

**The Science of Computing — Assignment 1 2023**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module title:** | The Science of Computing | | |
| **Module code**: | UI110010 | **Word-count:** | 2100?? |
| **Deadline:** | 27th September 2024, 23:59 | **Date submitted:** | 27/09/2024 |
| **Student number:** | 21010093 | | |

**This assessment is worth 40% of the total marks for this module**.

**Completing your assessment:**

**IMPORTANT**

**This assessment should be word processed using a plain 12-point font, with one-and-a-half spacing and 1 inch margins. Submit your completed report, including this front cover sheet, via Blackboard. Please include your word count in the box above. The word limit is 2100 words. As usual, penalties will apply if your submission is over by more than 10%.**

**Declaration of originality and authorisation to hold this assessment electronically and verify that it is original:** UHI recognises that plagiarism, where deliberately engaged in, is unacceptable and is considered serious academic malpractice. **Students are responsible for ensuring the work they submit is their own**. If you have any queries you should contact your PAT or the Module Leader before submitting your assessment.

**By submitting this assessment I declare that the attached piece of work is my own**. I have acknowledged all the sources I have consulted and where I have used words which are not my own, I have clearly indicated this in the references.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student number:** | 21010093 | **Date:** | 26/09/2024 |

Contents

[Task 1 2](#_Toc178262941)

[Introduction 2](#_Toc178262942)

[The Role of Computer Science in Developing and Maintaining Robust Hashing Algorithms 2](#_Toc178262943)

[The Emerging Threat of Quantum Algorithms 3](#_Toc178262944)

[Contemporary Algorithms 4](#_Toc178262945)

[Conclusion 5](#_Toc178262946)

[Task 2 5](#_Toc178262947)

[Introduction 5](#_Toc178262948)

[Answers 5](#_Toc178262949)

[Conclusion 8](#_Toc178262950)

[References 9](#_Toc178262951)

[Table 1. Time & space complexity in relation to password protocol. 8](#_Toc178262806)

# Task 1

## Introduction

Hashing transforms an arbitrary length sequence of bits into a hash (fixed length sequence of bits) which can be later compared against a second hash for authentication, data integrity verification or data retrieval purposes.

For a hashing function to be considered cryptographically secure it must be infeasible to:

* Find 2 inputs that produce the same hash (collision attack).
* Find an input that produces a given hash (preimage attack).
* Find a different input that produces the same hash as a given input (second-preimage attack).

This report will analyze the role of computer science in this field as well as evaluate recent studies going more in depth on some of their key findings.

## The Role of Computer Science in Developing and Maintaining Robust Hashing Algorithms

The importance of hashing has been recognized since the early stages of computer science (Chi & Zhu, 2017, p. 3) and it plays a pivotal role in the field of hashing algorithms with many important responsibilities.

Computer scientists must design and implement new algorithms as old algorithms are shown to have vulnerabilities for example with MD-5 which was first broken in 2004 (Wang, Feng, Lai, & Yu, 2004) further improvements were made to attacks with a collision being found in less than a minute on a consumer notebook in 2006 (Klíma, 2006) and more recently SHA-1 which had theoretical collision attacks faster than brute-force as early as 2005 but proved too computationally expensive until after slow improvements being made a practical attack was first performed in 2017 being able to "craft two colliding PDF documents containing arbitrary distinct images" (Stevens, Bursztein, Karpman, Albertini, & Markov, 2017). This clearly shows the importance of continually creating new algorithms to protect the users' data which is reliant on these fundamental algorithms remaining secure.

High-performance hashing algorithms are required to prevent excessive wait times for users processing large sets of data, but performance cannot be at the expense of highly secure algorithms. Due to the ever-increasing processing power available to the average user algorithms naturally become more performant over time, however, this also naturally diminishes the security of hashing algorithms as attackers can perform hashes faster as well and thus attacks become easier. This naturally facilitates the need for gradual improvements to and replacements of algorithms, which can be easily seen in the case of SHA first published in 1993, revised in 1995 producing SHA-1, SHA-2 was subsequently proposed in 2001 (Chaves, et al., 2016, p. 82) and finally SHA-3 announced in 2012 after a competition running since 2008 ended with the Keccak algorithm winning and becoming the new SHA-3 standard (Chaves, et al., 2016, p. 93).

As more secure hashing standards are created, and older algorithms are compromised it is the responsibility of computing professionals to adopt these new standards and migrate old systems with potentially large amounts of data and intricate compatibility concerns. There seems to be a reluctance in the industry to move to new standards where there is no strong incentive, examples of this include the move from IPv4 to Ipv6 taking over 20 years, and the transition from SHA-1 to SHA-2 to SHA-3 still ongoing with very slow adoption of SHA-3 (Loebenberger, et al., 2024, pp. 1-2). A practical example is the migration of hashes in a database where they cannot be simply converted due to the one-way nature of the hashing algorithm but instead require a progressive transition as users login providing their passwords which can then be used to generate the new hash.

## The Emerging Threat of Quantum Algorithms

Recent studies have discussed the growing risks considering quantum computers, with one paper stating, "Recent groundbreaking achievements in quantum computing mean that in the nearest future, even more effective attacks on hashing algorithms will be possible." (Cherckesova, Safaryan, Lyashenko, & Korochentsev, 2022, p. 1), and another corroborating with, "Quantum computers threaten traditional hash functions." (Kushwaha & Shirsat, 2024, p. a197). Within the quantum setting the best algorithm for a generic or "brute-force" attack is the BHT algorithm presented in 1998 by Brassard, HØyer, & Tapp, but this requires an environment with an exponentially large qRAM (Lee & Hong, 2024, p. 239) and current quantum computers have very limited qRAM with IBM hoping to have a system with a mere 200 qubits by 2029 and 2000 qubits by 2033 (Schneider & Smalley, 2024). A generic attack can be performed with the CNS algorithm with linear memory complexity but at the cost of increasing time complexity compared to a BHT attack (Chailloux, Naya-Plasencia, & Schrottenloher, 2017). There is currently active research on reducing the time complexity of quantum-based attacks, it has been shown this can be reduced as compared to a quantum generic attack (Lee & Hong, 2024, p. 257) and over time it is only likely to be improved upon bringing us ever-closer to genuine threats from the quantum computing space.

## Contemporary Algorithms

Another area of discussion is the need to find novel hash functions (Kushwaha & Shirsat, 2024, p. a197) as aside from the quantum threats a "steady increases in computing power has already made many cryptographic attacks that were previously considered only theoretical now practical for malicious users" (Cherckesova, Safaryan, Lyashenko, & Korochentsev, 2022, p. 1). Ullah and Pun present a novel algorithm that uses self-learning neurons and a forwarding and backward propagation process during the implementation phase to generate collision free hash values with initial experimentation showing that the proposed algorithm is faster and more collision resistant than existing state-of-the-art algorithms (Ullah & Pun, 2023). Another new algorithm is Nik-512 which uses a reduced number of rounds compared to alternatives with each round being more computationally expensive. During experimentation Nik-512 was able to out-perform other hashing algorithms in terms of the avalanche effect, indicating increased collision resistance (Cherckesova, Safaryan, Lyashenko, & Korochentsev, 2022).

## Conclusion

Although recent studies have focused on the growing threat from quantum computers capable of performing attacks significantly faster than conventional algorithms and the improvements to these quantum attacks, we are speaking on the order of years and not weeks or even months before these attacks are likely to be practical and there are multiple algorithms that have been shown to be quantum resistant with no doubt more to come in that time. Aside from quantum threats there are the same conventional threats that have existed since the beginning of hashing algorithms which are continuously bolstered from improving hardware and algorithms as well.

Computer science plays a significant role in developing and maintaining robust hashing algorithms. The very creation, testing and improvement of these algorithms is dependent on the field of computer science, additionally, computing professionals must migrate existing systems to new more secure algorithms as expediently as possible to protect the data of users, with potentially catastrophic results where they fail to do so.

# Task 2

## Introduction

Password hashing is extensively used in authentication systems protecting systems and user data from unauthorized access. A cryptographic hash function transforms given passwords into fixed-size sequences of bits which are often represented as hexadecimal numbers for brevity. The time and space complexity of the set of hashes for a given password scheme varies greatly depending on the number of possible passwords defined by the password scheme as we will explore here.

## Answers

1. A password is hashed by using a cryptographic hashing algorithm which takes a string as input and produces a fixed-size sequence of bits. When a password is created only the hash of the password is stored and then when authentication is performed the hash of a given password can be calculated and compared to the stored hash. This is more secure than storing the password because attackers are unable to steal the password directly and acquiring the hash is less useful as it would then need to be cracked which is practically infeasible where a secure hashing algorithm has been used. The most important characteristics of cryptographic hashing functions are as follows:
   * Determinism - The same input must always produce the same output.
   * Avalanche Effect - A small change to the input must result in a significant change to the output.
   * Preimage Resistance - It must be practically impossible to find an input that produces a specific output hash.
2. Attackers can mount an array of attacks on password-protected systems, for example:
   * Brute-Force Attack - Every possible combination of characters that could make up a password are systematically tried until a match is found; this is usually incredibly computationally expensive due to the sheer number of combinations possible.
   * Dictionary Attack - A list of common passwords or words gathered about a specific target is used to construct passwords and test them against the authentication system until a match is found.
   * Password Recovery Attack - If a recovery mechanism can be compromised then an attacker can simply recover the victim’s password to gain access.

If an attacker has a list of stolen password-hashes then they can perform offline attacks on this list by hashing guessed passwords and comparing the generated hash to the stolen one which is faster than having to interact with the login page, potentially by a great margin where powerful hardware with a high hash rate is used.

1. A salt is a random value added to a password before it is hashed. The salt needs to be stored alongside the hash so that the same hash can be calculated during future authentication. The main purpose of salting a password is to prevent attackers from precomputing a dictionary of known passwords into a list of hashes known as a rainbow table which could then be directly compared to a list of stolen hashes without having to compute the hashes in real-time, when salted, the salt for each user has to be added to each known password in the list then hashed before it can be compared to the stolen hashes. This will slow down attackers but does not actually prevent them from compromising any accounts where they are able to guess the password using a dictionary attack or any other method.

4) 5) 6) Following are the explanations for each calculation succeeded by the final table.

* + Based on today's date 11/09/2024 the oldest 65-year-old would be turning 66 tomorrow and so would have been born on 12/09/1958 and the youngest 15-year-old was born 11/09/2009. There are 18628 days between these dates inclusively (13 29th of February's) using 2 significant figures this equates to 19000 days.
  + Time to compute dictionary of passwords: 18628 x 5 µs = 0.093s
  + Dictionary storage space requirement: 18628 x 54 bytes = 980 KB
  + There are 26 lower case letters and 10 digits which gives 26 x 26 x 26 x 26 x 10 x 10 x 10 x 10 or 26^4 x 10^4 which gives a grand total of 4,569,760,000 possible passwords or 4.6 million
  + Time to compute dictionary of passwords: 4,569,760,000 x 5 µs = 6.3 hours
  + Dictionary storage space requirement: 4,569,760,000 x 54 bytes = 220 GB
  + 12 digits makes the total number of possible passwords 10^12 which equals 1,000,000,000,000 or 1 trillion.
  + Time to compute dictionary of passwords: 1,000,000,000,000 x 5 µs = 58 days
  + Dictionary storage space requirement: 1,000,000,000,000 x 54 bytes = 49 TB

1. * 12 lower case characters gives a total of 26^12 possible passwords which equals 95,428,956,661,682,176 or 95 quadrillion.
   * Time to compute dictionary of passwords: 95,428,956,661,682,176 x 5 µs = 15,000 years
   * Dictionary storage space requirement: 95,428,956,661,682,176 x 54 bytes = 4.5 EB (Exabytes)
2. * All case-sensitive alpha numeric characters give us a-z, A-Z and 0-9 or 62 possible characters and thus we have 62^12 possible passwords which equals 3,226,266,762,397,899,821,056 or 3.2 sextillion.
   * Time to compute dictionary of passwords: 3,226,266,762,397,899,821,056 x 5 µs = 510 million years
   * Dictionary storage space requirement: 3,226,266,762,397,899,821,056 x 54 bytes = 150 ZB (Zettabytes)
   * There is some ambiguity in what counts as a printable ASCII character but excluding the debatable characters there are 95 possible characters giving us 95^12 possible passwords which equals 540,360,087,662,636,962,890,625 or 540 sextillion.
   * Time to compute dictionary of passwords: 540,360,087,662,636,962,890,625 x 5 µs = 86 billion years.
   * Dictionary storage space requirement: 540,360,087,662,636,962,890,625 x 54 bytes = 24 YB (Yottabytes)

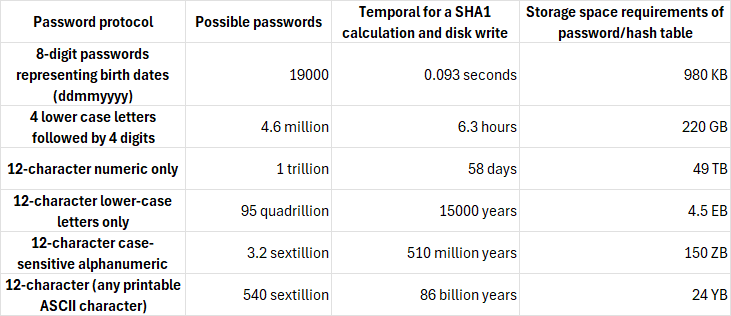


Table 1. Time & space complexity in relation to password protocol.

## Conclusion

The calculations show quite clearly that for a given password protocol any increase in its complexity will produce exponentially greater quantities of possible passwords and thus make brute-force attacks as well as dictionary attacks take significantly longer, additionally, the storage requirements to pre-compute a full hash table quickly become inconceivable. Due to this it makes sense for users to create passwords that are both long and use as diverse a set of characters as available and it stands to reason that attackers should focus on a smaller subset of potential passwords while, conversely focusing on as great a population of targets as possible.

# References

Brassard, G., HØyer, P., & Tapp, A. (1998). Quantum cryptanalysis of hash and claw-free functions. *Latin American Symposium on Theoretical Informatics*, (pp. 163-169). Campinas, Brazil. Retrieved September 23, 2024, from https://link.springer.com/chapter/10.1007/BFb0054319

Chailloux, A., Naya-Plasencia, M., & Schrottenloher, A. (2017). An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography. *International Conference on the Theory and Application of Cryptology and Information Security.* *2017*, pp. 211-240. Hong Kong: IACR. Retrieved September 23, 2024, from https://eprint.iacr.org/2017/847.pdf

Chaves, R., Sousa, L., Sklavos, N., Fournaris, A., Kalogeridou, G., Kitsos, P., & Sheikh, F. (2016). Secure Hashing: SHA-1, SHA-2, and SHA-3. In *Circuits and Systems for Security and Privacy* (pp. 81-106). CRC Press. Retrieved September 21, 2024, from https://www.researchgate.net/publication/303590313\_Secure\_Hashing\_SHA-1\_SHA-2\_and\_SHA-3

Cherckesova, L. V., Safaryan, O. A., Lyashenko, N. G., & Korochentsev, D. A. (2022). Developing a New Collision-Resistant Hashing Algorithm. *Mathematics, 10*(15), 2769. Retrieved September 20, 2024, from https://doi.org/10.3390/math10152769

Chi, L., & Zhu, X. (2017). Hashing Techniques: A Survey and Taxonomy. *ACM Computing Surveys (CSUR), 50*(1), 1-36. Retrieved September 21, 2024, from https://www.cse.fau.edu/~xqzhu/papers/ACS.Chi.2017.Hashing.pdf

Klíma, V. (2006). Tunnels in Hash Functions: MD5 Collisions Within a Minute. *IACR Cryptology ePrint Archive, 2006*, 105. Retrieved September 21, 2024, from https://eprint.iacr.org/2006/105.pdf

Kushwaha, P., & Shirsat, G. (2024). Comprehensive Review of the Security Flaws of Hashing Algorithms. *Journal of Emerging Technologies and Innovative Research, 11*(2), a195-a198. Retrieved September 20, 2024, from https://www.jetir.org/papers/JETIR2402024.pdf

Lee, D., & Hong, S. (2024). Improved Quantum Rebound Attacks on Double Block Length Hashing with Round-Reduced AES-256 and ARIA-256. *ACR Transactions on Symmetric Cryptology, 2024*(3), 238-265. Retrieved September 23, 2024, from https://tosc.iacr.org/plugins/generic/pdfJsViewer/pdf.js/web/viewer.html?file=https%3A%2F%2Ftosc.iacr.org%2Findex.php%2FToSC%2Farticle%2Fdownload%2F11816%2F11321%2F12860

Loebenberger, D., Gazdag, S.-L., Herzinger, D., Hirsch, E., Näther, C., & Steghöfer, J.-P. (2024). Formalizing the Cryptographic Migration Problem. Retrieved September 21, 2024, from https://researchgate.net/publication/383060202\_Formalizing\_the\_Cryptographic\_Migration\_Problem

Schneider, J., & Smalley, I. (2024, August 5). *What is quantum computing*. Retrieved September 23, 2024, from IBM Website: https://www.ibm.com/topics/quantum-computing

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017). The First Collision for Full SHA-1. *Annual International Cryptology Conference*, (pp. 570-596). Retrieved September 21, 2024, from https://www.researchgate.net/publication/318750485\_The\_First\_Collision\_for\_Full\_SHA-1#fullTextFileContent

Ullah, F., & Pun, C.-M. (2023). Deep self-learning based dynamic secret key generation for novel secure and efficient hashing algorithm. *Information Sciences, 629*, 488-501. Retrieved September 24, 2024, from https://www.sciencedirect.com/science/article/pii/S0020025523001780

Wang, X., Feng, D., Lai, X., & Yu, H. (2004). Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. *IACR Cryptology ePrint Archive, 2004*, 199. Retrieved September 21, 2024, from https://eprint.iacr.org/2004/199.pdf