ASSIGNMENT 2 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 5: Security		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Thai Kim Xuan Quynh	**Student ID**	GCS210236
Class	GCS1004A	**Assessor name**	Nguyen Xuan Sam

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P1	P2	P3	P4	M1	M2	D1

❒ Summative Feedback: ❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:
Lecturer Signature:		

Table of Contents

Table of Figure

1. Introduction

Data protection is a crucial aspect of information security management, and organizations rely on best practices and guidance to help them achieve their goals. According Fundamentals of Information Systems Security about ISO/IEC 27002 provides recommendations on information security management practices, and it is designed to direct these recommendations to management and security personnel responsible for information security management systems. The standard outlines information security within the context of the CIA triad, which includes confidentiality, integrity, and availability.

Confidentiality ensures that only authorized users can access data while ***Integrity*** ensures only authorized users can modify data.

Finally, ***Availability*** ensures that authorized users can access information when requested (Kim and Solomon, 2010).

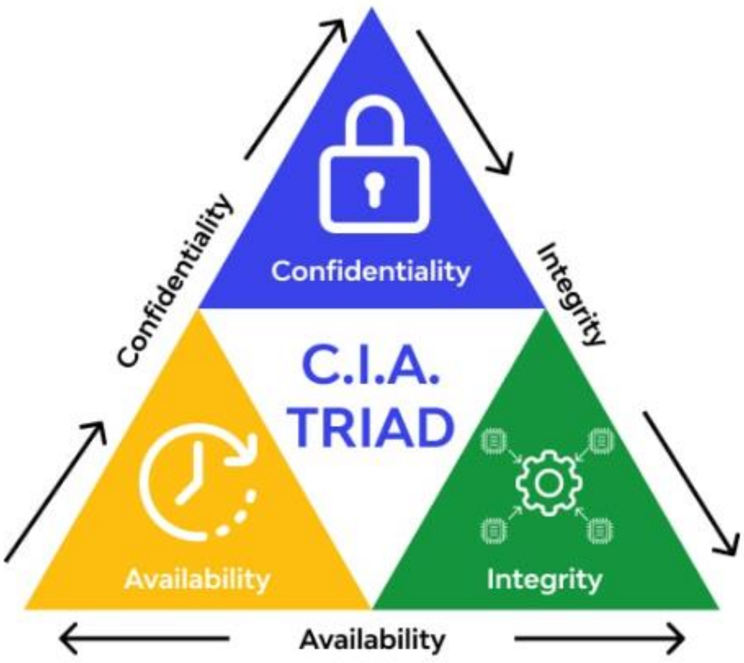

Figure 1: C.I.A Triad

- Confidentiality: Confidentiality is the assurance that information is not disclosed to unauthorized individuals, entities, or processes. In other words, confidential data must be protected from unauthorized access, disclosure, or exposure. This type of data may include personal information, trade secrets, intellectual property, financial information, and confidential business information. Organizations that are responsible for storing and processing confidential information must ensure that it is protected by using security mechanisms that limit access to authorized personnel only. Confidentiality is particularly important in industries such as healthcare, finance, and government where the exposure of private data could lead to significant damage.

- Integrity: Integrity refers to the trustworthiness of data and the assurance that it is accurate and has not been altered or tampered with in any way. Integrity ensures that data is reliable and maintains its accuracy and validity. The integrity of data may be compromised by malfunctions, technical glitches, or unauthorized access. To ensure data integrity, organizations must implement security techniques such as access controls, data backups, and monitoring procedures to detect and address any modifications or tampering attempts.

- Availability: Availability refers to the assurance that authorized users can access data when they need it. Data must be available to authorized personnel at all times so that they can perform their work-related tasks quickly and effectively. Downtime or loss of access to data can lead to significant operational losses, reduced productivity, and damaged reputations. Organizations must implement appropriate security measures to ensure that data is always available, including backup systems, redundancy, and failover mechanisms.(Samonas and Coss, 2014)

1.1 Motivations

Ensuring the confidentiality, integrity, and availability of data is a critical aspect of information security management. Organizations rely on best practices and guidance, such as the ISO/IEC 27002 standard, to help them achieve their goals. One of the biggest concerns for information security is brute-force attacks, where attackers use repetitive and automated login attempts to crack user passwords. Brute-force attacks put sensitive data at risk, and organizations must implement measures to prevent them from accessing their systems and data.

1.2 Objectives

The objective of this paper is to explore an approach to protect data from brute-force attacks using hashing algorithms. We will discuss the effectiveness of this approach, its potential limitations, and how it can help organizations safeguard sensitive data.

My report also aims to prevent brute force attacks, which are considered a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks. The hacker tries multiple usernames and passwords, often using a computer to test a wide range of combinations until they find the correct login information.

1.3 Overview of the brute-force attacks

Brute-force attacks consist of automated login attempts to crack user passwords by trying different combinations of characters. Attackers can quickly and easily apply machine algorithms to guess passwords and exploit username and password login pages to gain unauthorized access to data. Weak or simple passwords make it easier for attackers to conduct brute-force attacks, and organizations must take steps to avoid risk. One approach to preventing brute-force attacks is through the use of hashing algorithms, which can provide a means of securely storing passwords and making them difficult to crack.

1.4 Summary

In conclusion, organizations must implement measures to safeguard sensitive data against brute-force attacks. The ISO/IEC 27002 standard provides recommendations on information security management practices, and its guidance directs its recommendations to management and security personnel responsible for information security management systems. The CIA triad is a framework for information security management that consists of three core principles: confidentiality, integrity, and availability. Organizations can use these principles as a foundation for a strong information security posture and implement measures such as password hashing to prevent unauthorized access to data.

2. Methodology

The methodology for this paper involves a review of previous research on information security, data protection, and brute force attacks or maybe also has an RFID system. We will analyze different approaches to protecting against brute-force attacks, including using hashing algorithms. We will conduct a case study to evaluate the effectiveness of password hashing in protecting data against brute-force attacks. The case study will involve simulated brute-force attacks on a database using different hashing algorithms. The results will be analyzed to determine which algorithm is most effective in preventing unauthorized access

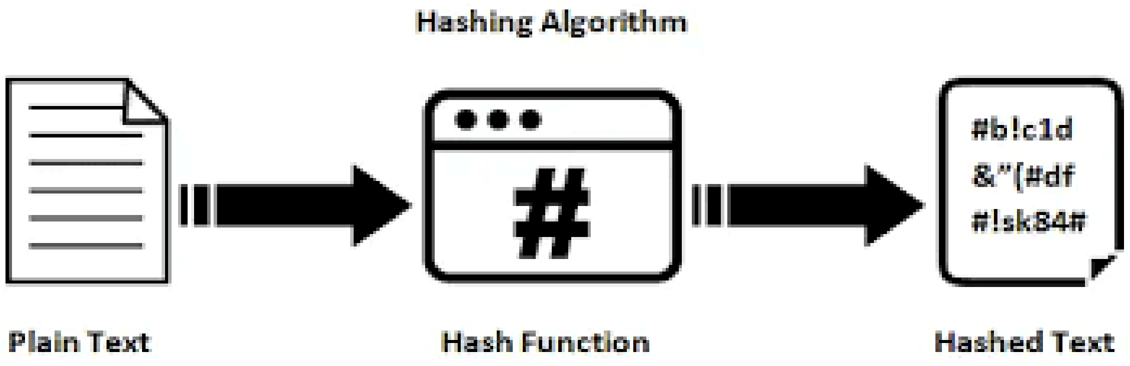

Figure 2: Hashing Algorithm

Explain more about the RFID system is a contactless automatic identification system that has attracted much attention recently. The most important problem with an RFID system is that an adversary can access the tag information, which gives rise to privacy and forgery problems. This paper presents a hash-based mutual authentication protocol as a solution. The proposed protocol is designed to send a random number generated by a tag to a back-end server without disclosure. Moreover, it substitutes a random number with a secret value, which is employed in a response message. The properties of the proposed protocol enable the constant creation of distinct response messages without interferences from intended or meaningless requests generated by an adversary, while the secret value is not directly transmitted. Our proposed protocol makes it difficult for an attacker to launch successful brute-force attacks against our approach (Cho et al., 2011)

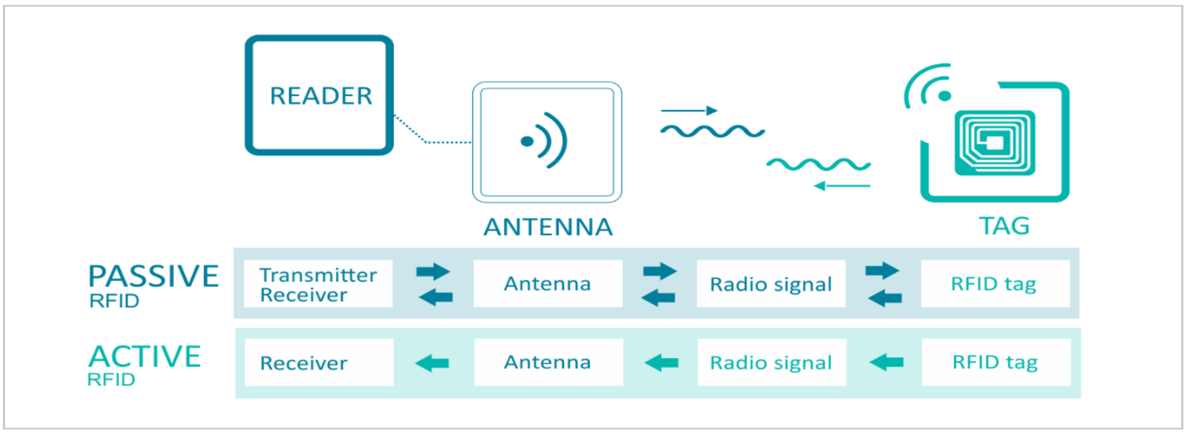

Figure 3: RFID system

Every RFID system consists of three components: a scanning antenna, a transceiver, and a transponder. When the scanning antenna and transceiver are combined, they are referred to as an RFID reader or interrogator. There are two types of RFID readers -- fixed readers and mobile readers. The RFID reader is a network-connected device that can be portable or permanently attached. It uses radio waves to transmit signals that activate the tag. Once activated, the tag sends a wave back to the antenna, where it is translated into data.

The transponder is in the RFID tag itself. The read range for RFID tags varies based on factors including the type of tag, type of reader, RFID frequency, and interference in the surrounding environment or from other RFID tags and readers. Tags that have a stronger power source also have a longer read range (Cho et al., 2011).

3. Proposed the algorithm

Firstly, to hash data based on a String (Wei et al., 2017), I propose the following algorithm:

- Convert the user's password to a String.
- Input the String into the desired hashing algorithm (e.g. SHA-256 or MD5) to generate a unique hash value.
- Store the hash value in the database instead of the user's actual password.
- When a user logs in, their password is hashed using the same algorithm, and the resulting hash is compared to the hash stored in the database.
- If the hashes match, the user is authenticated and granted access to their account.

By following this algorithm, sensitive information can be kept secure while still allowing for user authentication. However, it is important to note that even with this algorithm there are still potential security risks, and users should take steps to protect their information such as using strong passwords and enabling two-factor authentication.

String-based data hashing is a simple and effective method for keeping passwords and other sensitive information safe in a database. Data hashing algorithms such as SHA-256 and MD5 are commonly used in information security because of their irreversible nature.

When the user's password is generated, it is first converted to a String. This string is then input into a hash function or hashing algorithm to generate a unique numerical value, called a hash.

The hash will be stored in the database instead of the user's original password. When a user tries to log in to his or her account, the password will be re-hashed using the same algorithm and the new hash will be compared with the old hash stored in the database. If the two hashes match, the user is authenticated and allowed access.

An important benefit of using String-based data hashing is the security of the user's password. Once the password is hashed, it is no longer plain text and cannot be read. Storing the hash as a value instead of the original password reduces information security risks, if a piece of software or agency accidentally or intentionally discloses the passwords of some users.

However, there are also some security issues associated with using String-based data hashing. Attackers can use dictionary attack techniques, use hash tables, or use hash analysis tools to break the hash and find the original password. Therefore, to enhance security, users need to create strong passwords and use strong security tools such as two-factor authentication or password managers.

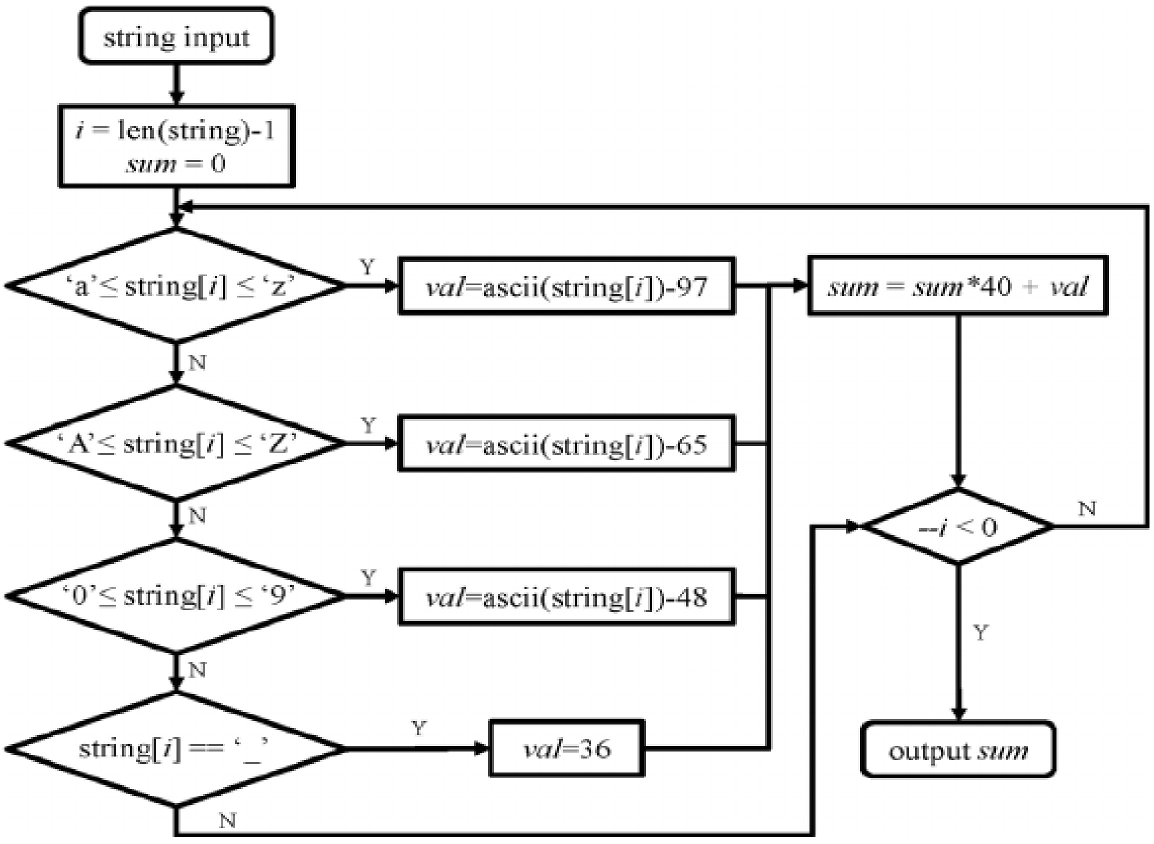

Figure 4: Hash data based on a String

Secondly, this flowchart describes the process of generating a hash value corresponding to an input character string with the hash algorithm used as SHA-256. The steps are as follows:

- ***Step 1:*** Starting the hashing process, the input is converted to a binary string.
- ***Step 2:*** Chunking: This binary chain is divided into fixed-size chunks so that these blocks can be processed independently of each other. The size of these blocks is 512 bits, which is the optimal size for processing.
- ***Step 3:*** Using Padding bits: After the string is divided into blocks, padding bits are added to the last block to ensure that the length of the string is divisible by 512 bits.

- ***Step 4:*** Initialize: SHA-256 uses 8 initial values, which are hexadecimal integers, to initialize the hash. These values are stored in an array.
- ***Step 5:*** Word Processing Function: Each block is processed by the dict integer function and passed through 64 loops to generate a hash value. Each loop uses a different coefficient and is calculated by intermediate values and stored in an output chain.
- ***Step 6:*** Final hash: After all blocks are processed, the final result is a hexadecimal string representing the corresponding hash value.

This flowchart is a visual way to understand the processing in SHA-256. The use of techniques such as padding bits and word processing function helps to ensure that the hashing algorithm will be highly reliable and irreversible.

Thirdly, SHA-256 (Gilbert and Handschuh, 2004) is an iterative one-way hash function that can compress a message < 264 bits in length to produce a reduced 256-bit value called a message digest. The compression process can be divided into four stages as shown in Figure 5:

- *Pre-Processing*: this stage occurs first and consists of two steps: segmenting and buffering the message into data blocks (Mt) of 512 bits in length and setting the initial hash variables (H(0~7).)) will be used in the hash calculation.
- *Message Schedule*: the sequence of data blocks (Mt) is fed into a second stage, called the Message Schedule, where they are expanded to 64 32-bit words (Wt).
- *Digest Calculation*: each 32-bit word (Wt) is put into a third stage called Digest Calculation. During this phase, there are eight 32-bit working variables (a ~ h), initialized with the initial hash variables (H(0~7)). These variables and two 32-bit temporary variables (T1, T2) are computed and updated in each loop 64 times. After 64 iterations, the values of the 8 working variables are sent to the final stage, called the Digest Update.
- *Digest Update:* during this phase, the hash variables (H(0~7)), containing the original hashes, are updated by adding them to the working variables. Typically, a data block operation will include the Message Schedule, Digest Calculation, and Digest Update phases. The hash calculation will complete when all data blocks have been processed.

Due to the one-way hashing nature and the non-existent analysis of the hash, SHA-256 is considered a secure solution for protecting information. The use of SHA-256 helps to ensure the security of the data and ensures that the user's information will not be easily exposed.

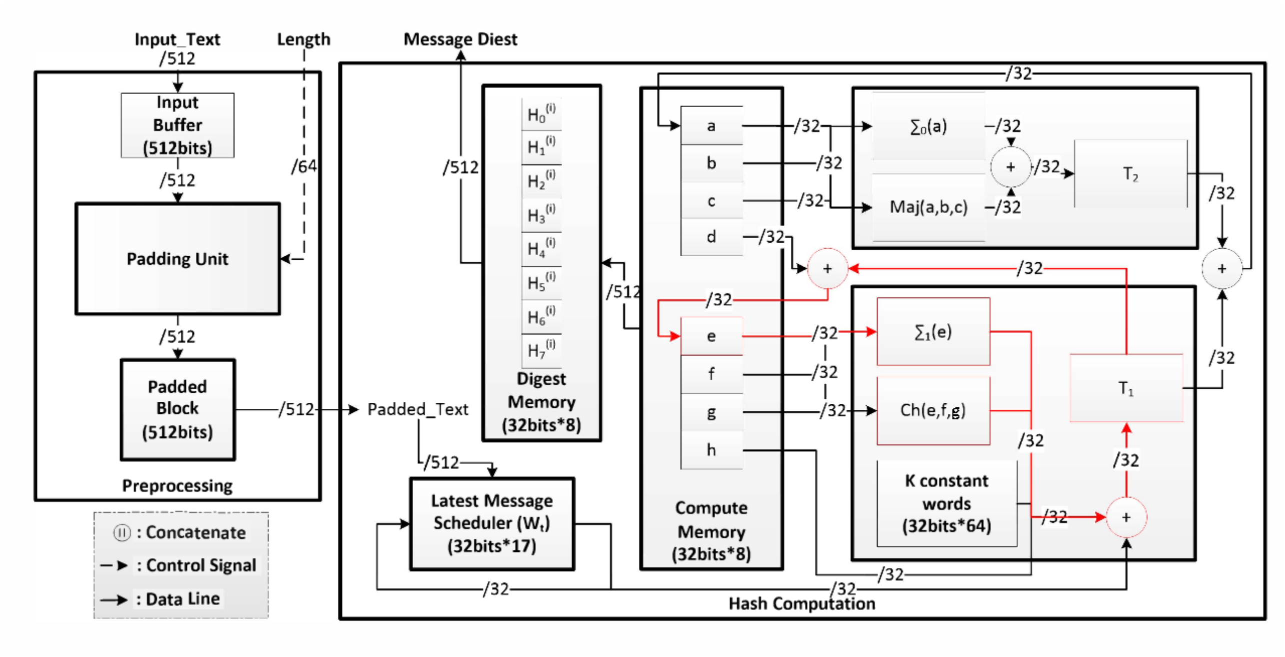

Figure 5: SHA-256 algorithm flow diagram

Below is the pseudocode of the SHA-256 algorithm. And we will explain about it

- ***Step 1: Initialize hash values***

Initialize variables:

h0 = 0x6a09e667

h1 = 0xbb67ae85

h2 = 0x3c6ef372

h3 = 0xa54ff53a

h4 = 0x510e527f

h5 = 0x9b05688c

h6 = 0x1f83d9ab

h7 = 0x5be0cd19

- ***Step 2: Pre-process the message***

Pad the message with a 1-bit followed by 0s so that the length is congruent to 448 modulo 512

Append a 64-bit representation of the original length of the message

- ***Step 3: Process the message in 512-bit chunks***

```
For each 512-bit chunk of the message:
    Create a 64-entry message schedule array, word W[0..63], of 32-bit words
    Initialize working variables:
        a = h0
        b = h1
        c = h2
        d = h3
        e = h4
        f = h5
        g = h6
        h = h7
    for i from 0 to 63:
        If i < 16:
            W[i] = the i-th 32-bit word of the block
        else:
            s0 = (W[i-15] rightrotate 7) xor (W[i-15] rightrotate 18) xor (W[i-15] rightshift 3)
            s1 = (W[i-2] rightrotate 17) xor (W[i-2] rightrotate 19) xor (W[i-2] rightshift 10)
            W[i] = W[i-16] + s0 + W[i-7] + s1

        t1 = h + ((e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)) + ((e and f) xor ((not e) and g)) + k[i] + W[i]
    t2 = ((a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)) + ((a and b) xor (a and c) xor (b and c))
    h = g
    g = f
    f = e
    e = d + t1
    d = c
    c = b
    b = a
    a = t1 + t2
  Update the hash values:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e
    h5 = h5 + f
    h6 = h6 + g
    h7 = h7 + h
```

- ***Step 4: Produce the final hash value***

Concatenate the hash values h0, h1, h2, h3, h4, h5, h6, and h7, in that order, to produce a 256-bit hash value of the message.

The SHA-256 algorithm pseudocode can be divided into ***four main steps***:

- *Step 1:* Initializing the hash values: This step sets the initial hash values (h0 through h7) that will be used in the hashing process.

- *Step 2:* Pre-processing the message: The message is padded with a 1-bit followed by 0s so that the length of the message is congruent to 448 modulo 512. A 64-bit representation of the original message length is appended to the end of the message as well.
- *Step 3:* Processing the message in 512-bit chunks: The message is processed in 512-bit chunks. For each chunk, an individual message schedule array is created and then twelve 32-bit words are computed for each iteration of the 64-entry array, which is then used to update the eight working variables (a through h).
- *Step 4:* Producing the final hash value: After all the data chunks have been processed, the hash values are concatenated in a specific order to produce the final 256-bit hash value.

The SHA-256 algorithm is designed in a way that ensures that even small changes in the input message will result in a completely different output hash value. This makes it an extremely effective way to protect data from tampering or modifications.

4. Results and analysis

4.1 Proof and analyze the algorithm

The proposed algorithm for hashing data based on a string is simple and effective. By converting the user's password to a string and inputting it into a hashing algorithm like SHA-256, we can generate a unique hash value stored in the database instead of the user's original password. This ensures that sensitive information is kept secure while still allowing for user authentication.

The SHA-256 algorithm is a popular choice for hashing data due to its one-way nature and high level of security. By using techniques like padding bits and word processing functions, SHA-256 generates an irreversible hash value that is strongly resistant to attacks.

The pseudocode of the SHA-256 algorithm shows how it works in practice. The algorithm consists of four main steps: initializing the hash values, pre-processing the message, processing the message in 512-bit chunks, and producing the final hash value. By following these steps, the SHA-256 algorithm can compress a message to produce a reduced 256-bit value called a message digest, which is highly secure and resistant to tampering.

Overall, the proposed algorithm for hashing data based on a string and the SHA-256 algorithm has been proven to be effective and reliable in protecting sensitive information and ensuring user authentication.

While SHA-256 is a highly secure hashing algorithm, it is also worth noting, it is not invulnerable to attacks. As technology advances, it is possible that new vulnerabilities may be discovered. Therefore, it is important to regularly review and update security measures to stay ahead of potential threats.

Furthermore, while hashing is an important security measure, it is not a replacement for other security measures like encryption and access controls. These measures should also be implemented in conjunction with hashing to create a layered approach to security.

Lastly, it is important to consider the privacy implications of using hashing algorithms. In cases where data protection laws apply, such as GDPR or CCPA, hashing may be considered personal data and subject to privacy regulations. Organizations must ensure that they are properly handling hashed data and complying with relevant regulations.

In conclusion, while hashing algorithms like SHA-256 can effectively protect sensitive information, it is important to consider the potential security risks and privacy implications and implement a comprehensive approach to security (Gilbert and Handschuh, 2004)

4.2 Discussions

While the proposed algorithm and SHA-256 algorithm are effective, there are still potential security risks associated with using hashing algorithms. Attackers can use techniques like dictionary attacks, hash tables, or hash analysis tools to break the hash and find the original password. This is why it is important for users to create strong passwords and use security tools like two-factor authentication to enhance security.

In addition, it is essential to stay up to date with the latest advancements in information security and continuously review and enhance security measures to ensure the highest level of protection for sensitive information.

Overall, the proposed algorithm and SHA-256 algorithm are solid solutions for hashing data based on a string and protecting sensitive information. However, it is important to remain vigilant and proactive in addressing any potential security risks.

5. Conclusions

The use of hashing algorithms is an essential approach to preventing brute-force attacks, thereby improving information security. Several hashing algorithms can be applied, including bcrypt, PBKDF2, and scrypt. Additionally, to enhance security further, salting can be used in combination with password hashing. Salting is a random string of characters that is appended to the password before it is hashed. This makes it more challenging for attackers to use precomputed tables of commonly used passwords to crack the hash.

It is essential to recognize that while hashing algorithms can help prevent brute-force attacks, they are not foolproof. To improve security measures, organizations must also include efforts in educating users on creating strong passwords and maintaining them securely. Further requirements include limiting login attempts, implementing multi-factor authentication, amongst others.

Guidance from industry-recognized standards such as ISO/IEC 27002 can contribute significantly to improving information security management systems. Implementing these standards and best practices can help organizations safeguard data against brute-force attacks and other data breaches. In conclusion, organizations that take this approach may enhance their information security measures and meet the requirements of best practices in protecting sensitive data.

References

CHO, J.-S., YEO, S.-S. & KIM, S. K. 2011. Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value. *Computer communications,* 34**,** 391-397.

GILBERT, H. & HANDSCHUH, H. Security analysis of SHA-256 and sisters. Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003. Revised Papers 10, 2004. Springer, 175-193.

KIM, D. & SOLOMON, M. G. 2010. *Fundamentals of information systems security*, Jones & Bartlett Publishers.

SAMONAS, S. & COSS, D. 2014. The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security,* 10.

WEI, H., YU, J. X. & LU, C. 2017. String similarity search: A hash-based approach. *IEEE Transactions on Knowledge and Data Engineering,* 30**,** 170-184.