Chen, X. Wang, G. Wu, K. Yu and T. Lu, ‘The design of keyed hash function based on CNN-MD structure’, *Chaos Solit. Fractals,* vol. 152, p. 111443, 2021.

[24] E. V. V. Da Silva, ‘A New Architecture for Hash Functions’, In Proceedings of the Future Technologies Conference (FTC), Vancouver, BC, Canada, Nov. 28–29, 2021.

[25] H. E. N. Liquan et al., ‘New hash function based on C-MD structure and chaotic neural network’, *Chin. J. Netw. Inf. Secur.* vol. 9, no. 3, 2023.

[26] M. Al-Zubaidie. ‘Implication of lightweight and robust hash function to support key exchange in health sensor networks’, *Symmetry,* vol. 15, no. 1, p. 152, 2023.

[27] S. M. Myint, S. Moe, M. M. Myint and A. A. Cho, ‘A study of SHA algorithm in cryptography’, *Int. J. Trend Sci. Res. Dev,* vol. 3, pp. 1453–1454, 2019.

[28] S. U. A. Laghari et al, ‘ES-SECS/GEM: An efficient security mechanism for SECS/GEM communications’, *IEEE Access,* vol. 11, pp. 31813–31828, 2023.

[29] S. A. A Hakeem, M. A. A. El-Gawad and H. Kim, ‘A decentralized lightweight authentication and privacy protocol for vehicular networks’, *IEEE Access,* vol. 7, pp. 119689–119705, 2019.

[30] P. P. Pittalia, ‘A comparative study of hash algorithms in cryptography’. *Int. J. Comput. Sci. Mob. Computing,* vol. 8, no. 6, pp. 147–152, 2019.

[31] X. Yi et al., *Blockchain Foundations and Applications*. Springer Nature, 2022.

[32] O. Vashchuk and R. Shuwar, ‘Pros and cons of consensus algorithm proof of stake. Difference in the network safety in proof of work and proof of stake’, *J. Electron. Inf. Tech,* vol. 9, no. 9, pp. 106–112, 2018.

[33] Z. Rong and W. K. V. Chan, ‘Evaluation of energy consumption in block-chains with proof of work and proof of stake’, *J. Phys. Conf. Ser*., vol. 1584, no. 1, 2020.

[34] J. Guo et al., ‘Practical collision attacks against round-reduced SHA-3’, *J. Cryptol.,* vol. 33, pp. 228–270, 2020.

[35] S. Peng, S. Pal and L. Huang, Ed. *Principles of Internet of Things (IoT) ecosystem: Insight paradigm*. Springer International Publishing, 2020.

[36] J. H. Nord, A. Koohang and J. Paliszkiewicz, ‘The Internet of Things: Review and theoretical framework’, *Expert Syst. Appl.,* vol. 133, pp. 97–108, 2019.

[37] S. E. Bibri, ‘The core academic and scientific disciplines underlying data-driven smart sustainable urbanism: An interdisciplinary and transdisciplinary framework’, *Computational Urban Science,* vol. 1, pp. 1–32, 2021.

[38] Q. Chen, H. Shi and J. Chen, ‘Development management of infant dairy industry integrating Internet of Things under the background of family planning policy adjustment’, *Secur. Commun. Netw.,* 2022.

[39] Y. Khan et al., ‘Application of Internet of Things (IoT) in sustainable supply chain management’, *Sustainability,* vol. 15, no. 1, p. 694, 2022.

[40] J. Jamali et al., ‘IoT architecture’, *Towards the Internet of Things: Architectures, Security, and Applications,* pp. 9–31, 2020.

[41] Alejandro Garcia, Machine Learning for Customer Segmentation and Targeted Marketing , Machine Learning Applications Conference Proceedings, Vol 3 2023.

[42] Begum, S. . S. ., Prasanth, K. D. ., Reddy, K. L. ., Kumar, K. S. ., & Nagasree, K. J. . (2023). RDNN for Classification and Prediction of Rock or Mine in Underwater Acoustics. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3), 98–104. https://doi.org/10.17762/ijritcc.v11i3.6326