

**A SECURE MODEL FOR STUDENT RESULTS VERIFICATION USING
SALTED HASH FUNCTIONS**

BY:

2021-01-03178

**A DISSERTATION SUBMITTED TO THE DIRECTORATE OF HIGHER
DEGREES AND RESEARCH (DHDR) IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE
IN INFORMATION TECHNOLOGY OF KAMPALA INTERNATIONAL
UNIVERSITY.**

NOVEMBER 2023

DECLARATION

I hereby declare that this thesis is entirely my work, and it does not breach any law of Copyright and that it has never been submitted to any academic institution for any academic award.

APPROVAL

I certify that this work has been done under our supervision and guidance. It is now ready for submission with our approval.

DEDICATIONS

To my parents, whose unwavering love, support, and encouragement have been the guiding light throughout my academic journey. Your sacrifices and belief in my abilities have inspired me to reach new heights and pursue my dreams with determination. This thesis is a tribute to your endless dedication and the values you instilled in me.

I dedicate this work to my mentors and professors, whose guidance and wisdom have shaped my intellect and fostered my passion for knowledge. Your invaluable insights and teachings have enriched my understanding and motivated me to excel in my field of study.

To my friends and loved ones, for standing by me with patience and understanding, offering a source of strength and laughter in both challenging and joyful times. Your unwavering support has been a constant reminder of the importance of genuine connections and shared experiences.

Finally, I dedicate this thesis to the pursuit of progress and the advancement of human understanding. May this work contribute, in its own small way, to the broader collective efforts aimed at making a positive impact on our world.

ACKNOWLEDGEMENT

I extend my deepest gratitude to my supervisors, whose guidance, expertise, and unwavering support have been instrumental in the successful completion of this thesis. Your insightful feedback, constructive criticism, and dedication to my academic growth have shaped this work and enriched my learning experience.

I would like to express my sincere appreciation to the faculty and staff of Kampala International University for providing a conducive environment for research and learning. The valuable resources, facilities, and opportunities extended to me, have significantly contributed to the quality of this thesis.

I am immensely thankful to my family for their constant encouragement, patience, and belief in my abilities. Your unconditional love and sacrifices have been my driving force and source of inspiration throughout this journey.

I also want to acknowledge my friends and colleagues for their camaraderie and support, whether it was through engaging discussions, shared challenges, or moments of celebration. Your presence has added a meaningful dimension to my academic pursuits.

My gratitude extends to the participants who generously contributed their time and insights to this research, without whom this work would not have been possible. Your willingness to share your experiences has enriched the depth and scope of this study.

Lastly, I would like to acknowledge all those whose work, whether directly cited or indirectly influential, has contributed to shaping my understanding and approach. The collective knowledge of scholars and researchers in the field has been a guiding beacon in navigating the complexities of this research.

In conclusion, this thesis stands as a testament to the collective efforts, support, and contributions of numerous individuals who have played a significant role in shaping my academic journey. Thank you all for your invaluable contributions.

ABSTRACT

The security and integrity of examination results are paramount in educational institutions, where maintaining confidentiality and ensuring the accuracy of student grades are essential. This dissertation explores the use of salted hash functions as a means to enhance the security of examination results. The primary objective of this study is to investigate the effectiveness, practicality, and implications of incorporating salted hash functions into result management systems.

The research encompasses a comprehensive literature review, the development of a prototype system, and an evaluation of its performance. The literature review examines existing systems and approaches in securing examination results, emphasizing the benefits and challenges associated with the use of salted hash functions. Drawing from this review, a prototype system is developed, integrating strong encryption algorithms and unique salts for each student record.

During the evaluation phase, different types of hashed data were manipulated with a python script that was used to try and reverse hash the values stored in the database. The results show that the hashed values which did not include a salt were easily cracked hence revealing the original data while those values that were hashed with a salt did not reveal the original data.

The conclusions stated that the developed model was effective at securing the results data stored in the database. Recommendations were provided for educational institutions seeking to implement similar systems. These recommendations include using hashing algorithms like Secure hash Algorithm (SHA) 256 to verify the integrity of the data stored in the database and the use of advanced encryption techniques to add another layer of security on the data. The dissertation also identifies areas for future research, such as exploring advanced encryption algorithms.

TABLE OF CONTENTS

DECLARATION.....	ii
APPROVAL	iii
DEDICATIONS	iv
ACKNOWLEDGEMENT	v
ABSTRACT	vi
LIST OF ACRONYMS	x
LIST OF FIGURES	xii
LIST OF TABLES	xiii
CHAPTER ONE: INTRODUCTION	1
1.1 BACKGROUND	1
1.2 STATEMENT OF THE PROBLEM	2
1.3 OBJECTIVES OF THE STUDY	2
1.3.1 GENERAL OBJECTIVE.....	2
1.3.2 SPECIFIC OBJECTIVES	2
1.3.3 RESEARCH QUESTIONS.....	2
1.4 SCOPE OF THE STUDY	3
1.4.1 CONTENT SCOPE	3
1.4.2 GEOGRAPHIC SCOPE	3
1.4.3 TIME SCOPE	3
1.4.4 THEORETICAL SCOPE	3
1.5 SIGNIFICANCE OF THE STUDY	4
1.6 DEFINITION OF KEY TERMS	4
CHAPTER TWO: LITERATURE REVIEW	6
2.1 INTRODUCTION.....	6
2.2 RELATED STUDIES	17
2.3 GAPS IDENTIFIED	26
2.4 PROPOSED MODELL FOR RESULTS VERIFICATION.....	27
CHAPTER THREE: METHODOLOGY	31
3.1 INTRODUCTION.....	31

3.2 RESEARCH APPROACH	31
3.3 TARGET POPULATION.....	31
3.4 SAMPLING TECHNIQUES	32
3.5 SAMPLE SIZE	32
3.6 DATA COLLECTION METHODS	33
3.6.1 DATA COLLECTION OF THE EXISTING SYSTEM	33
3.6.2 DATA COLLECTION METHODS USED DURING THE DEVELOPMENT OF THE MODEL.	33
3.6.3 DATA COLLECTION METHODS USED FOR THE EVALUATION	34
3.7 ANALYSIS OF COLLECTED DATA	34
3.7.1 DATA ANALYSIS METHODS USED WHEN INVESTIGATING THE CURRENT SYSTEM	34
3.7.2 DATA ANALYSIS METHODS USED WHEN DEVELOPING THE MODEL	34
3.7.3 DATA ANALYSIS METHODS USED FOR EVALUATION	34
3.8 METHOD FOR DEVELOPING A SECURE MODEL	35
CHAPTER FOUR: DATA PRESENTATION, ANALYSIS AND FINDINGS	38
4.1 INTRODUCTION	38
4.2 Data Presentation and Analysis of the Findings	38
4.2.1 Objective One: To Investigate the current information systems for storing and managing students' results	38
4.2.2: Objective Two: To develop a secure model that uses salted hashed functions to verify student's marks	43
SYSTEM PROTOTYPE OPERATION	48
4.2.3 Objective Three: To Evaluate the secure model	51
4.3 BENEFITS OF THE DEVELOPED PROTOTYPE.....	55
4.4 CHALLENGES FACED BY THE DEVELOPED PROTOTYPE	56
4.5 COMPARISON OF THE DEVELOPED PROTOTYPE WITH THE EXISTING SYSTEMS DISCUSSED IN THE LITERATURE REVIEW	57
CHAPTER FIVE: DISCUSSION, CONCLUSION AND RECOMMENDATIONS	58
5.1 INTRODUCTION	58
5.2 DISCUSSION	58
5.3 CONCLUSION	61

5.4 RECOMMENDATION	63
5.5 CONTRIBUTIONS TO KNOWLEDGE.....	64
5.6 AREAS FOR FUTURE RESEARCH	65
APPENDICES	69
APPENDIX A	69
APPENDIX B	77
APPENDIX C	83

LIST OF ACRONYMS

AES	- Advanced Encryption Standard
API	- Application Programming Interface
CIA	- Confidentiality, Integrity, Availability
DBMS	- Database Management System
DEC	- Digital Encryption Standard
DDOS	- Distributed Denial of Service
HMAC	- Hash-based Message Authentication Code
HTTPS	- Hypertext Transfer Protocol Secure
IDE	- Integrated Development Environment
IP	- Internet Protocol
IT	- Information Technology
KIU	- Kampala International University
LMS	- Learning Management System
MD	- Message Digest
MFA	- Multi-Factor Authentication
MYSQL	- My Structured Query Language
NIST	- National Institute of Standards and Technology
NSA	- National Security Agency
NSO	- National Statistics Office
OOD	- Object-Oriented Design
OTP	- One-Time Password
PHP	- Hypertext Preprocessor
PNG	- Portable Network Graphics

PKI	- Public Key Infrastructure
SC	- System Control
SHA	- Secure Hash Algorithm
SPSS	- Statistical Package for the Social Sciences
SQL	- Structured Query Language
SSL	- Secure Sockets Layer
UDP	- User Datagram Protocol
UI	- User Interface
UML	- Unified Modeling Language

LIST OF FIGURES

Figure 2.1: A model for TEVETA Examination results verification system.....	21
Figure 2.2: Smart contract implementation of university examination of data management.....	24
Figure 2.3: A model for multifactor authentication scheme for online examinations.....	25
Figure 2.4: The Proposed model for results verification.....	28
Figure 4.1: The figure above shows the process of storing the results in databases.....	44
Figure 4.2: The figure above shows the process of retrieving the results from the database.....	45
Figure 4.3: The figure below shows the database storing encrypted marks and salted hashed values.....	47
Figure 4.4: The figure above shows the encryption keys and salt values stored in a separate database.....	48
Figure 4.5: Interface for inputting marks.....	49
Figure 4.6: Interface showing all verified results which are valid.....	50
Figure 4.8: Showing the results after reverse hashing.....	52
Figure 4.9: Showing results of failure after revers hashing.....	54

LIST OF TABLES

Table 2.1: The table below summarizes the reviewed systems and the gaps identified.....	27
Table 3.1: The sample size of the respondents that was used for data collection.....	32
Table 3.2. Summary Table for The Methodology Used to Achieve Each Objective.....	37
Table 4.1: The table below shows the Staff position of the respondents.....	38
Table 4.2: The table below shows responses regarding if marks are changed in the system.....	39
Table 4.3: The table below shows how often staff members attend to concerns of missing marks.....	40
Table 4.4: The table below shows some of the reasons why students submit complaints for missing marks.....	41
Table 4.5: Results of investigations of the current information systems.....	42
Table 4.6.: Shows the results that were obtained after try to reverse hashed data without salt.....	53
Table 4.7: Showing results that were obtained after revers hashing data with a salt.....	55

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND

Examinations are vital procedures in determining the performance of students in learning institutions. Thus, examination records form part of the body of records generated by universities that need proper management. Poor management of examination records would easily tarnish the university's image to the public as this would affect university students who wish to graduate with authentic academic degree certificates and transcripts that are acceptable in the job market (Samuel, 2021).

Educational establishments are required to keep records of students' assessments for a significant amount of time. These records are utilized not only to verify students' academic achievements but also for in-depth analysis to introduce more productive and effective improvements. It is crucial to safeguard such data in a secure environment and ensure that it's not altered or deleted (Ashis, 2021).

There has been a recent increase in the utilization of web and mobile applications for distributing students' examination results in educational institutions. However, this has led to security concerns regarding how to maintain the confidentiality, integrity, and authenticity of these results. The reason for this is that web and mobile applications are vulnerable to cyber-attacks, which can compromise the authenticity of the data (Lister, 2019).

Andrew Ssenyonga (2017) noted that Makerere University faced challenges when some students had issues with their grades during an academic audit aimed at recalling undeserved degrees. The audit required a thorough review of the marks of students over several years to determine if any certificates had been obtained through forged or altered results, which resulted in the affected students being unable to graduate on schedule.

To prevent the disappointment of students failing to graduate on time, it is crucial to exercise more caution in managing examination records throughout the university student's lifecycle at every stage (Samuel, 2021).

1.2 STATEMENT OF THE PROBLEM

The convention of storing student examination results in a less secure electronic system is often characterized by highly fraudulent practices which include invalid results due to unauthorized amendments and unauthorized disclosure (Lister, 2019).

Currently, the university staff has to verify the marks of every student before they appear on the graduation list. Verification is a very tiresome and slow process because it involves checking every single mark stored in the system to make sure that there are no invalid results. Students who are found to have invalid results are inconvenienced a lot to an extent that some miss graduation and they are advised to start afresh the process of clearing for the next graduation after their marks are sorted.

Thus, there is a need to create a model for students' results verification that will use salted hash functions that will ensure data integrity.

1.3 OBJECTIVES OF THE STUDY

1.3.1 GENERAL OBJECTIVE

The general objective of this research is to develop a model for students' results verification.

1.3.2 SPECIFIC OBJECTIVES

- i. To investigate the current information system that is used to store and manage student results at Kampala International University.
- ii. To develop a secure model that uses salted hashed functions to verify students' results.
- iii. To evaluate the Secure model for student's results verification.

1.3.3 RESEARCH QUESTIONS

- i. What are the strengths and weaknesses of the existing system?
- ii. How does the developed secure model that uses salted hashed functions works to solve the problems?
- iii. What are the results of the evaluation of the secure model that uses salted hashed functions?

1.4 SCOPE OF THE STUDY

1.4.1 CONTENT SCOPE

The study focused on developing a Secure model that used salted hashed functions for the verification of students' results through the use of hash functions.

1.4.2 GEOGRAPHIC SCOPE

The Study was conducted at Kampala International University's main campus since it was an institute of higher learning that used an information system to manage students' results, and there had been cases where some students failed to graduate because of invalid results.

1.4.3 TIME SCOPE

The estimated period that was used to carry out the research was eight months, starting from July 2022 to August 2023, as it involved an investigation of the then-current system used at the university, data collection from the different respondents, and data analysis.

1.4.4 THEORETICAL SCOPE

The theoretical scope of this thesis revolves around a thorough exploration of cryptographic techniques and information security principles within the context of the Technical, Entrepreneurial and Vocational Education and Training Authority (TEVETA) system. This encompasses an in-depth analysis of encryption methods, hashing algorithms, and data integrity mechanisms, specifically tailored to enhance the security of examination results and student records stored within the Technical, Entrepreneurial and Vocational Education and Training Authority (TEVETA) system.

The theoretical scope extends to comprehending the concept of salted hash functions and their role in augmenting the protection of sensitive data. It involves delving into the theoretical foundations of database management, data storage, and retrieval, with a specific focus on strategies that ensure the confidentiality, integrity, and authenticity of crucial educational information.

1.5 SIGNIFICANCE OF THE STUDY

- I. This study will enable the researcher to create a secure model that will be used in the verification of students to ensure that they are authentic.
- II. The findings of this study will guide the university to initiate and carry out policy reforms and make appropriate changes in the examination results management to enhance the effective delivery of student results.
- III. This study will benefit and motivate other researchers by encouraging further research in areas of information security management to improve the security, storage and dissemination of examination results in other educational institutes.

1.6 DEFINITION OF KEY TERMS

Salted Hash Functions:

Salted hash functions are cryptographic algorithms that combine a password or data input with a randomly generated salt value before applying a one-way hashing function.

Salt value:

A salt value is a random or unique piece of data that is added to the input of a cryptographic hash function before the hashing process occurs.

Hash function:

A hash function is a mathematical algorithm that takes an input (or 'message') and produces a fixed-size string of characters, which is typically a sequence of numbers and letters.

Result Management Systems

Result management systems refer to software applications or platforms designed to store, manage, and process examination results in educational institutions.

Data Security

Data security refers to the protection of data from unauthorized access, use, disclosure, alteration, or destruction.

Cryptographic Algorithms

Cryptographic algorithms are mathematical functions used to transform data into a secure and unreadable format.

Confidentiality

Confidentiality refers to the principle of ensuring that data or information is accessible only to authorized individuals or entities.

Integrity

Integrity refers to the assurance that data remains unchanged and uncorrupted during storage, transmission, or processing.

Result Verification

Result verification refers to the process of confirming the integrity and authenticity of examination results.

Encryption

Encryption is the process of converting plain text or data into an unreadable form (cipher text) using an encryption algorithm and a secret key.

User Authentication

User authentication is the process of verifying the identity of individuals accessing a system or platform.

CHAPTER TWO: LITERATURE REVIEW

2.1 INTRODUCTION

This chapter presents a comprehensive literature review on the topic of securing examination results using salted hash functions. The literature review serves as a foundation for understanding the current state of research, existing systems, and best practices in the field of data security and result management in educational institutions. By critically analyzing relevant scholarly articles, research papers, and industry reports, this chapter aims to provide a comprehensive overview of the theoretical and practical aspects surrounding the use of salted hash functions in securing examination results.

2.1.1 SALTED HASH FUNCTION

All modern systems should in one way or another be able to protect sensitive information from being modified without authorization from the right personnel. Information can be compromised if access is granted to the wrong individuals. There are different methods which can be used to make sure that information in the database has not been tampered with. One of the most popular methods is the use of hashing. Hashing can be used to validate that the current information stored in the database is the original and has not been tampered with.

"Salted" refers to the practice of adding a random or unique value (called a "salt") to data before applying a hash function. This technique is commonly used in cryptography and security to enhance the security of hashed values, particularly in password storage.

A hash function is a mathematical algorithm that takes an input (or 'message') and produces a fixed-size string of characters, which is typically a sequence of numbers and letters. The output, often referred to as the hash value or hash code, is unique to the specific input data. Hash functions are commonly used in various fields of computer science and cryptography for data integrity verification, digital signatures, password storage, and more.

Hashing is an essential component of any conventional security plan aimed at preventing sensitive and confidential information from falling into unauthorized hands. Generating a robust hash value is the initial step in ensuring the security of sensitive user data (Chowdhury, 2017). The goal is to increase data security and to add to the body of knowledge regarding information security, including salting methods. The foundation of the salting strategy, which protects against pre-computed dictionary attacks, one method to strengthen the security of a password before hashing it is by adding a random sequence of letters and numbers, also known as "salt," to the beginning or end of the password (Boonkrong, 2016).

Hashing is a one way function which is used in the validation of information stored in the database. During the hashing process, plain text is transformed into a string of fixed lengths of random values which carry no meaning when read by someone. This process is known as a message digest. Unlike encryption, it is impossible to reverse a hash back to plain text which makes it a better option for validating information to make sure it was not changed in the database. The idea of a trap door is used to build hash functions. Hash functions are unidirectional functions and one-way functions (Martinez, 2020).

When a hash function is applied to a variable-sized message that belongs to a specific set of messages, M , it can produce a fixed, predetermined output of bit size x . Since the hash function compresses a message of any length to a set of x bits, the number of possible hashes is significantly lower than the number of different input messages. (Handschuh, 2018). Two security requirements for hash functions are one-way and collision resistance. The former ensures that the underlying function is not invertible, while the latter indicates that it is impossible to locate two inputs that share the exact hash value (Wang, 2018).

Hashing is commonly employed to enhance the security of the authentication process. Authentication is a procedure that ensures an item's genuineness, verifiability, and reliability, and it instills a strong belief that the transmission, message, or sender of the item is trustworthy. It verifies that the input provided by the user is authorized to be entered into the system and originates from a dependable source. The goal of hashing is to ensure the integrity of information in case of a breach, but not to safeguard data (Brown, 2015).

One of the principles required to guarantee a system's security is authentication. Another principle required to support the CIA Trio is authentication along with accountability. The security concept known as the CIA Trio, which stands for Confidentiality, Integrity, and Availability, is highly well-known (Quadri, 2016).

The CIA triad is the basic model of information security, and while there are several other models that share some of its features, authentication is a crucial step in addition to protecting data from unauthorized users. It also maintains the integrity of the information (Brandão, 2018). To minimize the risk of receiving manipulated data from a hacker, the authentication process utilizes algorithms and hashing methods. Hashing algorithms are used in the authentication process for tasks such as login (password) authentication, verifying the authenticity of documents, storing passwords, generating keys, creating pseudo-random numbers, authenticating tokens on applications in distributed databases, digital signatures, and more (Sahu, 2018).

A hash is employed in information systems for the authentication login procedure. Using specific hashing techniques, data is modified, resulting in unique characters that are later saved in the database. The two most used hashing algorithms are SHA1 and MD5 (Pittalia, 2019).

2.1.2 MESSAGE DIGEST (MD) AND SECURE HASH ALGORITHM (SHA)

A message digest, also known as a hash value or hash code, is a fixed-size numeric representation of data generated by a hash function. It is commonly used in various cryptographic applications, including data integrity verification and password storage (Kasgar, 2013).

Secure Hash Algorithm (SHA) is a widely used family of cryptographic hash functions.

A Secure Hash Algorithm (SHA) is a family of cryptographic hash functions designed to take an input (or 'message') and produce a fixed-size hash value, typically a sequence of numbers and letters. The primary purpose of a secure hash algorithm is to ensure data integrity, verify the authenticity of messages, and provide a one-way function that is computationally infeasible to reverse (Kasgar, 2013).

One of the most commonly known members of the SHA family is SHA-256. Here's how a secure hash algorithm like SHA-256 works:

Input Data: The data (message) that needs to be hashed is provided as input to the SHA algorithm.

Pre-processing: The input data is pre-processed to ensure a consistent format. This may involve padding the data to a specific length, so it can be divided into blocks for processing (Kasgar, 2013).

Message Digest Computation: The algorithm processes the input data in blocks through a series of logical and mathematical operations. Each block is transformed, and the algorithm maintains an internal state.

Compression Function: The compression function takes the current internal state and processes the current block of data. This process involves bitwise operations, modular arithmetic, and logical functions.

Iteration: The compression function and internal state are updated repeatedly for each block of data. The number of iterations depends on the specific SHA variant being used.

Finalization: Once all blocks have been processed, the algorithm performs final operations on the internal state to produce the hash value (Kasgar, 2013).

Secure hash algorithms are widely used in various applications:

Hash functions are used to verify that data hasn't been tampered with. By comparing hash values before and after transmission, one can determine if the data has been altered.

Hash functions are used in digital signatures to ensure that a message or document has not been modified after it was signed.

Hash functions are used to securely store passwords. When a user logs in, their password is hashed and compared to the stored hash value, without the need to store the actual password (Kasgar, 2013).

Hash functions are used in various cryptographic protocols and constructions, such as HMAC (Hash-based Message Authentication Code) and digital certificates (Kasgar, 2013).

It was developed by the National Security Agency (NSA) in the United States and has become a standard hash function in many security applications. The SHA family includes various iterations, such as SHA-1, SHA-256, SHA-384, and SHA-512, with the numbers representing the digest sizes they produce (in bits) (Handschuh, 2018).

A message digest (MD) is a piece of random value that is generated algorithmically from a system that specifically identifies that document. The message digest will change if the document changes (Mohanty, Sarangi, & Bishi, 2010). The mathematical operation that can be performed on a string of varying lengths is described in Message Digest. In simple terms, the number "5" signifies that MD5 replaced MD4. MD5 is essentially a checksum used to verify the integrity of a file or string, and it is commonly used for this purpose (Kasgar, 2013). There are many flaws in the MD5 algorithm, including its susceptibility to various exploits including rainbow tables, dictionary attacks, birthday attacks, etc. (Bhandari, 2017).

The National Security Agency created a collection of cryptographic hash functions called SHA. The SHA family of hash functions is susceptible to a weakness where two different input values may produce the same output value during the computation, which is referred to as a hash collision. Therefore, it is critical to have strong diffusion in the hash function to reduce the probability of such collisions occurring. As a result, each round's output will be dispersed and won't match up exactly with the output of the following stages (Mirvaziri, 2017).

Message Digest 5 (MD5), Message Digest 2 (MD2), Message Digest 4 (MD4), and Secure Hash Algorithm are some common hashing methods (SHA). But MD4, MD2, and MD5 algorithms are no longer considered appropriate hashing methods (Lister, 2019).

Due to numerous flaws in MD5 and SHA1, attackers can quickly obtain the system information by knowing the hash value. As time goes on, a new hash algorithm that offers greater security than the preceding method emerges; examples include SHA2, SHA3, BCrypt, and others. Without a doubt, as technology advances, new algorithms will also reveal attackers' vulnerabilities (Sutriman, 2019).

The incorporation of salt into the plaintext data before hashing can enhance the use of hashes in the authentication process (Handschuh, 2018).

Numerous mathematicians have written articles describing the shortcomings of these algorithms. Modern crypt analytic attacks showed that the SHA-1 algorithm has flaws, which prompted the development of SHA-2, which now has four variants: SHA-224, SHA-256, SHA-384, and SHA-512 (Lister, 2019).

52 out of 64 rounds of SHA-256 and 57 and 80 rounds of SHA-512 pre-image resistance were broken by attacks from 2011, respectively (Khovratovich, Rechberger, & Savelieva, 2015). Additionally, vulnerable to length extensions attacks are SHA-256 and SHA-512; length attacks on SHA-224 and SHA-384 can be conducted by guessing the secret portion of the state. The newest algorithm in the Secure Hash Algorithm family, SHA-3 (Secure Hash Algorithm 3), was released by NIST in 2015.

An updated version of SHA-1 was used in place of SHA-0 since it had a "paramount defect" and was deleted shortly after release. In a manner similar to MD5, SHA-1 generates a 160-bit message digest. The National Security Agency created this to be a component of the digital signature algorithm. After 2010, it was no longer usable for the majority of cryptographic purposes due to the working's cryptographic flaws (Surbhi, 2017). Additionally, developed by the NSA was SHA-2. The word size of the two members of the same hash function family, SHA-256 and SHA-512, which have differing block sizes, varies. While SHA-512 utilizes 64-bit words, SHA-256 employs 32-bit words (Surbhi, 2017). Following a public competition among non-NSA designers, SHA-3 was suggested in 2012. It supports the same hash lengths as SHA-2 and has a very different internal structure from the other members of the SHA family (Surbhi, 2017).

A hash function must meet certain requirements, such as preimage resistance, second preimage resistance, and collision resistance, in order to be considered secure (Wang, 2016). According to the concept of preimage resistance, it is computationally unfeasible to discover an input that produces the same output as a specified input. In contrast, second preimage resistance indicates that it is challenging to locate an input that generates the same output as a given input. Meanwhile, collision resistance suggests that it is tough to find two inputs that result in the same output. (Wang, 2016). Attackers are launching brute force attempts on SHA-256 in response to these three characteristics. The majority of contemporary hash function attacks concentrate on locating

collisions, and the birthday attack is the most well-known generic technique for doing so (Sharma, 2018).

2.1.3 REASONS WHY (MESSAGE DIGEST 5) MD5 WAS SUPERSEDED BY (SECURE HASH ALGORITHM) SHA

To securely store sensitive information like passwords or credit card numbers in databases like MySQL, it is common to use a hashing algorithm like MD5. Typically, MD5 hashes shorter character sequences to protect sensitive data (Surbhi, 2017).

Any time two provided inputs yield the same hash output, MD5 produces collisions. Furthermore, MD5 is subject to destructive collision attacks. Due to these threats, MD5 practically offers no security against collisions (Surbhi, 2017).

Collisions in SHA were are not frequently found. For SHA-1, no collision has yet been discovered. SHA-256, which is currently unbroken and significantly “larger” (has many more operations than SHA-1 but with a comparable structure) (Surbhi, 2017).

In terms of hardware or software, MD5 cannot be applied at speeds greater than 256 Mbps or 86 Mbps, respectively, using current technology. While SHA can be implemented with current technology more quickly than it is now. SHA-1 that has been assembled optimized is generally quicker than MD5. The suggested authentication method for IPv6 is MD5, which should support 130 Mbps UDP-capable current networking technologies (Surbhi, 2017).

It seems like SHA1 is significantly more secure. Despite the fact that SHA1 has been the target of a few known attacks, these incidents pale in comparison to those against MD5 (Surbhi, 2017).

Instead of using MD5 or SHA1, which are vulnerable to known attacks, there are probably better modern hash functions available, such as SHA256. There are no known attacks of any practical relevance against those (Surbhi, 2017).

2.1.4 Reverse hashing

Reverse hashing, commonly known as "hash cracking" or "hash decryption," involves the attempt to recover the original input data (plaintext) from a provided hash value. Hash functions are intentionally designed to be one-way functions, which means that reversing the process to retrieve the original data from the hash is intended to be exceedingly challenging, if not practically impossible (Hornby, 2021).

Hashing is a process of converting input data of any size or length into a fixed-size value called a hash code or hash value. The input data can be of any type, such as text, files, or binary data. The hash function performs a mathematical calculation on the input data to produce a unique, fixed-length output (Handschuh, 2018).

Due to the several relatively simple methods for finding modifications that may be added to the end or beginning of a payload to make it look to be genuine, MD5 is sadly severely degraded in this aspect (Surbhi, 2017).

Recently found small weaknesses in this area affect SHA-1 as well, however they are less serious than the problems with MD5. Using SHA256, for example, has been even safer because there are presently no identified hash collision attacks against it (Surbhi, 2017).

The primary objective behind the utilization of hash functions is to ensure data security by irreversibly transforming input data into a fixed-size hash value. This attribute is especially critical in applications like password storage, where the safeguarding of original passwords is of paramount importance, even if the hash itself is compromised. The concept of reverse hashing seeks to subvert this security by operating in the opposite direction (Handschuh, 2018).

The process of reverse hashing typically involves the following methods:

Brute-Force Attack: This approach involves an attacker generating potential input data (plausible plaintext) and then hashing it using the same algorithm. The resultant hash is subsequently compared to the target hash. This cycle continues until a match is found, indicating that the attacker has successfully deciphered the hash and ascertained the original input data. Brute-force attacks are resource-intensive and time-consuming, particularly when dealing with robust hash functions.

Rainbow Tables: Rainbow tables consist of precomputed hash values and their corresponding inputs. Attackers employ these tables to rapidly search for hash values and identify corresponding input data. To counter rainbow table attacks, the practice of salting (introducing a random value before hashing) is often adopted (Mirvaziri, 2017).

Dictionary Attacks: In a dictionary attack, an attacker employs a list of commonly used passwords or phrases, hashes them, and then checks if the hash matches the target hash. If a match is found, the original input is deemed to be discovered. This technique proves effective when the original input is a weak or frequently used password (Handschuh, 2018).

GPU and Parallel Computing: Modern graphics processing units (GPUs) and parallel computing techniques have significantly accelerated the speed of hash cracking. Attackers can simultaneously employ multiple processing units to test a large volume of inputs (Surbhi, 2017).

It is essential to emphasize that the security of a hash function hinges on its resistance to reverse hashing attempts. Robust hash functions, such as those within the SHA-2 family (e.g., SHA-256), are deliberately engineered to withstand brute-force and other forms of attacks, rendering the process of reverse hashing exceedingly laborious and time-intensive. As computational capabilities advance over time, older hash functions may become more susceptible, underscoring the significance of staying updated with advancements in cryptography and employing the latest secure algorithms (Handschuh, 2018).

2.1.5 Application of Hash Functions

A hash function is a mathematical algorithm that takes an input (or "message") and processes it to produce a fixed-size output called a hash code or hash value. The primary purpose of a hash function is to efficiently map data of arbitrary size to a fixed-size value (Handschuh, 2018).

Hashing is a process of converting input data of any size or length into a fixed-size value called a hash code or hash value. The input data can be of any type, such as text, files, or binary data. The hash function performs a mathematical calculation on the input data to produce a unique, fixed-length output (Handschuh, 2018).

Hash Functions can be used to authenticate that an authorized person signed a document but rather not an imposter and this can be done through the use of digital signatures. It can be done by

comparing the current signature with previous signatures (Diro, 2017). If the signatures match, then the receiver can verify that they were not sent by an unauthorized entity (Puthal, Nepal, Ranjan, & Chen, 2019).

To ensure that files have not been modified, hash functions can be used (Raza, Seitz, Sitenkov, & Selander, 2016). Some websites provide a checksum for an individual to use and confirm that the file they downloaded has not be tempered with (Huang, 2016).

Hash Functions are used in systems that require password authentication to make sure that the user provides the right password during the log in process. When a user types a new password, its hashed and stored in the database and later the hashes values are verified against the current password that the user inputs to make sure that it's the right input (Sutriman, 2019).

Attacker can install root kits on the targeted machines which they can use to perform certain special attacks on the victims' computers. These rootkits can hide their presence on the target machines which makes them undetectable (Vucinic, 2015). These root kit can be found through a process known as signature-based discovery. It involves the use of antivirus applications to find traces of know rootkit signatures in the operating system (Shivraj, 2015).

Hashing functions are also used in block-chain technology to ensure data integrity. The use of hash functions in a block-chain can be divided into six primary areas, including pseudo random number generation (PNG), address creation, message digest in signatures, and bridge components (known as Fiat-Shamir mechanism) (Wang, Cryptographic primitives in blockchains, 2018). The Secure Hash Algorithms family of cryptographic hash functions, includes the most widely used hash function in the block-chain sector known as SHA-2, is a group of hashing algorithms (SHA) (Raikwar, 2019).

2.1.6 Types of Hashing Functions

Salt Hashing

Salting is when a sequence of characters is appended to information before its transformed into a hashed value. It makes our hashed values different from what they would be if they had been digested alone and as a result, protects information from brute force attacks. Prior to hashing, salt is typically added as a prefix or postfix to the plaintext (Sutriman, 2019).

There are different types of salts that we can use when hashing information and these include;

A fixed salt, which is a sequence of constant characters or bytes that we use for hashing every information. The salt is hidden and is considered to be an added security value. It is not advisable to use this type of salt because it can make the system vulnerable to attacks (Hornby, 2021).

A variable salt, is safer than a fixed salt because it offers random bytes which cannot easily be predictable. A variable salt is generated separately for each information being hashed and it allows each stored data to be decoupled from the others, creating stronger protection while improving safety for attacks driven against our database (Hornby, 2021). If two or more pieces of information of the same kind are stored in the database, they will have the same hashes. This becomes easy for the attacker to try to find out exactly what information is being hashed. We can prevent these attacks by making the hashes random so that when the same information is digested twice, the hashed information is different (Hornby, 2021).

We can make the hashes random by adding a random string, called a salt, to the information before it is hashed. In order to enhance the security of using hashes in the authentication process, the addition of salt to the plaintext data is recommended. This method results in a different string being generated for the hashed data every time. To verify the validity of the information, the salt is necessary, which is typically stored in the system database alongside the hash or as part of the hash string. This way, when a user enters their password, the system can add the salt to the plaintext password, generate a new hash, and compare it to the stored hash to determine if the user inputted the correct password (Hornby, 2021).

Adding a salt to the plaintext before hashing can enhance the security of using hashes in the authentication process. This changes the hashed data into a different string each time, and to verify the validity of the information, the salt is stored in the system database along with the hash or as

part of the hash string. The salt doesn't have to be kept secret, but randomizing the hashes makes certain attacks such as rainbow tables, lookup tables, and reverse lookup tables ineffective. If each user's password is hashed with a unique salt, the reverse lookup table attack won't work because the attacker wouldn't know what the salt is (Hornby, 2021).

One of the mistakes commonly made while using salt is the reuse of the same salt for multiple hashes or using a salt that is too brief. While adding salt to information can make them more secure, it does not eliminate the possibility that a hacker will be able to guess them using the produced values. Password and salt combinations were the starting point for the several sorts of password cracking programs that are currently circulating in cyberspace. Still vulnerable is prefixing or postfixing salt (Sutriman, 2019).

2.2 RELATED STUDIES

In order to increase the security of information systems used to disseminate exam results, numerous studies have been carried out.

2.2.1. Technical Education, Vocational and Entrepreneurship Training Authority (TEVETA) Examination Results Dissemination and Verification System

The Technical Education, Vocational and Entrepreneurship Training Authority (TEVETA) is a statutory body established by the Government of Zambia through the Technical Education, Vocational and Entrepreneurship Training Act No. 13 of 1998. TEVETA is responsible for the regulation, monitoring and coordination of technical education, vocational and entrepreneurship training in Zambia (Lister, 2019).

The purpose of this document is to examine the utilization of encryption and cryptographic hash functions as a means of enhancing the security of student test results, and ensuring that data is kept confidential, integral, and authentic during its storage and transmission. The implementation of such measures can help improve the accuracy and reliability of student test results (Lister, 2019). The system allowed secure storage and communication of results, as demonstrated by the prototype that was created from the model. This is largely due to the encryption and hashing that were used to store and distribute the findings (Lister, 2019).

The researchers created a prototype which enables students and other parties to access student test results through both mobile phones and the web. The web component was built using PHP, while the mobile component was built using Java. Both applications incorporated the AES encryption algorithm and SHA3-224 cryptographic hash function to achieve confidentiality, integrity, and authenticity in regards to the examination results. The prototype that was developed from this model proved to be a secure method for storing and transmitting test results (Lister, 2019).

Related Systems that were reviewed

The author reviewed a number of systems that were related to his study and some of the system that he reviewed include:

In Nigeria authors created a Short Message Service system that enables students at the University to send messages in order to be able to access their new and old results. But the short messages that were sent were not encrypted which made them susceptible to hacking (Lister, 2019).

Another system that was reviewed used both short messages and Emails to provide alert notifications about the availability of results. The problem with the system is that the results are transmitted and stored in the database when they are not encrypted which can make easy for hackers to access them (Lister, 2019). In Zambia a system was implemented for Examinations Council of Zambia that enabled university students to access them when they were available using short messages. The results were stored in plain text with no encryption (Lister, 2019).

System Modeling and Design

System modeling and design is a structured and systematic approach to developing a blueprint or representation of a system before it is implemented. It involves creating models and designs that capture the requirements, behavior, structure, and interactions of a system to guide its development or improvement (Lister, 2019).

Object-Oriented Design (OOD) is a software design approach that focuses on organizing and structuring software systems based on the concept of objects. It promotes the use of classes, objects, and their interactions to model and represent real-world entities and their relationships (Lister, 2019).

The author used an Object-Oriented Design to analyze the existing systems and develop a prototype.

The Unified Modeling Language (UML) is a standardized visual modeling language used in software engineering to represent, design, and document software systems. UML provides a set of graphical notations and a standardized syntax for representing different aspects of a system, including its structure, behavior, interactions, and architecture (Lister, 2019).

The author used Unified Modelling Languages like the class diagrams to capture the static behavior of the system. The use case diagrams were used to capture the dynamic behavior of the system (Lister, 2019).

The system was implemented using PHP for the web application. It also used a MySQL database to store the information. The short message system was developed using java programming language. It allows a mobile phone to send and receive requests. Sending of short messages was expensive to implement and so an Unstructured Supplementary Service Data (USSD) simulator was created by the researcher to enable students to forward requests to the system in order to receive the information they need (Lister, 2019).

Unstructured Supplementary Service Data (USSD) is a communication protocol used by mobile network operators to provide interactive services to mobile phone users. It allows users to access various services and information by typing specific codes (known as USSD codes) into their mobile devices (Lister, 2019).

When the students' marks are altered after they are published, the systems will detect and alert the users that the marks have been altered. This will happen because the hashes of the changed marks will be different from the hashes of the real marks (Lister, 2019).

How the system work

In the first phase for registration, the students provide their details like student name and registration number and it is stored in the system. These details are used to identify each student. In the second phase, the exams entry officer logs into the system but before they are given access, they need to first be authorized to log in and later they can enter the marks into the system. When

the mark is entered into the system, it is encrypted and at the same time the data is hashed with SHA 256 hashing algorithm and later stored in the database.

In the third phase another office in charge of examination results verification and publishing will also have to first login before the system can authorize him or her to verify results. In this stage the results are picked from the database and they are decrypted, then the decrypted data is rehashed again with SHA 256 and the hashed value is compared with the other hash value stored in the database. If the hash values match, then the results are published for the students to access.

In the final stage, the students can use their personal devices like the laptops, and smart phones to access the system and request for the display of their results. Before the results are displayed to the student, the system will have to verify the integrity of the data by decrypting the encrypted marks, then a hash value will be generated to the deciphered text. The new hashed value is compared against the already existing value in the database. If the two values match, then the marks are displayed to the user, if they don't match, the system will return an error message alerting the user that the marks have been tampered with.

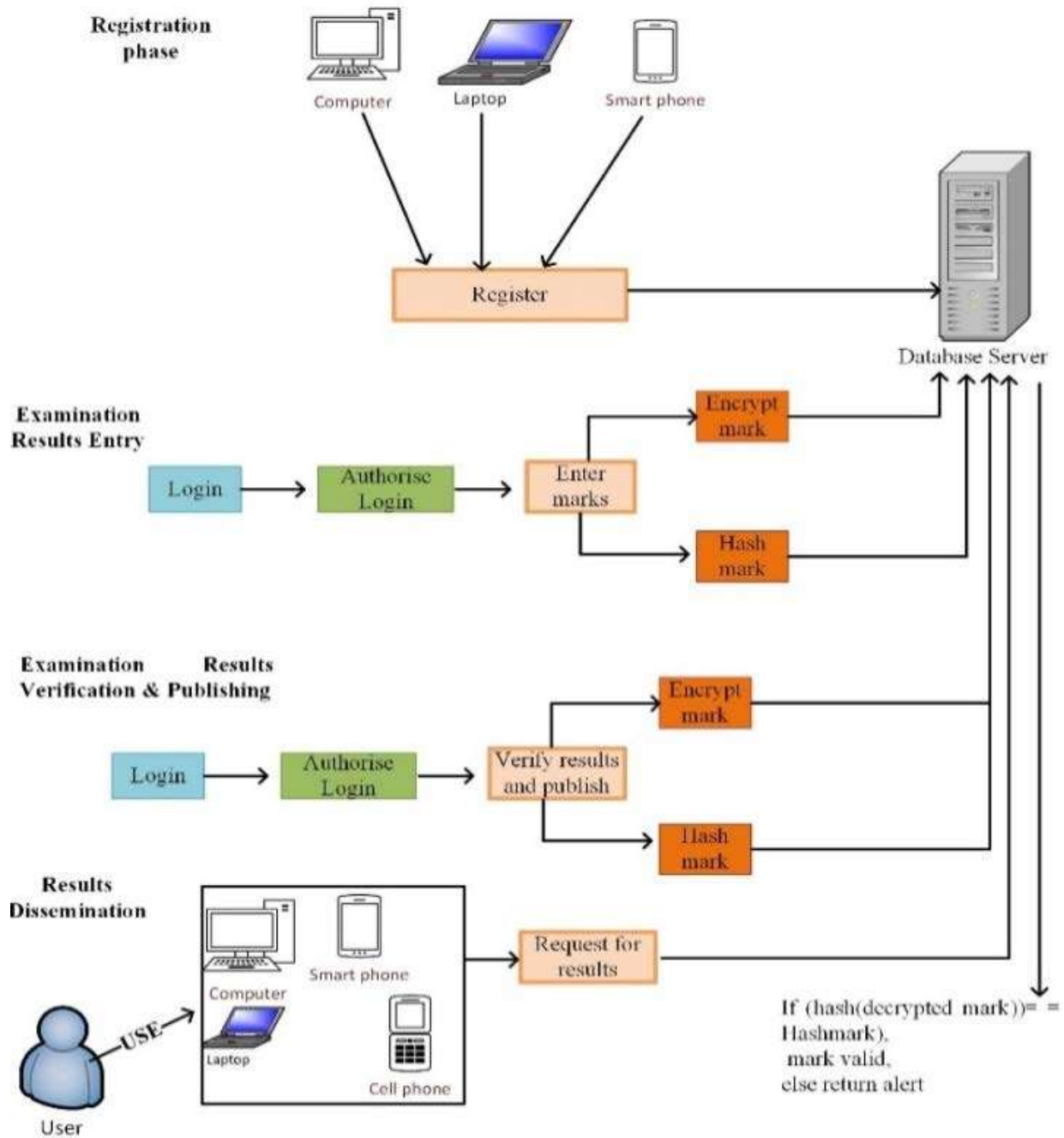


Figure 2.1: A model for TEVETA Examination results verification system

Testing and Validation

After developing the system, the researcher used acceptance testing whereby he provided the developed prototype to a number of students and staff to interact with the system. Feedback was provided and the researcher improved on the system until it was fully accepted by the users.

2.2.2. A Blockchain-Based Smart Contract Towards Developing Secured University Examination System

Blockchain is a decentralized, distributed ledger technology that allows multiple parties to maintain a shared and immutable record of transactions or data in a transparent and secure manner. It is the underlying technology behind cryptocurrencies like Bitcoin, but its potential applications extend beyond digital currencies (Ashis, 2020).

A blockchain-based smart contract is a self-executing contract with the terms and conditions encoded within a blockchain. It combines the capabilities of blockchain technology with programmable logic to automate and enforce the execution of contractual agreements without the need for intermediaries (Ashis, 2020).

In these related studies, the model that was developed relied on the use of smart contracts to ensure that the students record where secured. A smart contract (SC) is a digital mechanism for negotiations between two or more anonymous parties in the absence of any reliable middlemen (Ashis, 2020). Smart contract is a type of technology that only be used in block chain systems. Once data is input into the system, it cannot be modified. This can be a good option because it ensures the integrity of the information. But if mistakes are discovered in the already existing data, there is no possible way of correcting it. This is one of the main limitations of using blockchain technologies to store information. The figure below shows a blockchain model that was designed to secure examination results for universities (Ashis, 2020).

The system was developed on a Hyperledger composer. The framework for the tool is a Hyperledger fabric. Fabric is a Distributed Ledger Technology shared among many individuals which makes it a decentralized network (Ashis, 2020).

Methodology for the proposed smart contract model

The system is separated into two different parts. The first section is a smart contract designed to meet existing challenges. In this section, there is a contract and the movement of information between the stockholders and the University to provide a trustworthy and secure system (Ashis, 2020).

Some of the related studies reviewed by the researcher include:

The author review a Blockchain Smart contract distributed Leave Management system developed using Solidity and Ethereum. Mobile Phones can be used as an IoT devices to access data records. The authors have claimed that the application ensures security and privacy by saving time and cost (Ashis, 2020).

The author also reviewed a Japan E portfolio which is a university admission system using smart contracts. Tokens like Study Document Information E-Portfolio Information and Qualification Information are used. A class of unique token ERC-721 over Ethereum blockchain is used. Security is taken care by suing a public key cryptosystem and common key sharing. Data falsification and loss of information are taken care in this application model to enhance security management (Ashis, 2021).

Development of the system

The operating system that was used is Ubuntu 16.04 LTS 64bit. Hyperledger Fabric 2.0 and Composer where installed. Docker Compose 1.13.0, and Docker Engine version 19.03.8, Node Js 8.17.0, git 2.9, npm 5.x visual studio editor and python 2.7 were used. The REST server provided by the Hyper Ledger Composer was used which provides a transparent Application Programming Interface. It is easier to implement the blockchain and integrate it with web and mobile applications (Ashis, 2021).

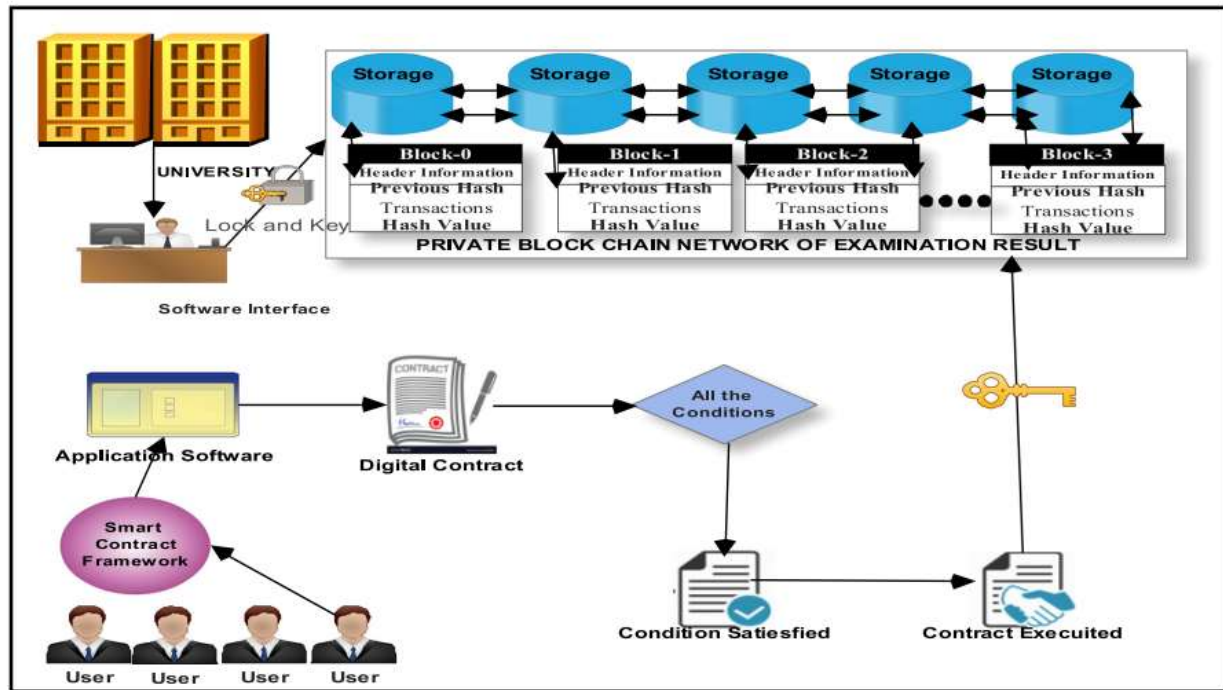


Figure 2.2: Smart contract implementation of university examination of data management

How the proposed smart contract works.

The model works for the generation and verification of on demand documents.

In the first step, the university stores the information of results published through the software that the users interact with to the blockchain.

In the second step the smart contract which is developed in a python framework is initiated to generate the detailed conditions and terms.

In the third step, each term and condition and requisite fee are checked by both the university and the user.

In step four, if step three is accepted, the agreement will be executed which are inform of a smart contract.

In step five, the records are delivered and a message sent on phone is generated.

In step six, if there is a case of illegal activity, the university has the right to halt the services with legal actions. In step seven, the exit action happens (Ashis, 2020).

2.2.3. Multi-factor authentication scheme for online examination

Multi-factor authentication (MFA) is a security scheme that requires users to provide multiple forms of identification or credentials to access a system or service. In the context of online examinations, MFA adds an extra layer of security to verify the identity of test takers and mitigate the risk of unauthorized access or cheating (Naveen, 2018).

The system being discussed in this study employs a multi-factor authentication process that comprises Face Recognition, One Time Password (OTP) Verification, and Fingerprint Authentication modules.

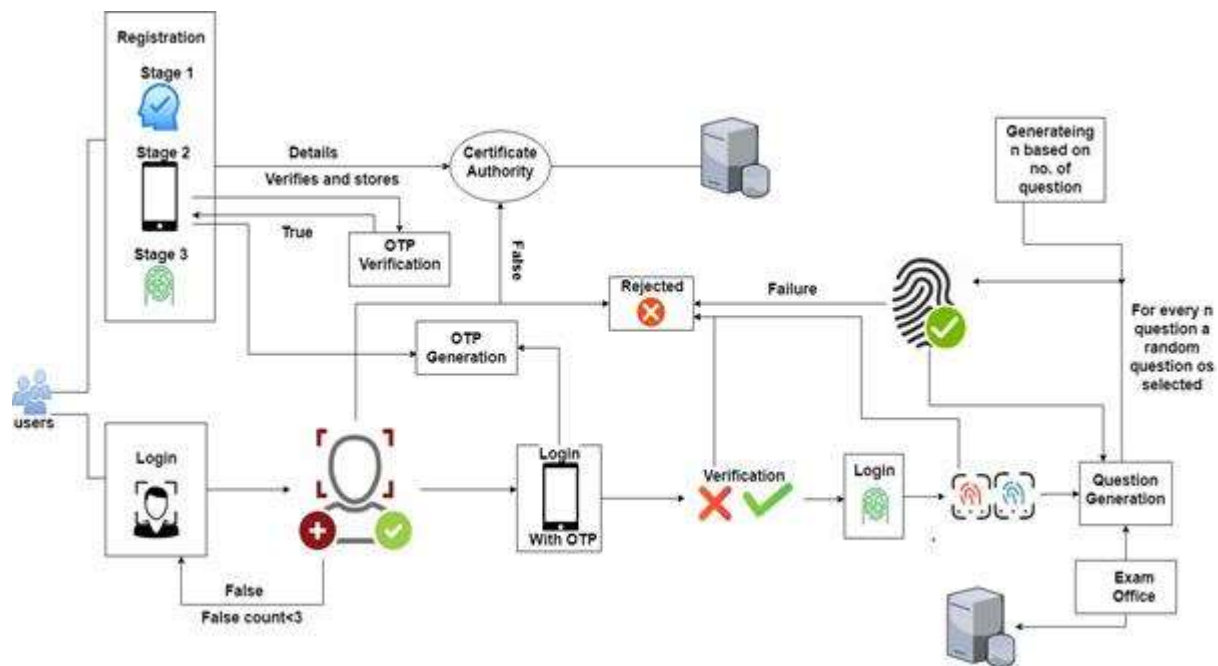


Figure 2.3: A model for multifactor authentication scheme for online examinations

How the prototype works.

The system uses a hierarchical structural methodology, and the user initiates the process by completing the mandatory Registration procedure (Naveen, 2018).

After receiving a unique Register ID issued by the university, the user is required to register with the system by submitting a right forefinger impression and a personalized mobile number that has been verified through OTP.

Once the Registration process is complete, the system proceeds to the Login module. Users can log in to the System using the Login module (Naveen, 2018).

The process starts by asking the user to input their registration ID. Next, a webcam shot of the user is taken and put through a facial recognition process to verify their identity. The user is given a second chance if they fail the first time. But if the user fails twice, access to the system is restricted and a report is forwarded to the controlling authority. The user must skip an OTP verification phase after the Face Recognition module (Naveen, 2018).

An OTP is created automatically and sent to the registered mobile phone in this module. The user is denied access if the OTP is not entered correctly (Naveen, 2018). The user is directed to the Fingerprint Verification Module after completing the OTP process successfully, where the registered fingerprint of the user is compared to the fingerprint received from the fingerprint scanner. The user is given access to the system after completing this test (Naveen, 2018).

2.3 GAPS IDENTIFIED

Using block chains for securing and verifying examination results would be a good option but unfortunately, blockchain technology does not provide room for modifying information if the action is requested by the owner. (Ashis, 2020). Integration of blockchain technology is also expensive since there are costs which are incurred on every single transition that is made (Ashis, 2021).

TEVETA Examinations Verification System provides hashing and encryption of information which may not be enough to stop attackers from accessing sensitive information. Special attacks like brute force, rainbow tables and lookup tables and reverse lookup tables can still be performed on the hashed information. The system does not involve the use of salt hashing which would have helped to make the system more secure (Lister, 2019).

Multifactor scheme for authentication can be bypassed if an unauthorized personal gets direct access to the database. Multifactor authentication does not help much if the stored data in the system is not encrypted or even hashed. The system only prevents non-technical people from

accessing it, but it will not stop advanced hackers from finding vulnerabilities that will enable them access the records stored in the database (Naveen, 2018).

Table 2.2: The table below summarizes the reviewed systems and the gaps identified.

No	Topic	Type of Hashing Method used	Comments
1	TEVETA Examination Results Dissemination and Verification System	The system does not support salt hashing. But it relies on the use of SHA256 to hash the store data.	The developed system uses hashing and encryption to protect user data. But the hashed data can be reveled through the use of reverse look up.
2	A Blockchain-Based Smart Contract Towards Developing Secured University Examination System	The model developed uses SHA 256 hashing algorithms.	The developed system uses hashing but once data is stored in the system, it cannot be changed. It is expensive to implement this type of system.
3	Multi-factor authentication scheme for online examination	The developed model uses hash algorithms to verify user but it is not used to store information.	The system relies more on multifactor authentication to protect access to data.

2.4 PROPOSED MODELL FOR RESULTS VERIFICATION

This study is adapting a model for the Technical, Entrepreneurial, and Vocational Education and Training Authority (TEVETA). The objective of this adaptation was to enhance the existing

model's applicability and effectiveness. By carefully studying TEVETA's unique requirements, goals, and challenges, the study focused on customizing the model to improve on the security. This involves making necessary modifications to the model's architecture. This adaptation provided valuable insights, recommendations, and decision-making support thereby contributing to the advancement and success of developing an improved model for results verification. Only part of the model was developed and the main focus was on storage and verification of the data stored in the system while putting emphasis on the use of salted hash functions.

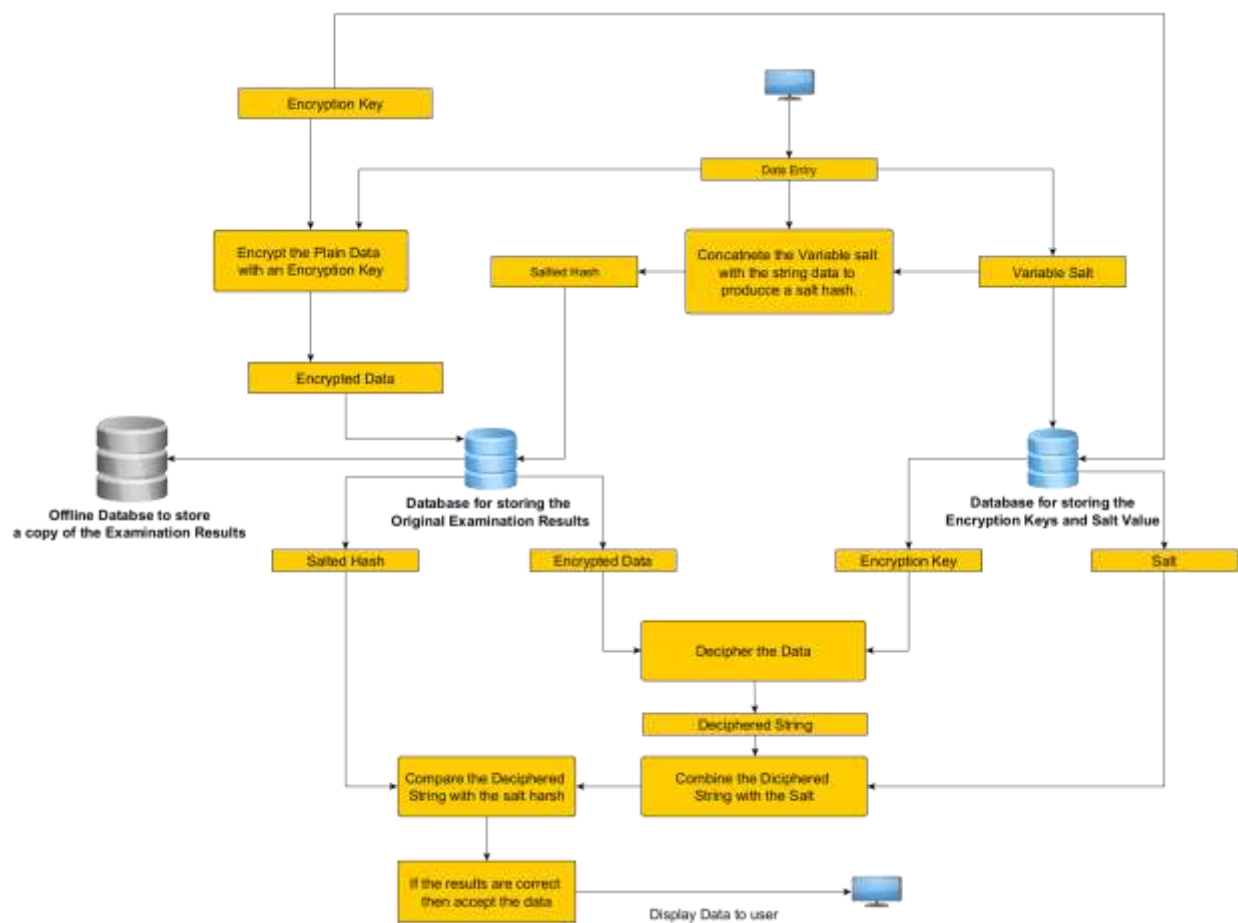


Figure 2.4: The Proposed model for results verification.

How the proposed model work.

The data is input into the system. A variable salt is generated and then it is concatenated to the original data so that it can be hashed. The hashed results are stored in the database and the variable salt is also stored in a separate database. The data entered in the system is also encrypted and an encryption key is generated. The encrypted data is stored in the database and the encryption key is stored in the same database where the variable salt was stored (Lister, 2019).

When the user tries to retrieve the stored data, the encrypted data will be retrieved from the system and the encryption key will be retrieved from the second database so that it can be used to decrypt the data. The hashed data will also be retrieved from the first database and the variable salt will be picked from the second database. The decrypted data will be concatenated with the retrieved variable salt and hashed so that it can be compared with the hash that was picked from the first database. If the hashes match, the user will be informed that the data is valid. If the hashes don't match, the user will be informed that the data was tampered with (Lister, 2019).

Difference between the proposed model and existing models.

The Technical Education, Vocational and Entrepreneurship Training Authority (TEVETA) system prototype supports hashing algorithms for SHA 256 meaning that the data can be hashed. The challenge with this method is that the hashed data can easily be reversed to reveal the original information. The proposed model introduced the use of salts which makes it difficult to reverse the hashed data.

The second system of a Blockchain-Based Smart Contract Towards Developing Secured University Examination System uses blockchain technology to store the data safely. This data cannot be tampered with or modified. The disadvantage of this method is that it is very expensive to implement since blockchain is a new technology. The proposed system used PHP and MySQL to implement all the required functions since these technologies are cheaper to integrate.

Multi-factor authentication scheme for online examination system relies more on multifactor authentication to protect access to data while the proposed model only focused on the storage and protection of the data stored in the database.

Unique functionality about the proposed model.

The unique feature about this proposed model is, it provides another layer of security to the hashed values stored in the database by concatenating a salt to the original data before hashing it. Hence making it more difficult for an attacker to use brute force attacks or rainbow tables to try and reverse hash the values stored in the system which may compromise the integrity of the information (Hornby, 2021).

Generating the Salt

The salt that will be mixed with the information is created during this step. A random function that belongs to the PHP programming language is used to create the salt. It significantly contributes to the durability of hashed values. A straightforward piece of data will gain strength by employing salt, the more distinctive and the longer the better. To make the testing process simpler, the salt generated in this study can only be 3-character numbers (Sutriman, 2019).

CHAPTER THREE: METHODOLOGY

3.1 INTRODUCTION

Research techniques are a strategy to gather varied data that is then turned into knowledge. The knowledge is employed as a resource to address the issues under study. Data collecting, data grouping, analysis, and drawing of findings are the stages that were taken in this study (Sutriman, 2019).

The methods used in this study to gather, arrange, and analyze data were emphasized in this chapter, along with the presentation of the research findings and the rationale behind them. In keeping with this, the chapter described how the selected research procedures and methods were applied to address the research questions and subsequently achieve the research's objectives.

3.2 RESEARCH APPROACH

The research approach used was both qualitative and quantitative. The qualitative research approach was focused on the website administrators who manage the backend of the system. There are three website administrators who know exactly how the backend of the system operates and they provided very useful information for the research. The quantitative research approach focused on the staff members who interact with the system because they were more in numbers and it would assist in getting everyone's personal opinion.

3.3 TARGET POPULATION

The target population for the research included three website administrators who control and manage the system. These website administrators know exactly how the back end of the system operates and they provided very useful information. The other target population included in this research were twenty-five staff members from different colleges which included the Heads of Department, Principal Deans, Associate Deans and Lecturers who interact with the front end of the system and are responsible for both filling in the marks and checking for the results.

3.4 SAMPLING TECHNIQUES

Purposive sampling technique was used for this research. The justification for using purposive sampling was because the research only needed feedback from the people who had interacted with the current system on a daily basis since they provided feedback on exactly what it can do and what it cannot do. This helped in collecting accurate data because it provided firsthand information.

3.5 SAMPLE SIZE

The table below show the sample size of the respondents that was used for data collection.

Table 3.1: The sample size of the respondents that was used for data collection.

Category	Position in organization	Number of respondents
Teaching Staff Members	Heads of Department	7
	Principle Dean	2
	Associate Dean	2
	Lecturer	4
	DEC	7
Other Staff	Website Administrators	3
Total		25

In the Table 3.1 above, three website administrators were chosen because they are the only staff available that manage the backend of the system. Seven heads of department were selected because they are responsible for displaying the results to the students, two principal deans and two associate deans were chosen from each college because they are in charge of overseeing all the activities that run with in the respective colleges, seven DEC's (Departmental Examinations Coordinator) were selected for the study because they are in charge of inputting the marks in the system, and four lectures who have interacted with the system before were selected to also provide their insights about the system. This makes up the total of up to twenty-five staff members.

3.6 DATA COLLECTION METHODS

The type of data that was collected throughout the research was completely primary data because the research was carried out at the university premises.

3.6.1 DATA COLLECTION OF THE EXISTING SYSTEM

3.6.1.1 INTERVIEW

The researcher engaged with the website administrators face-to-face with the sole purpose of collecting data about the backend of the system. The researcher conducted formal interviews to gather additional information on how the examination system operates and how data storage is handled internally.

3.6.1.2 QUESTIONNAIRE

The researcher used this method of data collection because the respondents were many in number and it was better to let them answer the questionnaire in their own convenient time. The questionnaire was distributed to the staff members who have interacted with the system before since they had some insight on how it operates. Multiple choice questions were provided both on physical papers and online through google forms meaning that the respondent was required to choose the appropriate answers basing on their experience when interacting with the current system.

3.6.2 DATA COLLECTION METHODS USED DURING THE DEVELOPMENT OF THE MODEL.

3.6.2.1 EXISTING SYSTEM REVIEW

Three website administrators were interviewed on some of the functionalities that the system can perform and feedback was provided. Other staff members like the principal dean, the Head of Department and other School administrators also provided their feedback on how the system operates which helped the research to know exactly what to focus on when developing the model.

3.6.3 DATA COLLECTION METHODS USED FOR THE EVALUATION

3.6.3.1 EVALUATION WITH PYTHON SCRIPT

A python script was developed and the script was used to evaluate the system. The python script contained code that would be used to try and reverse engineer the salted hashed values and the time was recorded for how long the code would take to successfully reverse engineer the given input.

3.7 ANALYSIS OF COLLECTED DATA

3.7.1 DATA ANALYSIS METHODS USED WHEN INVESTIGATING THE CURRENT SYSTEM

3.7.1.1 USE OF SPSS TO ANALYSE DATA COLLECTED FROM QUESTIONIER

SPSS was use to record the input for every questioner and later the data was organized accordingly. After organizing the data, SPSS was used to generate tables that summarized the collected information. Later the information in the tables was interpreted by the researcher to provide explanations for the findings.

3.7.2 DATA ANALYSIS METHODS USED WHEN DEVELOPING THE MODEL

3.7.2.1 CONTENT ANALYSIS

After the interview process with website administrators, the researcher analyzed the collected information which involved investigating the programming language which was used to create the back end of the system and the type of databases which were used to store the data. This guided the researcher to decide on which technologies to use when developing the proposed model in chapter two.

3.7.3 DATA ANALYSIS METHODS USED FOR EVALUATION

3.7.3.1 PERFORMANCE ANALYSIS

A python script was used to try and reverse hash the values that were stored in the database. During the reverse hashing process, the time that it took for the script to attempt recovering the original data was recorded. This data was used to evaluate if the techniques used to hash the data was effective or not. Tables were created to summaries the data and determine the effectiveness of the proposed model.

3.8 METHOD FOR DEVELOPING A SECURE MODEL

The waterfall model was chosen for this research as the preferred development methodology. The decision to utilize the waterfall model was based on several factors and considerations. Firstly, the waterfall model offers a sequential and structured approach to software development, allowing for a systematic progression from one phase to another. This was particularly important in ensuring a clear and well-defined development process for the secure model using salted hash functions. Additionally, the waterfall model provided a well-defined set of deliverables for each phase, enabling a more organized and traceable development process. Moreover, the waterfall model aligned well with the research objectives and timeline, as it allowed for thorough planning, requirements gathering, and design before proceeding to the implementation and testing phases. This approach ensured that each phase was completed before moving on to the next, promoting a more controlled and predictable development process. The phases that are involved in this model include:

Requirement Analysis

Analysis was conducted to identify the requirements for the secure model. This involved understanding the specific type of database to use and the appropriate programming language for developing the model.

The appropriate technologies and tools were selected for developing the secure model. This included choosing a programming language that supported the implementation of salted hash functions and ensured secure data storage and processing. PHP programming language and MySQL database were chosen for this purpose. PHP is a popular programming language and it is

easier to set up for web projects. MySQL is the most widely used technology for developing databases and it was chosen for this purpose since the current system also relies on the same technology. Using these technologies meant that little time and money would be spent on trying to develop the prototype.

System Design

Based on the requirements analysis, a system design was created to outline the architecture, components, and interactions of the secure model. UML diagrams was used to show exactly how the system would operate.

Implementation

The secure model was implemented according to the defined architecture and functionalities. The development process included coding the necessary algorithms for salted hash functions, result verification, and database operations.

Evaluation

During this phase, a python script was written in an Integrated Development Environment (IDE) know as PyCharm with the use of Python version 3.11. The computer used for evaluation was running on Windows 11 operating system. It had 16 Gigabytes of Ram and had an intel Core i5 Processor. On executing the python script, a hashed value was provided as input and the computer tried to reverse hash the value against all possible hashing algorithms possible until the original data was uncovered. The time that it took for the computer to uncovered the original data was display in seconds that it was recorded. Then the same process was repeated with a salted hashed value and again the time take to try and recover the original data was also recorded and compare with the previous data.

Table 3.2. Summary Table for The Methodology Used to Achieve Each Objective

	Objective	Methodology	Data analysis tools	Outcome of each objective
1	To investigate the current information systems that is used to store and manage student results at KIU	Literature review, Questioners and Interviews.	Qualitative and Quantitative analysis.	Gain insights into the existing systems at KIU and identify strengths and weaknesses.
2	To develop a secure mode that uses salted hashed functions to verify students' results.	System development using the Water fall model.	Programming languages like PHP and MySQL	Develop a secure model using salted hashed functions for verifying student results.
3	To evaluate the Secure model for students' results verification.	Evaluation using test scenarios and data analysis of evolution results.	Python programming language.	Evaluated the effectiveness of the secure model in protecting the students results.

CHAPTER FOUR: DATA PRESENTATION, ANALYSIS AND FINDINGS

4.1 INTRODUCTION

This chapter contains research findings, data presentation, and discussions from in-depth interviews and questionnaires, regarding the use of salted hash algorithms to secure examination results. This chapter also consists of the demographic statistics of the study respondents followed by the processed and analyzed structured interview questions, which have been analyzed with a software package known as Statistical Package for the Social Sciences, and the data is well presented in a table.

4.2 Data Presentation and Analysis of the Findings

4.2.1 Objective One: To Investigate the current information systems for storing and managing students' results

QUESTIONNAIRE

Questionnaire were provided to the staff at the university and data was collected regarding on what they experience when the interact with the system.

Table 4.1: The table below shows the Staff position of the respondents.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Head of Department	7	31.8	31.8	31.8
	Principle Dean	1	4.5	4.5	36.4
	Associate Dean	2	9.1	9.1	45.5
	Lecturer	4	18.2	18.2	63.6
	DEC	7	31.8	31.8	95.5
	Other	1	4.5	4.5	100.0
	Total	22	100.0	100.0	

Source: Primary Data 2023

The data in Table 4.1 provided shows the job positions of the respondents in a survey. There was a total of 22 respondents. The analysis provides a clear breakdown of the distribution of job positions among the respondents. The highest proportion of respondents are in the "Head of Department" and "DEC" positions, each accounting for 31.8% of the total respondents. Lecturers make up the next significant group, with 18.2% of respondents identifying as such. Associate Deans and Principal Deans each represent 9.1% and 4.5% of respondents, respectively. Additionally, there is one respondent categorized as "Other."

Do marks get changed in the system without any notice?

One of the questions provided in the questioner was aimed at finding out if there had been an instance where marks were altered in the system without the knowledge of the staff that interact with the system and the results are provided below.

Table 4.2: The table below shows responses regarding if marks are changed in the system.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Yes	13	59.1	59.1	59.1
	No	9	40.9	40.9	100.0
	Total	22	100.0	100.0	

Source: Primary Data 2023

The provided dataset in table 4.2 offers insights into respondents' answers concerning whether marks were subject to changes in the existing system. Out of the total participants, 59.1% indicated that marks were indeed altered, while 40.9% of respondents reported that marks remained unchanged. This data sheds light on the prevalence of alterations within the system, highlighting the need for improved security measures to ensure the integrity of examination results.

How often you attend to concerns of missing marks?

The purpose of this question was to find out how often the respondents addressed the issues concerning missing marks and the table below provides the summarized data collected.

Table 4.3: The table below shows how often staff members attend to concerns of missing marks.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Daily	12	54.5	54.5	54.5
	At the Beginning of the Semester	2	9.1	9.1	63.6
	Mid Semester	2	9.1	9.1	72.7
	Once or twice a week	6	27.3	27.3	100.0
	Total	22	100.0	100.0	

Source: Primary Data 2023

The data presented in the table 4.3 sheds light on the frequency with which concerns related to missing marks are addressed. A detailed analysis of the responses reveals the patterns in addressing this issue. Notably, 54.5% of participants indicated that they attend to concerns of missing marks on a daily basis, suggesting a consistent and proactive approach to addressing student complaints. Additionally, 9.1% mentioned addressing such concerns at the beginning of the semester, possibly to ensure a smooth start to the academic term. Similarly, another 9.1% mentioned addressing these concerns mid-semester, likely indicating a periodic review to prevent and rectify any discrepancies. Furthermore, 27.3% shared that concerns are tackled once or twice a week, potentially reflecting a more structured and scheduled approach to handling missing marks.

Why Students Submit complaints for Missing Marks?

The purpose of this questioner was to investigate why students experience issues concerning missing marks. The table below summarizes the finds that were got during data analysis.

Table 4.4: The table below shows some of the reasons why students submit complaints for missing marks.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Lecturers don't want to submit marks in given time.	7	31.8	31.8	31.8
	Students claim to have done the exam and yet they did not write it.	2	9.1	9.1	40.9
	The results go missing unexpectedly	7	31.8	31.8	72.7
	Other	6	27.3	27.3	100.0
	Total	22	100.0	100.0	

Source: Primary Data 2023

The data in the table 4.4 delves into the reasons behind students submitting complaints for missing marks within the context of the existing system. An analysis of the responses reveals various factors contributing to these complaints. Among the participants, 31.8% attributed the issue to lecturers not adhering to the designated mark submission timelines. Another 9.1% mentioned instances where students claimed to have taken an exam that they did not actually write, suggesting possible miscommunication or misunderstanding. Equally noteworthy, 31.8% of respondents cited unexpected occurrences where results disappeared without explanation, causing concerns among students. An additional 27.3% expressed other reasons for missing marks. The cumulative percentages highlight the distribution of these contributing factors, emphasizing the multifaceted nature of the problem.

INTERVIEW

An interview was carried out with the websites administrators to find out more about the current system and to analyze its strengths and weaknesses. The table below summarizes the data that was collected during the interview.

Table 4.5: Results of investigations of the current information systems.

	Feature	Strengths	Weaknesses
1	Data Back up	The system provides back up of information which is stored in the cloud. The cloud services provider offers the ability to make rollbacks in case there is a data breach.	There is no offline back up of the data. According to the system administrator it is tiresome and it requires a lot of resources to manage.
2	Validation		There is no automated validation of information in the system. Validation of the data is done manually by the school staff members who have to check the data provided by the print out and compare it with the information that is on the results slip.
3	Data Integrity.	Data Integrity is provided by ensuring that the data which is entered into the system is encrypted.	There are no hashing algorithms that can be used to check if the current information stored in the system is valid.
4	Insider Threats	Encryption can help to reduce on the risk of external insider threats.	There is no adequate mechanism that can be used to stop internal insider threats.

Source: Primary Data 2023

Table 4.5 shows the summary of the responses that were got from the website administrators who manage the backend of the system. The investigation of the current system focused of five major sections which include, backup, data integrity, insider threats, and data validation.

4.2.2: Objective Two: To develop a secure model that uses salted hashed functions to verify student's marks

SYSTEM PROTOTYPE DESIGN

The system prototype employs a combination of encryption and salted hash functions to achieve its goals. There are two phases that are involved in this model, the first one is to store the results in the databases and the second one is to retrieve those results and verify them.

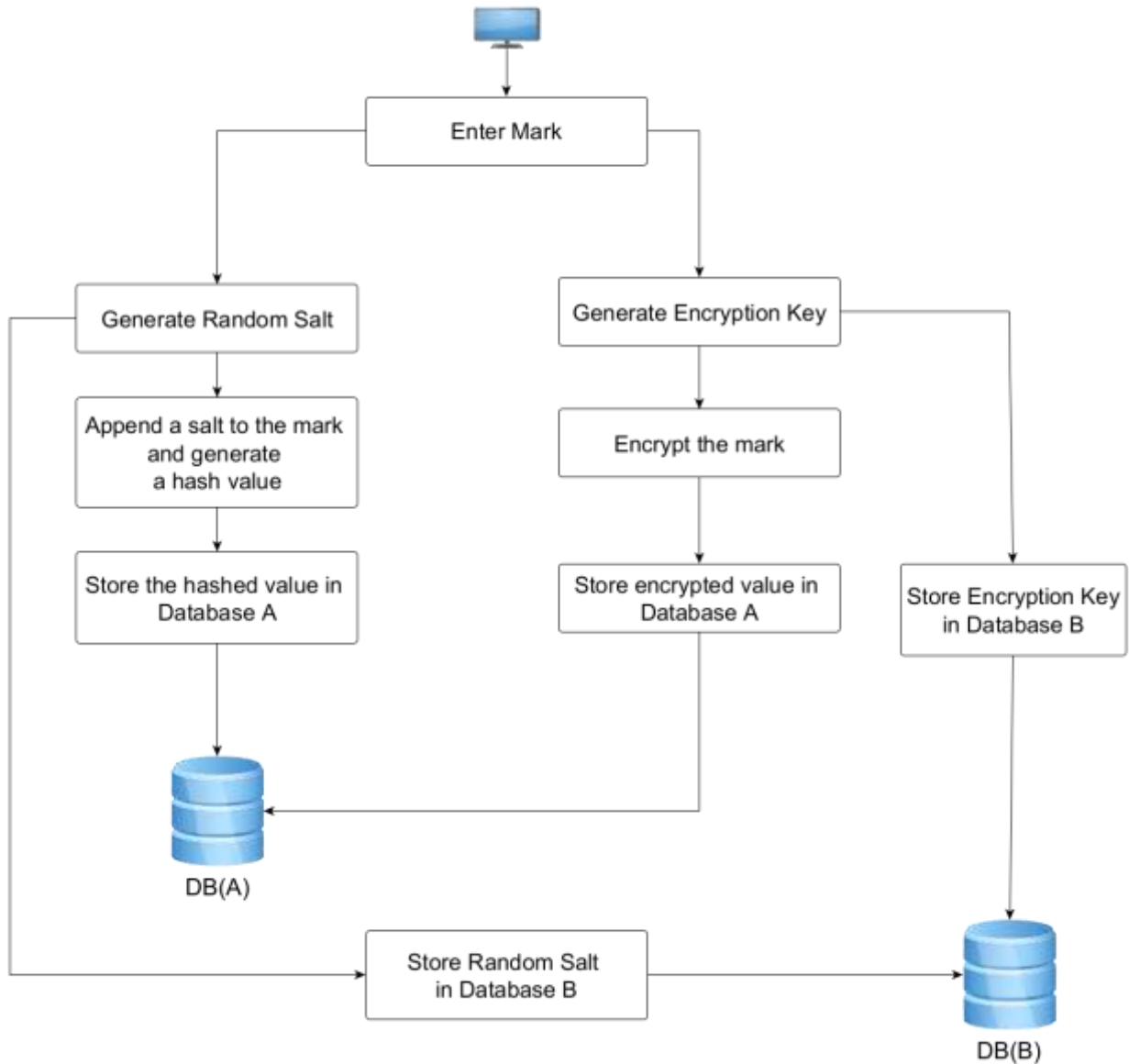


Figure 4.1: The figure above shows the process of storing the results in databases.

As shown in Figure 4.1, the examination results are encrypted using a strong encryption algorithm such as AES (Advanced Encryption Standard). This encryption process ensures that the data remains confidential and unreadable to unauthorized individuals even if the database is compromised.

To further enhance the security, salted hash functions are utilized to store a hashed representation of the examination results. Before hashing, a unique randomly generated salt is added to each student's result data. This salt is securely stored in a separate database, ensuring that it remains independent of the encrypted data. By incorporating the salt into the hashing process, the resulting hash becomes unique and unpredictable, making it significantly more difficult for potential attackers to crack or reverse-engineer the original results.

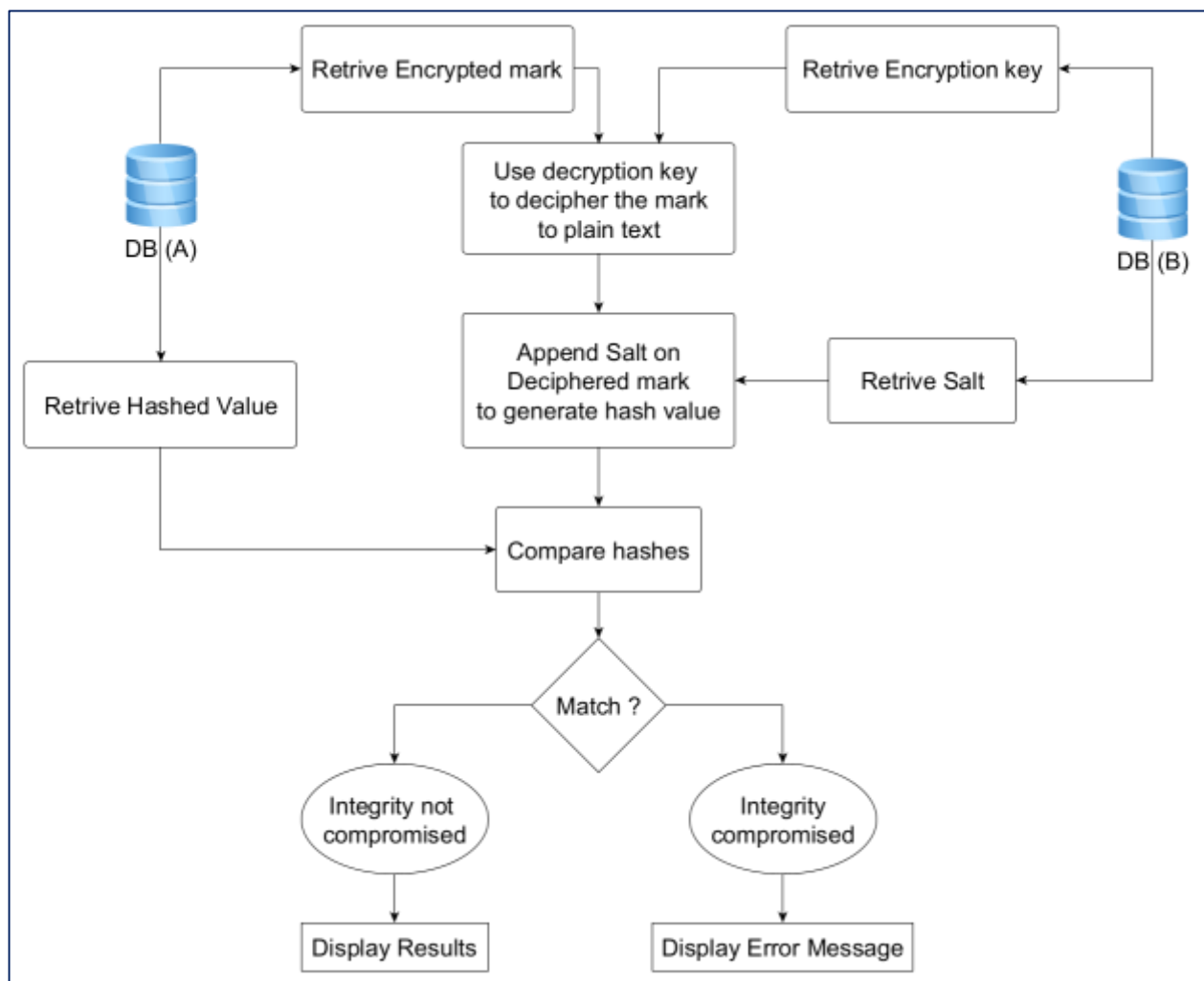


Figure 4.2: The figure above shows the process of retrieving the results from the database.

As shown in figure 4.2, during the verification process, the system retrieves the stored salt for the specific student and combines it with the entered examination result. The system then applies the salted hash function to the combination and compares it with the stored hashed result. If the generated hash matches the stored hash, the system confirms the integrity of the result and confirms that it has not been tampered with.

To implement the system prototype, a robust database schema is designed to store the encrypted examination results, along with the associated salt and hashed values. The database schema includes appropriate table structures and relationships to efficiently store and retrieve the data.

It is important to note that while the system prototype design offers a strong level of security for examination results, it is not immune to all potential security risks.

SYSTEM PROTOTYPE DEVELOPMENT

The development process began by selecting a suitable programming language to build the system prototype. Considerations are given to factors such as performance, security features, and ease of implementation. The chosen technology stack provided a solid foundation for developing a scalable and secure system. In this case, PHP was chosen as a suitable programming language because, it is easy to set up, it does not require any further configuration to get the prototype to work and it is easy to maintain the source code. PHP is also supported by majority of the web hosting platforms and so it becomes easier to find an alternative which is cheaper to use for hosting the prototype.

User Interface

A user-friendly interface was developed that allowed administrators and faculty to interact with the system. The interface provided functionalities for data entry, result validation, and reporting.

Result Data Storage

A database system was implemented to securely store examination result data. The database supported efficient data retrieval and data integrity to ensure the confidentiality of sensitive information.

id	registration_number	hashed_mark	encrypted_mark	subject	semester	year
51	2021-01-03178	fb0e53b01caacc5ed61f475530fa22769274a777c34714ee6...	MW9kMQTnAKaW1UXgY/YYBA==	UCC1100	1	1
52	2023-04-09866	4d90960cbbcae3eab885fb836314522c63cee06b511a35deec...	9+sOS1ncMO8pYptmnV1qw==	UCC1201	2	1
53	2020-01-0-23345	4c497f268cf3d040c353b0dabac421a344bddfb85f084400e...	xGDhyz05VgobnRoeWCH6Ga==	DLS2104	2	2
54	2021-03-45322	8a4c84fbd17e51565f455d84f712f5d3f42aca279a2a76c433...	xJLGE/WgKcDPT3D38wyetb==	DLS2202	2	3
55	2021-02-98843	481075eb461681c395e1dd3948f5eb2d9e373bc5c3b5d75d80...	EfWoppEm1n6Pcbb8Xmrg==	DLS1104	2	2
56	2023-3-03422	4245d7a1bbdbbdee9170cd2c0ff441cec32ce5d69ffa264fe...	/ITQn+0a/h/DLisH58TFA==	IFM3102	1	1
57	2022-3-09843	d7cdd45e211f668a05b017e3acc0a3f4bf6e9176d49e55c24...	xGTdxBdmUciYGVCA7QyJw==	DLS2203	2	2
58	2021-1-033458	b2b1dbe9eab5d7d9b659524011e13a77c88b5f07f1bbc5954...	gh+sur99FXeVbbOxfYZBIQ==	UCC1100	1	3
59	2021-2-99983	488295b0d245fa3337c4c9ae2a32ab4453ee14297ad5635b97...	IOkgE/IRnWC+kz871lygnw==	DLS2205	2	3
60	2023-2-3456	dbfd0b5ca08bb54794581016707e9ac0cb59cc5aabb9cd0...	YYcNEsK8ALYqmw4LuSXv6A==	DLS2107	2	2
61	2021-3-43454	76ce2ad6e745a9365465a639ea38fd45562f9c04feda26ea8...	rZ34lqF03Lp+8Ut+kp5/1w==	DLS2203	1	3
62	2022-2-23456	425e315682f7c54c3b02ce7dfe84a138b617bd230ad11cd892...	M7uNMET89HFUz27KuntLg==	COS1202	1	2

Figure 4.3: The figure below shows the database storing encrypted marks and salted hashed values.

In figure 4.3, it shows how the results are stored. The marks are encrypted and salted hash value is generated for every mark stored in the database.

Encryption and Salt Hash Functions

The necessary algorithms and logic were developed for encrypting and decrypting examination result data using salted hash functions. This included integrating the chosen cryptographic algorithms and applying the salted hash function to ensure data integrity.





































← T →				id	registration_number	encryption_key	salt
<input type="checkbox"/>				51	2021-01-03178	f84efcb79dee5b3fa8d7f933effe77f9	1M3UF
<input type="checkbox"/>				52	2023-04-09866	1c902d89790717b8f017235682a9780e	nZPUE
<input type="checkbox"/>				53	2020-01-0-23345	ae6dd6e1555ad4d66746c22f956b5f34	rJY1I
<input type="checkbox"/>				54	2021-03-45322	d38d46663dc52112440adaffbc3f5ab6	EdniV
<input type="checkbox"/>				55	2021-02-99843	ab4e8b048b79db8d3dd8da5007eb266e	mzEIP
<input type="checkbox"/>				56	2023-3-03422	56b6218db89995b2829a53a003ff9d83	XW3QU
<input type="checkbox"/>				57	2022-3-09843	61110c349b857325b775c7d9a5b0f276	drHX
<input type="checkbox"/>				58	2021-1-033458	35f3d07fadfadf134b162e0906d7409b	Z9fVE
<input type="checkbox"/>				59	2021-2-99983	2a864f106d89a0eadbdd40bbd9966cf4	RyVgd
<input type="checkbox"/>				60	2023-2-3456	9cf3137918499344c70194452a417e31	uPXGM
<input type="checkbox"/>				61	2021-3-43454	58d96e224d97f8450d2b7bdae4c85a86	XpwM6
<input type="checkbox"/>				62	2022-2-23456	54b137380cc637c519354283c6ed1ea3	hD1IS

Figure 4.4: The figure above shows the encryption keys and salt values stored in a separate database.

In figure 4.4, it shows database B where the encryption keys and salt values are stored. They are stored in a separate database so that in case of a data breach these keys remain safe.

SYSTEM PROTOTYPE OPERATION

The system prototype was designed to provide a user-friendly interface for administrators to interact with the examination result data securely. Upon accessing the system, administrators can securely upload encrypted result data and assign unique salts to each student's record. The system ensures that the salts are stored separately from the encrypted data, enhancing the overall security of the results.

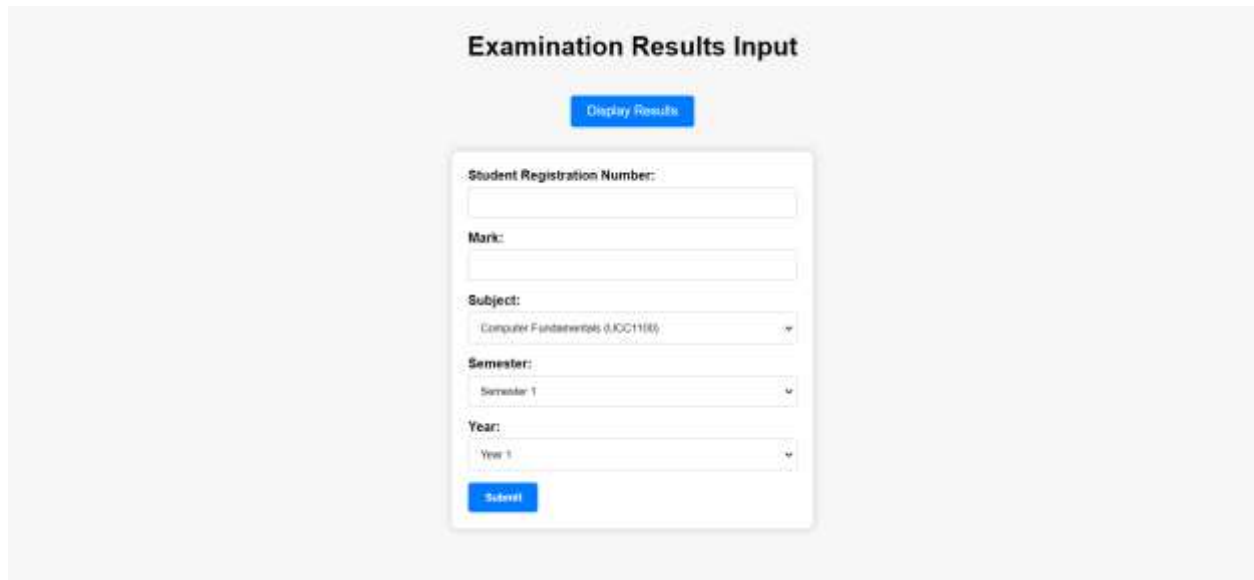
The image shows a web interface titled "Examination Results Input". At the top center is a blue button labeled "Display Results". Below this is a white form box with a light gray border. Inside the form, there are five input fields: "Student Registration Number:" (a text input), "Mark:" (a text input), "Subject:" (a dropdown menu showing "Computer Fundamentals (JGC1100)"), "Semester:" (a dropdown menu showing "Semester 1"), and "Year:" (a dropdown menu showing "Year 1"). At the bottom of the form is a blue button labeled "Submit".

Figure 4.5: Interface for inputting marks

The figure 4.5 shows the interface that will be used to input the data. It provides fields for imputing the students, registration number, marks, subject, semester and year.

End users can access the examination results for their respective courses through a controlled and authenticated interface. The system retrieves the appropriate salts for the entered results and applies the salted hash function to verify the integrity of the data. This process ensures that the results remain protected from unauthorized modifications or tampering.

Examination Results					
Filter by: Registration Number <input type="text"/> Filter value <input type="button" value="Filter"/>					
<input type="button" value="Add Results"/>					
Registration Number	Subject	Year	Semester	Mark	Validation
2021-01-03178	UCC1100	1	1	67	Valid
2023-04-09866	UCC1201	1	2	78	Valid
2020-01-0-23345	DLS2104	2	2	89	Valid
2021-03-45322	DLS2202	3	2	54	Valid
2021-02-99843	DLS1104	2	2	45	Valid
2023-3-03422	IFM3102	1	1	36	Valid
2022-3-09843	DLS2203	2	2	53	Valid
2021-1-033458	UCC1100	3	1	60	Valid
2021-2-99983	DLS2205	3	2	64	Valid
2023-2-3456	DLS2107	2	2	84	Valid
2021-3-43454	DLS2203	3	1	71	Valid
2022-2-23456	COS1202	2	1	88	Valid

Figure 4.6: Interface showing all verified results which are valid.

In figure 4.6, if the data is not tempered with, the system will display a green text mark showing that the data is valid.

Examination Results					
Filter by: Registration Number <input type="text"/> Filter value <input type="button" value="Filter"/>					
<input type="button" value="Add Results"/>					
Registration Number	Subject	Year	Semester	Mark	Validation
2021-01-03178	UCC1100	1	1	67	Invalid
2023-04-09866	UCC1201	1	2	78	Valid
2020-01-0-23345	DLS2104	2	2	89	Valid
2021-03-45322	DLS2202	3	2	54	Valid
2021-02-99843	DLS1104	2	2	45	Invalid
2023-3-03422	IFM3102	1	1	36	Valid
2022-3-09843	DLS2203	2	2	53	Valid
2021-1-033458	UCC1100	3	1	60	Valid
2021-2-99983	DLS2205	3	2	64	Invalid
2023-2-3456	DLS2107	2	2	84	Valid
2021-3-43454	DLS2203	3	1	71	Invalid
2022-2-23456	COS1202	2	1	88	Valid

Figure 4.7: It shows the model display some invalid results shown with a red text.

As shown in figure 4.7 if the data has been modified, the system will display a text message in red indicating that the data is invalid as shown in the image above.

4.2.3 Objective Three: To Evaluate the secure model

A python script was used to evaluate the different hashing algorithms used. The python script was developed in PyCharm which is an Integrated Development Environment (IDE). The python script included code that would be used to hash different values and then a reverse hashing function was used to try and reverse the hashing. The time interval was recorded for how long it took to the script to try revers hashing the different values.

Evaluation Objectives

The objectives of the evaluation were to check how long it would take for the salt hashed data to be reversed back to the original data as to compared with non-salt hashed data.

Different hashing algorithms have been used to try and reverse the data as show in the evaluation results below.

Evaluation Results

In the first attempt of evaluation, a value of 76 was hashed without providing a salt to the data and a python script was written to evaluate the results as shown below.


```
Enter the hash to be reversed: f74efabef12ea619e30b79bddef89cffa9dda494761681ca862cff2871a85980
Attempting to reverse the hash...

--- MD5 ---
Unable to reverse the hash.

--- SHA1 ---
Unable to reverse the hash.

--- SHA256 ---
Original Input: 76
Reverse Time: 0.000000 seconds

--- SHA512 ---
Unable to reverse the hash.

--- Performance Evaluation ---
Average Reverse Time: 0.816419 seconds
Total Time Taken: 3.265677 seconds

--- Time Taken for Individual Hashing Algorithms ---
MD5: 0.996987 seconds
SHA1: 0.952493 seconds
SHA256: 0.000000 seconds
SHA512: 1.316198 seconds

Process finished with exit code 0
```

Figure 4.8: Showing the results after reverse hashing

In figure 4.8, the evaluated data shows the results of attempting to reverse a given hash "f74efabef12ea619e30b79bddef89cffa9dda494761681ca862cff2871a85980" using various hashing algorithms in a Python script. The script tried different hashing algorithms, including MD5, SHA1, SHA256, and SHA512, to determine which one was used to hash the original input "76". The script was able to successfully reverse the hash using the SHA256 algorithm, revealing the original input "76" in a very short time of 0.000000 seconds. However, the other hashing algorithms, namely MD5, SHA1, and SHA512, could not reverse the hash because the hashed data did not match any of the algorithms. The performance evaluation shows that the average time taken

to attempt reversing the hash using all algorithms was 0.816419 seconds, with a total time taken of 3.265677 seconds for all attempts.

Summary of Results:

Table 4.6.: Shows the results that were obtained after try to reverse hashed data without salt.

Hashing Algorithm	Reversed Input	Time Taken (seconds)
MD5	Not Reversed	0.9969987
SHA1	Not Reversed	0.952493
SHA256	76	0.000000
SHA512	Not Reversed	1.316198

The table 4.6 summarizes the results of the evaluation, indicating the success or failure of each hashing algorithm in reversing the given hash, along with the time taken for each attempt. The data highlights the effectiveness of the SHA256 algorithm in quickly reversing the hash and identifying the original input, "76", while the other algorithms demonstrate their resistance to hash reversal attempts. The Python script's ability to determine the correct hashing algorithm used for the given hash showcases its utility in cryptographic analysis and security applications. The performance evaluation provides valuable insights into the efficiency and reliability of each hashing algorithm for secure data storage and verification purposes.

In the second attempt of evaluation, a salt was appended on the original data before hashing was done. The original value was "82" and salt value was "af3ff2dc8d"

```

Enter the hash to be reversed: 6550e1d9a76701c7bc2255f84f5e410e3a87b36afd5c9e62a72cfeb848281c7f
Attempting to reverse the hash...

--- MD5 ---
Unable to reverse the hash.

--- SHA1 ---
Unable to reverse the hash.

--- SHA256 ---
Unable to reverse the hash.

--- SHA512 ---
Unable to reverse the hash.

--- Performance Evaluation ---
Average Reverse Time: 1.077091 seconds
Total Time Taken: 4.308366 seconds

--- Time Taken for Individual Hashing Algorithms ---
MD5: 1.000095 seconds
SHA1: 0.982505 seconds
SHA256: 0.997085 seconds
SHA512: 1.328681 seconds

Process finished with exit code 0

```

Figure 4.9: Showing results of failure after revers hashing.

In the figure 4.9, the evaluated data demonstrates the results of attempting to reverse a given hash "6af2e128b1f2fa25ab2cd7b621810664c0bd776f2354d6caddec38daea3dec7" using various hashing algorithms in a Python script. The hash value "6af2e128b1f2fa25ab2cd7b621810664c0bd776f2354d6caddec38daea3dec7" is the result of hashing the original value "82" with a salt value of "af3ff2dc8d" appended to it. The Python script tested different hashing algorithms, including MD5, SHA1, SHA256, and SHA512, to determine which one was used to hash the original input. However, the script was unable to reverse the hash using any of the algorithms, reaffirming the cryptographic strength and one-way nature of these hash functions. The presence of the salt value in the hashing process significantly increased the difficulty of reversing the hash, showcasing the effectiveness of using salted hash functions for

data security. The performance evaluation indicates that the average time taken to attempt reversing the hash using all algorithms was 1.077091 seconds, with a total time taken of 4.3088366 seconds for all attempts.

Table 4.7: Showing results that were obtained after revers hashing data with a salt.

Hashing Algorithm	Reversed Input	Time Taken (seconds)
MD5	Not Reversed	1.000095
SHA1	Not Reversed	0.982505
SHA256	Not Reversed	0.997085
SHA512	Not Reversed	1.328681

The Table 4.7 above summarizes the results of the evaluation, indicating that none of the hashing algorithms were able to reverse the given hash. The data reaffirms the robustness of using salted hash functions to secure data, as it prevents the reverse engineering of the original data even when the hashing algorithm is not known. The addition of a salt value to the original data adds a layer of complexity that significantly increases the effort required to reverse-engineer the hashed value. This experiment successfully proves the concept that when adding a salt to the original data, it becomes highly challenging to reverse the hashed data, providing a secure method for storing sensitive information and ensuring data integrity in various applications.

4.3 BENEFITS OF THE DEVELOPED PROTOTYPE

By incorporating unique salts for each student's data and applying a robust hashing algorithm, the prototype ensures that the stored results are not only encrypted but also tamper-proof. This level of security provides reassurance to educational institutions, administrators, and students that their grades remain confidential and immune to unauthorized modifications or tampering.

Additionally, the use of salted hash functions in the prototype simplifies the process of result retrieval and verification. Authorized personnel can easily access the encrypted results, apply the corresponding salts, and verify the integrity of the data through hash comparison. This streamlined

approach reduces the chances of human error and expedites the result verification process, enabling instructors to provide timely feedback to students and facilitating efficient academic decision-making.

4.4 CHALLENGES FACED BY THE DEVELOPED PROTOTYPE

One of the primary challenges is the complexity of managing and storing the salts securely. The salts play a crucial role in generating unique hash values and ensuring the integrity of examination results. However, their proper management requires additional considerations and measures to prevent unauthorized access or loss. Establishing secure storage mechanisms, implementing access controls, and adhering to key management best practices are necessary to mitigate this challenge and maintain the confidentiality and availability of the salts.

Another challenge lies in striking the right balance between security and performance. While salted hash functions offer robust security, they can introduce computational overhead, especially when handling a large volume of examination results. The prototype needs to optimize the hashing process and database operations to ensure efficient performance without compromising security. This involves selecting appropriate hashing algorithms, fine-tuning system configurations, and employing caching techniques to minimize processing time and resource utilization.

Maintaining compatibility with existing systems and workflows poses another significant challenge. Educational institutions often rely on multiple systems and platforms for result management, student information systems, and reporting. Integrating the developed prototype seamlessly with these existing systems requires careful planning, data mapping, and ensuring compatibility of data formats and interfaces. It is essential to minimize disruption to current processes and provide a smooth transition to the new system to encourage widespread adoption. The prototype also faces challenges related to user acceptance and adoption. Introducing a new system that utilizes salted hash functions may require training and education for administrators, instructors, and other authorized personnel.

4.5 COMPARISON OF THE DEVELOPED PROTOTYPE WITH THE EXISTING SYSTEMS DISCUSSED IN THE LITERATURE REVIEW

In terms of security, the developed prototype offers robust protection against unauthorized access and tampering. The use of salted hash functions adds an additional layer of security, making it extremely difficult for attackers to decrypt the result data or manipulate the results without detection.

CHAPTER FIVE: DISCUSSION, CONCLUSION AND RECOMMENDATIONS

5.1 INTRODUCTION

This chapter serves as a vital component of this dissertation, providing an opportunity to engage in discussions and draw meaningful conclusions based on the research findings and outcomes. This chapter focuses on the key objectives of the study, namely investigating the current information system, developing a secure model using salted hash functions, and testing and evaluating the model for student's results verification. The discussions and conclusions derived from these objectives shed light on the effectiveness, strengths, and limitations of the developed secure model.

5.2 DISCUSSION

Objective One: To investigate the current information system that is used to store and manage student results at Kampala International University.

The investigation into the current information system used for storing and managing student results at Kampala International University revealed several important findings. Firstly, it was observed that the system does not utilize salted hash functions for securing examination results. This raises concerns about the integrity and authenticity of the stored data, as the absence of salted hash functions can make the system more vulnerable to unauthorized tampering. The lack of this security measure may compromise the overall trustworthiness and reliability of the examination results.

Additionally, the validation of results being done manually introduces potential human errors and inefficiencies. Manual validation processes are prone to inaccuracies, and they consume valuable time and resources. This manual approach may hinder timely and accurate dissemination of examination results to stakeholders, including students, faculty, and administrators.

Furthermore, the related systems that were reviewed in chapter two also implemented the use of hashing algorithms like SHA 256 for example Technical, Entrepreneurial and Vocational

Education and Training Authority (TEVETA) and the blockchain smart contract. For TEVETA, the stored hashed values could easily be reversed since they did not contain any salt to make it difficult for attackers to crack while in the findings for this research the study proved that the proposed model was secure since it used salts to harden the security of the data stored.

Objective Two: To develop a secure model that uses salted hashed functions to verify students' results.

The model was implemented using PHP programming language and MySQL database, and the system model was designed using UML diagrams, ensuring a structured and efficient development process.

By incorporating salted hashed functions into the model, the system provides an added layer of security to the examination result data. Salted hashes improve data integrity by generating unique hash values for each student's result, making it extremely difficult for unauthorized individuals to tamper with or manipulate the data without detection. This aspect enhances the trustworthiness and reliability of the examination results, instilling confidence among students, faculty, and other stakeholders.

The use of PHP and MySQL as the development technologies offers several advantages. PHP is a widely adopted and versatile programming language that provides robust support for web-based applications. Its integration with the MySQL database management system ensures efficient data storage, retrieval, and management. The combination of these technologies facilitates the seamless integration of the salted hashed functions into the existing system infrastructure, making it a practical and scalable solution for result verification.

The adoption of UML diagrams during the system modeling phase contributes to the overall clarity and comprehensibility of the system design. UML diagrams provide visual representations of the system's structure, interactions, and behavior, aiding in the effective communication and

understanding of the model among development teams, stakeholders, and future maintainers. This standardized notation ensures consistency and facilitates future enhancements or modifications to the system.

For the related systems mentioned in chapter two most especially Technical, Entrepreneurial and Vocational Education and Training Authority (TEVETA), the researcher used PHP and MySQL to implement the prototype since it was easier and widely used in most applications. The other related studies on Blockchain smart contract, the researcher relied on the use of Ethereum technology which was found out to be much more expensive to implement. This research focused on the use of PHP and MySQL to develop the proposed prototype since using those technologies was not expensive as compared to blockchain.

Objective Three: To evaluate the Secure model for student's results verification.

The evaluation phase of the secure model for student's results verification involved the development of a Python script to assess the effectiveness and reliability of the implemented salted hash functions. The script aimed to reverse hash the already hashed data. This process provided valuable insights into the security and integrity of the secure model. During the evaluation phase, a hashed value without a salt was provided to the system and the python script managed to reverse hash it while the salted value that was provided to the system failed to be reversed as the results show in chapter four under the subsection of evaluation.

By testing the hashing and reverse hashing processes, the script could validate the robustness and effectiveness of the salted hash functions in protecting the examination result data. The use of different hashing algorithms allowed for a comprehensive assessment of the model's resistance against various cryptographic attacks, including dictionary attacks and rainbow table attacks. This testing process also provided an opportunity to identify any potential vulnerabilities or weaknesses in the implementation of the salted hash functions.

Through the evaluation of the results, it was determined that the secure model demonstrated strong security and integrity in protecting student's examination result data. The salted hash functions proved effective in safeguarding against unauthorized access, data tampering, and brute force attacks. The model's resistance to reverse hashing attempts validated the robustness of the implemented salted hash functions, indicating the difficulty of decrypting hashed data without the corresponding salts.

For the related studies in chapter two on Technical, Entrepreneurial and Vocational Education and Training Authority (TEVETA), the researcher did not do an evaluation on the performance and security of the model but rather the researcher focused on doing an acceptance test to see if the students would be interested in using the prototype that was developed. The results showed that the students were so much interested in using the prototype when necessary. For this study, the research was focused on evaluating the effectiveness of salted hash function in protecting the students results and this research proved that the slated hashed values would not easily be reversed as compared to the hashed values which did not contain any salt.

5.3 CONCLUSION

Objective One: To investigate the current information system that is used to store and manage student results at Kampala International University.

The investigation of the current information system used for storing and managing student results at Kampala International University highlights critical gaps in terms of data security, backup mechanisms, and result validation processes as shown by the results in chapter basing in the feedback that was provided by the web administrators. The absence of salted hash functions compromises the integrity and security of the examination results, while the lack of offline backup increases the risk of data loss. Additionally, the manual validation of results introduces potential errors and delays.

Objective Two: To develop a secure model that uses salted hashed functions to verify students' results.

The incorporation of salted hashed functions strengthens the system's resistance against unauthorized modifications or tampering, reinforcing the trustworthiness of the examination result data.

The utilization of PHP and MySQL in the system development process enables a robust and scalable solution for result verification. PHP's versatility and extensive web-based application support, coupled with the efficient data management capabilities of MySQL, contribute to a seamless integration of the salted hashed functions into the existing system infrastructure.

The adoption of UML diagrams in the system modeling phase enhances the clarity and understanding of the system design, ensuring a standardized representation that facilitates future system maintenance and enhancements.

Objective Three: To evaluate the Secure model for student's results verification.

The development of the Python script allowed for comprehensive testing of the salted hash functions, assessing their resistance against reverse hashing attacks. The failure to reverse hash the already hashed data demonstrated the model's capability to maintain data integrity and verify examination results accurately.

From the evaluation, it is evident that all four hashing algorithms, namely MD5, SHA-1, SHA-256, and SHA-512, exhibit similar performance characteristics when attempting to reverse hash values. When attempting to reverse the values that did not contain the salt the system used an Average Reverse Time of 0.816419 seconds and a total time of 3.265677. While when the same evaluation was conducted on the values which contained a salt the system used an Average reverse time of 0.816419 seconds and a Total time take for 3.265677 seconds.

The time taken to reverse the hash values is relatively consistent across these algorithms, with MD5, SHA-1, and SHA-256 all showing close timings, and SHA-512 taking slightly longer. This

suggests that the choice of algorithm alone does not significantly impact the time required to reverse a hash value when the original input is not known.

Based on these findings, it can be concluded that the secure model, with its implemented salted hash functions, provides a reliable and secure solution for student's results verification. The model's effectiveness in maintaining data integrity reinforces its potential for real-world implementation within the context of securing examination results.

Overall, the evaluation phase affirmed the successful achievement of the objective to evaluate the secure model and validating its capabilities and providing confidence in its ability to ensure the integrity and security of student's examination results.

5.4 RECOMMENDATION

It is recommended to further enhance the key management practices associated with salted hash functions. Proper management and storage of the salts are crucial for maintaining the security of the encrypted data. It is important to establish robust procedures and mechanisms for generating and distributing salts, as well as securely storing them. This includes employing strong encryption techniques and access controls to protect the salts from unauthorized access.

Furthermore, future research and development efforts should focus on exploring advanced encryption algorithms and techniques. While salted hash functions provide a strong level of security, advancements in encryption algorithms can offer additional layers of protection against emerging threats. Exploring techniques such as asymmetric encryption, multi-factor authentication, or homomorphic encryption can further enhance the security and privacy of examination results.

Lastly, it is recommended to engage in collaborations and knowledge-sharing initiatives with other educational institutions and researchers in the field. By exchanging experiences, best practices, and lessons learned, the collective knowledge can be leveraged to develop more robust and efficient systems for securing examination results. Collaborative efforts can also lead to the

establishment of industry standards and guidelines, fostering a unified approach towards data security in the educational domain.

5.5 CONTRIBUTIONS TO KNOWLEDGE

The primary contributions of this study are the exploration and validation of the effectiveness of salted hash functions in securing examination results. By employing strong encryption algorithms and unique salts for each student record, the study has demonstrated the robustness of this approach in maintaining the confidentiality and integrity of result data. This contribution strengthens the theoretical foundation of utilizing salted hash functions as a reliable and secure method for protecting sensitive information in educational contexts. The salted hash helped to improve on the security of the system by making it very difficult for the attacker to reverse the hashes hence which may reveal the original data.

The development and implementation of the prototype system also contribute to practical knowledge in the field. The prototype serves as a tangible demonstration of how salted hash functions can be integrated into the result management process, offering educational institutions a viable solution for securing examination results. The system architecture, design considerations, and implementation strategies provide valuable insights into the practical aspects of incorporating salted hash functions, facilitating the adoption and implementation of similar systems in real-world settings.

The results got from evaluation offer valuable insights into the system's hardened security. This contribution aids decision-makers in assessing the feasibility and practicality of implementing similar systems and highlights areas for improvement and further research.

Additionally, this study contributes to knowledge by identifying and addressing challenges associated with the use of salted hash functions to secure examination results. The study highlights the complexity of salt management, the trade-off between security and performance, user acceptance, and scalability considerations. By recognizing these challenges and proposing strategies to mitigate them, this research provides valuable guidance for future studies and practical implementations in the field.

5.6 AREAS FOR FUTURE RESEARCH

One area for future research is the exploration of more advanced and sophisticated encryption algorithms. While salted hash functions provide a strong level of security, emerging encryption techniques such as elliptic curve cryptography, lattice-based cryptography, or post-quantum cryptography offer potential enhancements to data protection. Investigating the applicability and effectiveness of these algorithms in securing examination results can contribute to the development of even more robust and resilient systems.

Additionally, future research can explore the integration of salted hash functions with other educational systems and platforms. Examination results are often interconnected with other student information systems, learning management systems, or administrative databases. Investigating the seamless integration of salted hash functions across these systems can contribute to a more holistic and comprehensive approach to data security in educational institutions.

References

- Ashis, K. S. (2020). A Blockchain-Based Smart Contract Towards Developing Secured University Examination System. *Journal of Data, Information and Management* (2021) 3:237–249.
- Ashis, K. S. (2021). A Blockchain-Based Smart Contract Towards Developing Secured University Examination System. *Journal of Data, Information and Management*, 1-13.
- Bhandari, A. (2017). Enhancement of MD5 Algorithm for Secured Web Development. *J. Softw.*, vol. 12, no. 4, 240–252.
- Boonkrong, J. Z. (2016). Dynamic salt generating scheme using seeds warehouse table coordinates. *2015 IEEE 2nd Int. Conf InformationScience Secur. ICISS 2015*.
- Brandão, P. R. (2018). The Importance of Authentication and Encryption in Cloud Computing Framework Security. *Int. J. Data Sci. Technol* vol. 4, no. 1.
- Brown, W. S. (2015). Computer Security: Principles and Practice Global Edition.
- Chowdhury, E. M. (2017). Salty Secret: Let us secretly salt the secret. *Proc. 2017 Int Conf. Networking, Syst. Secur. NSysS 2017*, 115–123.
- Diro, A. A. (2017). Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing. *Mobile Networks and Applications*.
- Handschuh, H. G. (2018). Security analysis of SHA-256 and sisters.
- Hornby, T. (2021). Salted Password Hashing - Doing it Right.
- Huang, X. (2016). A security framework for the Internet of Things. *Security and communication networks.*, 9.
- Kasgar, A. K. (2013). A Review Paper of Message Digest 5 (MD5). *Int. J. Mod. Eng. Manag Res.*, vol. 1.
- Khovratovich, D., Rechberger, C., & Savelieva, A. (2015). Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family.
- Lister, M. (2019). A Secure Model for Storage and Dissemination of Examination Results: A Case Study of Zambia Technical Education Vocational and Entrepreneurship Training Authority. *Journal of Computer Science*.

- Lister, M. (2019). Web and Mobile Examination Results Dissemination and Verification System Using Encryption and Cryptographic Hash Functions: A Case of TEVETA. *International Journal of Future Computer and Communication*, 1-8.
- Lister, M. (2019). Web and Mobile Examination Results Dissemination and Verification System Using Encryption and Cryptographic Hash Functions: A Case of TEVETA. *International Journal of Future Computer and Communication*.
- Martinez, V. G. (2020). Analysis of the cryptographic tools for blockchain and bitcoin.
- Mirvaziri, H. (2017). A new hash function based on combination of existing digest algorithms. *2007 5th Student Conf. Res. Dev.*, 1–6.
- Naveen, J. (2018). MULTI-FACTOR AUTHENTICATION SCHEME FOR ONLINE EXAMINATION. *3rd International Conference on Recent Trends in Engineering and Technology*.
- Pittalia, P. P. (2019). A Comparative Study of Hash Algorithms in Cryptography.
- Puthal, D., Nepal, S., Ranjan, R., & Chen, J. (2019). A synchronized shared key generation method for maintaining end-to-end security of big data streams.
- Quadri, S. Q. (2016). Information Availability: An Insight into the Most Important Attribute of Information Security. *J. Inf. Secur.vol. 07, no. 03*.
- R. Mohanty, N. S. (2010). "A secured Cryptographic Hashing Algorithm," Analysis.
- Raikwar, M. (2019). SoK of Used Cryptography in Blockchain. *EEE Access*, vol. 7.
- Raza, S., Seitz, L., Sitenkov, D., & Selander, G. (2016). S3K: scalable security with symmetric keys DTLS key establishment for the Internet of things. *IEEE Transactions on Automation Science and Engineering.*, 13.
- Sahu, A. (2018). Review Paper on Secure Hash Algorithm With Its Variants. *International Journal of Technical Innovation in Modern Engineering Science (IJTIMES)*, pp. 0–7.
- Samuel, K. S. (2021). Gaps in Examination Records Management Procedures: The Reasons for Missing Marks Syndrome in Universities in Kenya. *International Journal of professional Practice (IJPP)*, 1-11.
- Sharma, A. K. (2018). ttacks on Cryptographic Hash Functions and Advances. 89–96.

- Shivraj, V. L. (2015). One time password authentication scheme based on elliptic curves for Internet of Things (IoT). *In Information Technology: Towards New Smart World (NSITNSW)*, 1-6.
- Surbhi, A. (2017). A review of Comparative Study of MD5 and SHA Security Algorithm. *International Journal of Computer Applications* (0975 – 8887).
- Sutriman, B. S. (2019). Analysis of Password and Salt Combination Scheme To Improve Hash Algorithm Security. *(IJACSA) International Journal of Advanced Computer Science and Applications*.
- Vucinic, M. (2015). OSCAR: Object security architecture for the Internet of Things. *Ad Hoc Networks*.
- Wang. (2016). Preimage attack on hash function RIPEMD.
- Wang. (2018). Cryptographic primitives in blockchains. *J. Netw. Comput. Appl.*, vol. 127, 43–58.

APPENDICES

APPENDIX A

Research Instruments used

Questioner for the Students

1. Course of study e.g., BBA, BAE, BIT

2. Year of study

3. Have you had any missing marks since you began your study?

a. Yes

b. No

4. If the above answer is yes, state in how many units you have faced the same problem?

a. 1- 2

b. 3-5

c. 5 and above

5. In case the missing marks have been repeatedly in a single course unit, please indicate it.

6. Give your thoughts about the possible causes of missing marks

7. Have you experience issues where your marks are changed in the Examination Results Database unexpectedly?

- a. Yes
- b. No

8. Suggest ways of improvement

QUESTIONER FOR THE ADMINISTRATORS

Dear Sir/Madam,

Re. SURVEY ON INTEGRATION OF IMPROVED EXAMINATIONS RESULTS MANAGEMENT SYSTEM TO MANAGE ISSUES IN REGARDS WITH MISSING MARKS.

I am carrying out research to identify factors that lead to missing marks in Kampala International University. You have been selected and requested to participate in this survey by giving your opinion freely and honestly on the basis of your personal experience. Any information you will

provide shall be treated confidential and used for academic purpose only and no information of such kind shall be disclosed to others. Joshua Tumusiime (0772391345)

1. Position of the Respondent in the Organization

- a. Head of Department
- b. Principle dean
- c. Associate Dean
- d. ICT Technical support personnel
- e. Any other (Specify).....

2. Indicate how often you attend to concerns of missing marks.

- a. Daily
- b. At the Beginning of the Semester (First Month)
- c. Mid Semester
- d. Once or Twice in a week
- e. Other:
Specify_____

3. What methods do you use to resolve the students missing marks?

- a. Manual System (Use of Paper Work)
- b. An Electronic System that Stores Students Marks.
- c. If Other please specify:

4. Do you think it would be convenient for you and the students to submit complaint forms online?

a. Yes

b. No

5. What is the commonest factor that contributes to missing marks?

a. Lecturers don't want to submit marks in given time.

b. Students claim to have done an exam and yet they did not write it.

c. The results go missing unexpectedly.

d. Other, please specify:

6. Do you experience issues where students' marks are changed in the Examination Results Database unexpectedly or due to some human errors when filling in the data?

a. Yes

b. No

7. Do you think it would be appropriate for a student to receive an email or SMS when their examination results have been resolved?

a. Yes

b. No

8. Given your experience and day to day interaction with the system, kindly suggest possible ways or functionalities that would help improve the current approach to missing marks management.

INTERVIEW QUESTIONS

1. Do you use any intrusion detection and prevention system to stop human insider threats and if yes, what type of systems do you use?

2. Is there any back up (online or offline) that you make for the data and if yes, how often do you make these backups?

3. What type of user authentication do you use in order to access the system and what access control measures do you use to prevent the wrong people from accessing the system?

4. Does the System provide Event Logs to help monitor all the activities that are performed by the user and do you review server logs periodically to monitor any suspicious activities that may be performed on the system?

5. Do you have any system recovery software that can be used to repair the data after a disaster or crash?

6. Are users allowed to access the systems from their own personal devices other than the institutional computers?

7. Are background checks performed on users before they are given the credentials to access the system?

8. Are the systems audited for vulnerabilities once in a while and if yes, does the university collaborate with any with any cyber security organization to ensure that there is proper security of information in the systems?

9. Does the university provide cyber security training to the users who access the system?

EVALUATION TOOLS

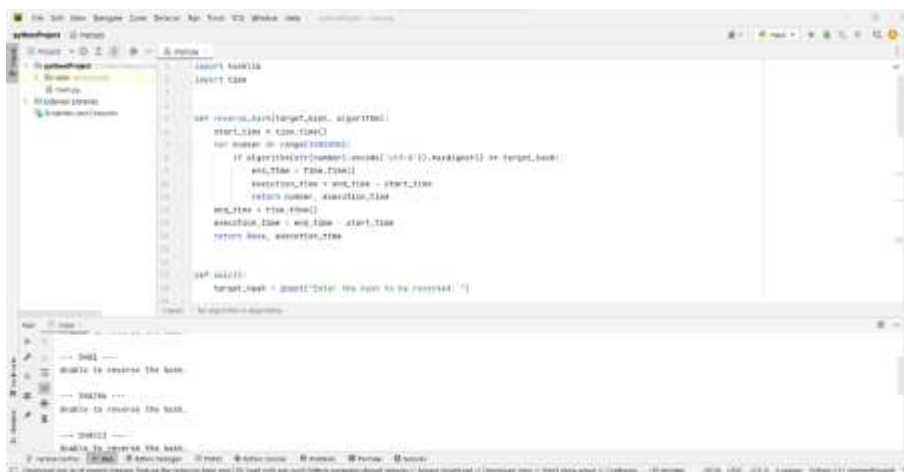
The tools that were used for the evaluation phase include:

Python version 3.11

Python is a high-level, interpreted programming language known for its simplicity and readability. It makes it easier to write code for many things include testing and evaluating functionalities for prototypes.

PyCharm version 2022.3.1

PyCharm is an integrated development environment (IDE) specifically designed for Python programming. PyCharm offers a wide range of features to assist programmers in writing, testing, and debugging Python code more efficiently.



APPENDIX B

Codes for the Secure model system

The code below is used to display the results from the database.

```
<!DOCTYPE html>

<html>

<head>

    <title>Examination Results</title>

</head>

<body>

<h2>Examination Results</h2>

<form method="GET">

    <label for="filter">Filter by:</label>

    <select name="filter">

        <option value="registration_number">Registration Number</option>

        <option value="subject">Subject</option>

        <option value="year">Year</option>

        <option value="semester">Semester</option>

    </select>

    <input type="text" name="value" placeholder="Filter value">

    <input type="submit" value="Filter">

</form>

<div style="text-align: center; margin-top: 20px; margin-bottom: 30px;">
```

```
<a href="input.php" style="display: inline-block; background-color: #007bff; color: #fff; padding: 10px 20px; border-radius: 4px; text-decoration: none;">Add Results</a>
```

```
</div>
```

```
<?php
```

```
// Assuming you have a database connection
```

```
$connA = new mysqli("localhost", "root", "", "dba");
```

```
$connB = new mysqli("localhost", "root", "", "dbb");
```

```
// Check connections
```

```
if ($connA->connect_error || $connB->connect_error) {
```

```
    die("Connection failed: " . $connA->connect_error . " or " . $connB->connect_error);
```

```
}
```

```
$filter = isset($_GET['filter']) ? $_GET['filter'] : "";
```

```
$value = isset($_GET['value']) ? $_GET['value'] : "";
```

```
$sql = "SELECT * FROM exam_results";
```

```
if (!empty($filter) && !empty($value)) {
```

```
    $sql .= " WHERE $filter = '$value'";
```

```
}
```

```
$result = $connA->query($sql);
```

```
if ($result->num_rows > 0) {
```

```
    echo "<table>";
```

```
    echo "<tr><th>Registration
```

```
Number</th><th>Subject</th><th>Year</th><th>Semester</th><th>Mark</th><th>Validation
</th></tr>";
```

```

while ($row = $result->fetch_assoc()) {

    $id = $row["id"];

    $registration_number = $row["registration_number"];

    $hashed_mark = $row["hashed_mark"];


    // Retrieve encryption key and salt from database B based on registration number

    $sqlB = "SELECT encryption_key, salt FROM encryption_keys WHERE id = '$id'";

    $resultB = $connB->query($sqlB);


    if ($resultB->num_rows > 0) {

        $rowB = $resultB->fetch_assoc();

        $encryption_key = $rowB["encryption_key"];

        $salt = $rowB["salt"];

        // Decrypt and validate the hashed mark

        $decrypted_mark = openssl_decrypt($row["encrypted_mark"], 'aes-256-cbc',
$encryption_key, 0, "eabpe?!aY43][5+a");

        $hashed_decrypted_mark = hash('sha256', $decrypted_mark . $salt);


        echo "<tr><td>" . $row["registration_number"] . "</td><td>" . $row["subject"] .
"</td><td>" . $row["year"] . "</td><td>" . $row["semester"] . "</td><td>" . $decrypted_mark .
"</td>";


        if ($hashed_mark === $hashed_decrypted_mark) {

            echo "<td class='valid'>Valid</td></tr>";

        } else {

```

```

        echo "<td class='invalid'>Invalid</td></tr>";
    }
} else {
    echo "<tr><td colspan='6'>Error: Encryption key not found</td></tr>";
}
}
echo "</table>";
} else {
    echo "<h1>No results found in the database.</h1>";
}
$connA->close();
$connB->close();
?>
</body>
</html>

```

Python Code for testing the model

```

import hashlib
import time

def reverse_hash(target_hash, algorithm):
    start_time = time.time()
    for number in range(1000000):
        if algorithm(str(number).encode('utf-8')).hexdigest() == target_hash:
            end_time = time.time()

```

```

        execution_time = end_time - start_time
        return number, execution_time
end_time = time.time()
execution_time = end_time - start_time
return None, execution_time

def main():
    target_hash = input("Enter the hash to be reversed: ")

    # List of hashing algorithms to evaluate
    algorithms = [
        hashlib.md5,
        hashlib.sha1,
        hashlib.sha256,
        hashlib.sha512
    ]

    print("Attempting to reverse the hash...\n")

    total_reverse_time = 0
    results = []

    for algorithm in algorithms:
        algorithm_name = algorithm().name.upper()
        print(f"--- {algorithm_name} ---")

        original_input, reverse_time = reverse_hash(target_hash, algorithm)
        total_reverse_time += reverse_time

```

```

if original_input is not None:
    print(f"Original Input: {original_input}")
    print(f"Reverse Time: {reverse_time:.6f} seconds")
else:
    print("Unable to reverse the hash.")

results.append((algorithm_name, reverse_time))
print()
average_reverse_time = total_reverse_time / len(algorithms)
total_time = total_reverse_time

print(f"--- Performance Evaluation ---")
print(f"Average Reverse Time: {average_reverse_time:.6f} seconds")
print(f"Total Time Taken: {total_time:.6f} seconds")
print("\n--- Time Taken for Individual Hashing Algorithms ---")
for result in results:
    print(f"{result[0]}: {result[1]:.6f} seconds")

if __name__ == '__main__':
    main()

```



**KAMPALA
INTERNATIONAL
UNIVERSITY**

Ggaba Road, Kansanga * PO BOX 20000 Kampala, Uganda
Tel: 0709654233/0774393791 Fax: +256 (0) 41 – 501974
E-mail: dhdrinquiries@kiu.ac.ug * Website: <http://www.kiu.ac.ug>

**Directorate of Higher Degrees and Research
Office of the Director**

Our Ref. 2021-01-03178

Tuesday 4th April, 2023

Dear Sir/Madam,

RE: INTRODUCTION LETTER

REG. NO. 2021-01-03178

The above mentioned student is a student of Kampala International University pursuing a Master's Degree in Information Technology.


The student is currently conducting a research study titled, "*A Secure Model for Student Results Verification Using Salted Hash Functions*".

Your organization has been identified as a valuable source of information pertaining to the research subject of interest. The purpose of this letter therefore is to request you to kindly cooperate and avail the student with the pertinent information needed. It is our ardent belief that the findings from this research will benefit KIU and your organization.

Any information shared with the researcher will be used for academic purposes only and shall be kept with utmost confidentiality.

I appreciate any assistance rendered to the researcher

Yours Sincerely,


Prof. Israel O. Oburoh
Director

C.c. DVC Academic
Dean SOMAC



KANSANGA, UGANDA

tbabuuli@gmail.com

+256772391345

4th JULY, 2023

THE WEBSITE ADMINISTRATOR

KAMPALA INTERNATIONAL UNIVERSITY,

P.O. BOX. 20000,

KAMPALA, UGANDA

Dear Sir,

RE: REQUEST FOR DATA COLLECTION FOR MASTER'S DISSERTATION RESEARCH

I am writing to formally request your assistance in collecting data for my Masters dissertation research project titled "A Secure Model for Student Results Verification Using Salted Hash Functions." As a student at Kampala International University pursuing a Masters degree in Information Technology, I am eager to conduct this research and make a meaningful contribution to the field.

The objective of my study is to develop a secure model for verifying student results using salted hash functions. This research aims to enhance the authenticity and integrity of student academic records, addressing potential security vulnerabilities and ensuring accurate verification processes.

I kindly request you to fill in the questionnaire I have provided which will gather insights on how the current system operates. The data collected will be crucial in understanding existing practices and identifying areas for improvement in terms of security and efficiency.

To ensure the confidentiality and anonymity of participants, I will implement the following measures:

1. All responses will be kept strictly confidential and used solely for the purpose of this research.
2. Personally identifiable information (PII) will not be collected. Responses will be analysed in a way that ensures anonymity.
3. The data collected will be stored securely and will only be accessible to me as the researcher.

I am committed to complying with any requirements or guidelines set forth by KAMPALA INTERNATIONAL UNIVERSITY regarding data protection and research ethics. I am open to discussing any additional measures or concerns you may have to ensure the smooth and ethical execution of this research project.

Thank you for considering my request. I am eager to collaborate with KAMPALA INTERNATIONAL UNIVERSITY and contribute to the advancement of knowledge in the field of Information Technology.