Strengthening the MD5 File Integrity Algorithm with User Fingerprint

تعزيز أداء خوارزمية سلامة الملفات MD5 باستخدام بصمة المستفيد

Prepared by

Marwa Hussein Issa Al-Awawdeh

Supervisor:

Dr. Mudhafar Al-Jarrah

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

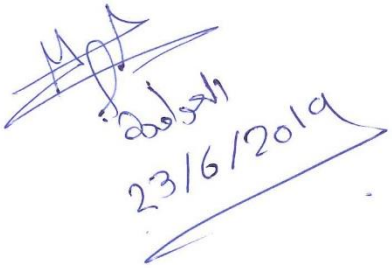
June, 2019

Authorization

I Marwa Hussein Issa Al- Awawdeh authorize Middle East University to provide and electronic copies of my thesis to the libraries, organization, or bodies and institutions concerned in research and scientific studies upon request.

Name: Marwa Hussein Issa Al- Awawdeh

Date: 3/6/2019

Signature: 

Discussion Committee Decision

This is to certify that the thesis entitled "Strengthening the MD5 File Integrity Algorithm with User Fingerprint" was successfully defended and approved on 3/6/2019.

Examination Committee Members	Signature

(Supervisor)

Dr. Mudhafar Al-Jarrah

Associate Professor, Department of Computer Science

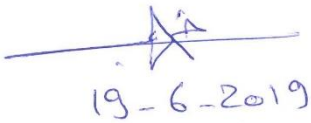
Middle East University

(Chairman of Examination Committee and Internal Committee Member)

Prof. Hamza A. Al-Sewadi

Professor, Department of Computer Science

Middle East University

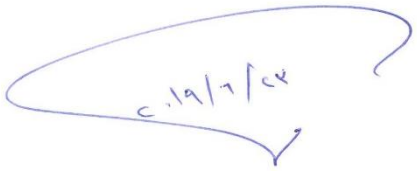


(External Committee Member)

Prof. Mohammad Ahmad Alia

Professor, Department of Computer Science

Al Zaytoonah University



Acknowledgment

All praise and thanks are due to the Almighty Allah who always guides me to the right path and has helped me to complete this thesis. There are many people whom I have to acknowledge for their support, help, and encouragement during the journey of preparing this thesis. So, I will attempt to give them their due here, and I sincerely apologize for any omissions.

First and foremost, I would like to record my heartfelt gratitude to my supervisor Dr. Mudhafar Al-Jarrah for his skillful supervision, advice and patient guidance throughout the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways. I am really indebted to him more than he knows.

I would also like to express my deepest gratitude to all the respectable lecturers at the Faculty of Information Technology, Middle East University and special thanks to the Graduate School Office and the Library Staff and for everyone who supported me especially my colleague to accomplish this work.

The Researcher

Dedication

To:

My parents who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

My beloved brothers and sisters for their love and kindness, endless support and encouragement.

Everyone gives me support.

I dedicate this thesis.

The Researcher

Table of Contents

List of Tables

List of Figures

List of Appendixes

Table of Abbreviations

DES	**Data Encryption Standard**
3-DES	**Triple Data Encryption Algorithm**
LCM	**least common multiple**
MD4	**Message Digest 4**
MD5	**Message Digest 5**
MDM	**Message Digest Modification**
RC4	**Complement-Receptor Type 4**
RC5	**Complement-Receptor Type 5**
RFC	**Request For Comment**
RSA	**Rivest, Shamir, and Adelman**
SHA256	**Secure Hash Algorithm 256**

Strengthening the MD5 File Integrity Algorithm with User Fingerprint

Prepared by

Marwa Hussein Issa Al-Awawdeh

Supervisor

Dr. Mudhafar Al-Jarrah

Abstract

The MD5 (Message Digest 5) is one of the algorithms used in digital signature processes as well as password protection to secure the integrity of both data source and the file manipulates. Over the past few years, the weakness in the original algorithm MD5 has been proved, The MD5 algorithm has an equal speed of SHA256 (Secure Hash Algorithm 256) where these algorithms differ in terms of security and execution speed to produce the hash value. But it is limited in terms of proving such an occurrence of a collision attack, which limited its Performance, and thus seeking to improve security in the MD5 becomes a mandatory requirement.

This thesis presents a new version of the algorithms used in Hash operations called MDM (Message Digest Modification), Its main goal is to improve the security level in MD5 and solve the problem of the main vulnerability by using variable values, which are not fixed in the original algorithm and are taken from the fingerprint, which maintaining the basic structure of the original algorithm. Also the thesis contains the comparisons of the implementation time difference to illustrate the strength of the new algorithm.

Then an adequate explanation and summary of the original MD5 algorithm is made, and also the weak point is identified in order to determine the general weakness of the algorithm. After that the proposed modifications are clarified and applied to produce a new MDM algorithm and compared with the other model as well as the algorithm currently adopted in SHA256 in terms of safety and execution time and 128bit out of message digest. The results indicate the strength of the new algorithm, and its significantly higher speed, even while using different file sizes. So, it is recommended to use the new MDM algorithm in applications requiring greater security and faster execution, for digital signature, verification and password protection.

Keywords: digital signature algorithms, authenticity, MD5, signature verification time complexity, cryptography, hash functions.

تعزيز أداء خوارزمية سلامة الملفات MD5 باستخدام بصمة المستفيد

إعداد

مروه حسين عيسى العواوده

المشرف

د. مظفر الجراح

الملخص

تعد MD5 هي إحدى الخوارزميات المستخدمة في عمليات التوقيع الالكتروني وحماية كلمات المرور لضمان صحة مصدر البيانات وسلامة الملفات من التعديلات. على مدار السنوات الماضية أثبت وجود ضعف في الخوارزمية الاصلية مما أتاح المجال لخوارزميات اخرى التنافس على المركز الأول مثل خوارزمية SHA256 والعديد غيرها التي صنفت تبعا لقوتها وسرعتها، حيث تختلف هذه الخوارزميات من حيث الأمان وسرعة التنفيذ لإنتاج قيمة التجزئة، تتمتع خوارزمية MD5 بسرعه مساويه لخوارزمية SHA256 ولكنها أضعف من ناحية إثبات وجود هجوم التصادم مما أضعف أدائها، وبالتالي يتم السعي إلى تحسين مستوى الأمان في خوارزمية MD5.

تقدم هذه الرسالة نسخه جديده من الخوارزميات المستخدمة في عمليات الهاش تسمى خوارزمية MDM. الهدف الرئيسي منه هو تحسين مستوى الأمان في خوارزمية MD5 وحل مشكلة الضعف الرئيسية عن طريق استخدام قيم متغيرة غير الثابتة في الخوارزمية الأصلية وتؤخذ من بصمة الإصبع، مع الحفاظ على البناء الأساسي للخوارزمية الأصلية، تحتوي الرسالة على مقارنات لفرق وقت التنفيذ لتوضيح قوة الخوارزمية الجديدة.

يتم إجراء توضيح كافي وموجز عن الخوارزمية الأصلية MD5 ثم تحديد نقطة الضعف فيها، لمعرفة الضعف العام للخوارزمية، تتم بعدها توضيح التعديلات المقترحة وتطبيقها لإنتاج خوارزمية جديده MDM ومقارنتها بالنموذج القديم والخوارزمية المعتمد حاليا في عمليات الهاش خوارزمية SHA256

من حيث الأمان والوقت المستغرق بالتنفيذ. تشير النتائج الى قوة الخوارزمية الجديدة وسرعة تنفيذ أقل وبشكل ملحوظ، باستخدام أحجام ملفات مختلفة. لذلك، يوصى باستخدام الخوارزمية الجديدة MDM في التطبيقات التي تحتاج أمانا أكبر وسرعة تنفيذ أقل للتوقيع والتحقق وحماية كلمات المرور.

الكلمات الرئيسية: خوارزميات التوقيع الرقمي، الموثوقية، MD5، تزمن التحقق من صحة التوقيع، التشفير، وظائف التجزئة.

Chapter One

Background and the study Importance

CHAPTER ONE

Background and the study Importance

1.1 Introduction

The increasing use of information systems, the proliferation of technology that makes users more dependent on the computer and the digital network have all revealed new risks to computer system security. The traditional way of providing security fails to keep pace with the dangers; as a result, researchers are looking for a new ways to provide the highest levels of security, to maintain the integrity of information and the absence of any manipulation. At present, there are many mechanisms to ensure that information is fully accessible that there is no change on their content, and also are currently dealing with ways to confirm the identity of the sender as well as the digital signature.

A digital signature is a mathematical scheme to validate digital messages or documents (Paul, 2017). The digital signature is used to confirm that the sender has created the signature to confirm his identity, so that the sender cannot deny that he has sent the message (not to repudiate) and that the message has not been changed during the transfer (integrity). A digital signature is a standard element of most cryptographic protocol suite and is commonly used for software distribution, financial transactions, and contract management software, in these cases, where it is important to detect forgery or tampering (Schaettgen. N, Levy. D & Schelnast. J, Socol. S, 2014).

One of the cryptographic protocols, commonly used, is the digital signature MD5 protocol. The MD5 protocol was initially designed to be used as a cryptographic hash function, producing a 128-bit hash value; it has been found to suffer from extensive vulnerabilities, but it is still widely used as a checksum function to verify data integrity after downloads or transfer of data. The cryptographic hash function has a basic requirement in, that it should be computationally infeasible to find two non-identical messages with the same hash value.

MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function MD4 and was specified in 1992 as RFC 1321(Request for Comments). The security of the MD5 hash function is severely compromised due to collision attack; there are also chosen-prefix collision attacks that can produce a collision for two inputs with specified prefix within hours. These attacks have been demonstrated in public in many various situations, including conflicting documents and digital certificates (Guneysu. T, Paar. C & Schage. S, 2018).

As of 2015, MD5 was demonstrated to be still quite widely used, most notably by security research and antivirus companies. Hence, with the weakness of the MD5 is still used; the problem addressed in this thesis depends on this, and explains how to make MD5 more efficient with a high-level security (Stevens, 2007).

The Message Digest 5 algorithm (MD5) takes as input message of arbitrary length and produces as output a 128-bit "message digest" of the input. The MD5 algorithm is intended for digital signature application, where a large file must be compressed in a secure manner before being encrypted with a private key under public key cryptosystem such as RSA (Rivest–Shamir–Adleman) (RFC 1321, 1992).

The MD5 algorithm is designed to be quite fast on 16-bit and 32-bit machines and can be extended to 64-bit machines. In addition, the MD5 has not required any large substitution tables; the MD5 can be coded quite compactly (Gupta & Kumar, 2014).

The MD5 algorithm is an extension of the MD4 Message-digest algorithm, Md5 is slightly slower than MD4; because of three rounds in MD4 and MD5 contains four rounds which makes it is slower, it is a one-way hash function that deals with security features, where the MD5 is more conservative in design.

MD4 was perhaps being adopted to be used more quickly than justified by existing critical review, so MD5 was designed to make sure justified the existing critical review. MD4 was designed to be exceptionally fast, but it stays at the edge in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security, both algorithms follow the same concept but with a different architecture (Kuzushko,2003).

The simple XOR hash function does not provide enough security to serve as a digital signature. Tom Berson attempted to use the differential technique of crypto analysis for a single round of MD5 (Berson, 1992). A more successful attack by Den Boer and Bossel produces collision using the compression function in MD5 (Robshaw, 1994). This does not lend itself to attacks against MD5 in practical application and does not affect the use of MD5, it does mean that one of the basic design principles of MD5 – to design a collision-resistant compression function – has been violated although it is true that "There seems to be a weakness in the compression function "(Robshaw, 1994).

Hashing algorithms are commonly used to convert passwords into hashes, which theoretically cannot be deciphered. A new approach to using MD5 in password storage

is proposed by using external information; such as fingerprint, a calculated salt; such as username to encrypt the password before the MD5 calculation.

The importance of the internet has become a highly valid environment for all use is, for organization and governments due to providing services and easily dealing with government services. All these services need high technical protection for their user's registration. Hashing algorithms are used to convert passwords into a string called hash values or digests. A hash is also a one-way function which theoretically cannot be reversed or get a plain text from the hash. Salted password hashing means to supply or prepend a random string to the user's password before hashing. To make hashes more secure, salt can be added to the hash. This means that a random string of characters is either prefixed or postfixes to the password before hashing it. Every password has a different salt. Even if the salts are stored on the database, it will be very complicated massive the passwords using a rainbow table as the salted passwords are long, complex and unique. Salted hashes can be brutally forced but the time taken is significantly longer. Using of two salts, one from the fingerprint and one from the password that user uses, can also protect them as password against offline attacks.

1.1.1 Fingerprint recognition

Recognition of persons by means of biometric distinguishing is an emerging phenomenon in the epochal community. It has a lot of turnout during the present time due to the requirement for security of the application. Through the many biometric features, the fingerprint is considered one of the most practical ones. The fingerprint recognition demands the least potential from the user, provides comparatively good performance to capture the necessary information for the recognition process.

The main reason behind using the fingerprint is the comparatively low price of fingerprint sensors, which enables easy integration into PC keyboards. (Maltoni. D, Maio. D & Jain. A, Prabhakar. S, 2009).

In order to access the internet or any other resource safely, a high-security authentication system is essential. (Florenco. D, Herley. C, 2007). Some studies based on general research on the use of passwords and user problems with the forgetting of the words that have been used based on a special key on the memory of the user. The new protocol will be affected if a special word in the build of the Encoding or Decoding is used. Even if the user goes through using a fingerprint, both encoding and decoding do not go over the user to remember the words that is used.

1.2 Problem Statement

The MD5, is one of the most popular hash protocols, which is commonly used to check for file integrity, even with the known weakness of the algorithm. The problem addressed in this thesis is to make the MD5 more robust to collision attacks. This thesis assumed that it is possible to strengthen the original MD5 and to make some updates on the round 4 of the algorithm, by modifying the algorithm through adding additional steps and features that are amid to deal with the weaknesses.

As has been noted in many studies that go through the improving of the weaknesses on the MD5. Libed. J, Sison. A & Dr.Medina. R, (2018). They pointed out what the study supports through MD5 cryptographic hash function is affected by collision attacks. Also they pointed out that security of the MD5 will be affected because of the collision attack, because if the security has been affected the file integrity, as it is the main work

for the MD5, will be affected and will give the same value for the message digest and take it surely.

Libed. J, Sison. A & Dr.Medina. R, (2018) showed that collision attacks of MD5 cryptographic protocol affects the data integrity and authenticity of the message digest (Hash value).

Several studies have demonstrated the vulnerability within the protocol, and many studies have implemented solutions such as improvements in collision values or imposing greater volume of special values for use such as salt, some of which indicate the use of a chain code, all of which have improved user benefit.

This study refers to essential improvement mode that greatly improves the safety for using, and prevents the weakness within the current protocol to be improved and made more powerful and more efficient to be used safely.

1.3 Goal and objectives

The goal of this research is strengthening the original MD5 protocol by developing a new protocol that deals with the weaknesses of the existing protocol. The new protocol will be in a high-level of security which depends on a unique value that will be hard to have a message with the same digest or same hash value. In the attempt to achieve the goal, the following objectives are taken into account:

• Identify, the weak points in the MD5 protocol.

• Investigate technical solutions to deal with the identified weaknesses, and then update the MD5 protocol, using new features that are supported by the original protocol.

• Implement the protocol in technical solutions.

• Test robustness of the new file integrity checksum protocol through practical experimentation using the same practical method that was used with the original MD5.

• Modify and evaluate the new protocol based on the outcome of the experimental work.

- Test applicability of using the new MD5 in a business environment.
- Develop a new protocol using Asymmetric techniques.
- Verify the efficacy of the new protocol.

1.4 Research questions

• What are the weaknesses in the MD5 that need to be dealt with?

• What is the technical solution that will deal with the identified weaknesses in the existing protocol?

• How to implement the new technical solution to deal with the identified weaknesses in the existing protocol?

- What are the results of evaluating the robustness of the new protocol?
- What the results are of implement the technical solution in the new protocol?
- What are the results of test robustness through practical experimentation?
- What are the results of verifying the efficacy of the new protocol?

1.5 Motivation

Data in transit can suffer from alteration by accident or intention, hence the need for integrity checking tools to verify that received file is identical to the original file. The most widely used file checksum protocol is MD5, which although it is accepted as the default protocol, this context has the weaknesses of possible collision attacks that give the same message digest for different file contents.

MD5 is still popular software that is used through an application such as password hashing, although it is not simple/plain MD5 and probably uses advanced techniques such as Salt, Key stretching or the chain code. MD5 seems like the professional and popular programs with many years of development .

Google Drive is still using MD5Sum for identification where security is not a factor under consideration.

1.6 Expected contribution

The expected contribution of this study will be in the following:

- Make a file integrity protocol available that is resilient to collision attacks in comparison with the existing MD5.
- Utilize the fingerprint data to enhance the file integrity checksum protocol.
- Apply and give evidence on using the new protocol on everyday used Microsoft office files.
- Demonstrate the effectiveness of using fingerprint or similar mechanisms in the process of building the Hash protocols to make them stronger and more secure.

- Develop new techniques in the hash protocols build as Asymmetric techniques.
- Verify the efficacy of using asymmetric techniques in the security environment of hash protocols use.

Chapter Two

Theoretical Background and Literature Review

CHAPTER TWO

Theoretical Background and Literature Review

2.1 Introduction

In this chapter, at first. It covers the definition of MD5, digital signature concept, password protection, verification and Fingerprint recognition, too. A brief of a comprehensive theoretical background will be described. Then, a literature survey of MD5 variants and others on digital signature, password protection and fingerprint will be presented.

2.2 Theoretical background

The MD5 algorithm is designed to be quite fast on 32-bit machines (operating system). In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. The MD5 algorithm is an extension of the MD4 message-digest algorithm. MD5 is slightly slower than MD4, but is more "conservative" in design. MD5 was designed because it was felt that MD4 was perhaps being adopted to be used more quickly, MD4 was designed to be exceptionally fast. (Rivest. R, 1992).

The MD5 algorithm is considered as one of the hash function’s, which compress an arbitrary length, taken as input message and produced as output a 128-bit “message Digest” of the input. It is supposed that it is computationally infeasible to produce two

messages having the same message digest, or having a collision attack. (Naito.Y, Sasaki.Y & Kuniniro.N, Ohta.K, 2005).

The MD5 algorithm is used in several fields, including digital signature and password protection. It is considered a powerful algorithm in terms of calculating the hash value for two messages despite the many studies that proved the weakness of the algorithm and the possibility of proving two messages that may carry the same value of Hash. (Thomsen.S, 2005).

All hash functions should be secure and fast at the same time, the MD5 algorithm It has some weakness in terms of complete safety which may reduce its use, or user confidence, one of the most famous weaknesses in this algorithm is the collision attack (where this problem is summarized in brief, the existence of two messages have the same hash values) as the other On the technical side, this problem significantly affects the performance of the protocol and weakens the user 's trust, and as a result limiting their use even if the percentage is low, which means that the apprehension of their use will be significant.

2.3 Introduction to cryptography

The development of information security is closely related to encryption and decryption, where the encryption process changes the original content of the message, and the decryption retrieves the original information of the message after it has been encrypted. This process is done using asymmetric key technology where there is a private key for encryption and a public key for decryption. The cryptography provides integrity, validation and non-repudiation, as well as confidentiality (Stallings. W, 2004).

Many cryptographic algorithms rely on the key and its ability as long as the key is secret, it will be difficult to foresee or to be known by third parties other than sender and recipient, thus ensuring the integrity of the file to the sender. Also, the recipient can check the files and the identity of the sender. (Kumar, Satish and Zabeer, 2004).

When relying on encryption and decryption, two systems are used: symmetric and asymmetric. In the case of symmetric, the same key is used in encryption and decryption (I.e.). If (K) and (M) are the key and the message, then we have Dk (Ek (M)) = M …… (1) Where (D) and (E) denote decryption and encryption algorithms. (Stallings. W, 2004) (Kumar, Satish and Zabeer, 2004).

A special advantage of this system is that the speed of performance and safety level is closely related to the strength of key. Examples of algorithms that support symmetric systems are DES, 3-DES, RC4, RC5, etc. However, such systems are not without flaws if they rely on the power of the key in the stage of ensuring safety, but there are still some flaws about how to manage the key-exchange and non-avoidance.

In the case of asymmetry, the key used in this stage is different depending on the public key. These keys are related to their mathematical relationship. The public key is handled and shared between the sender and recipient, while the sender's private key remains and is usually used in the encryption process, i.e. if K1 and K2 are public and private keys, respectively and (M) be the message (Stallings. W, 2004), then:

DK2 (EK1 (M)) = DK1 (Ek2 (M)) = M …… (2)

Systems that support the public key are the most secure systems that support non-repudiation, and ensure not to fall into the problem of how to manage to exchange the key, but as previously stated that no system can be free of defects despite of its strength. Accordingly, after the encryption process produces text larger than the

original text and is relatively slow. Examples of algorithms that support the asymmetric encryption system: RSA and Elliptic curve cryptography.

2.4 Hash function

A hash h is generated by a hash function H of the form

h = H (M) …… (3)

Where (M) is a message of variable length and H (M) is the hash value of fixed length.

A hash function should satisfy the following properties to be useful:

1. A hash function can be applied to a data block of any size.

2. It always produces an output of fixed length.

3. It must be easy and efficient to compute H (x) for any given x. Though the effort depends on the length of x, it should not be a function of its length.

4. One-way: It should not be possible to find x for any given value h, such that h = H (x).

5. Weak collision resistance: Given x, it is computationally infeasible to find $y \neq x$ such that H (y) = H (x) …… (4).

6. Strong collision resistance: It is computationally infeasible to find any pair (x, y) (Stamp. M, Low. R, 2007) such that H(x) = H(y) ……..… (5).

All the inputs in all the Hash functions are divided into a series of n-bit, after which the hash function coupling as one block an m-bit object and the final value is the

value of the Hash value. Hash associations are simple and quick through using bitwise exclusive- OR of every block of bits. (Stallings. W, 2004).

The hash function works to segment the message before being processed so that it is divided into equal lengths and is used Merkle-Damgard construction in most Hash functions, in which the input message (M) is partitioned into (L) blocks of size, as (b) bits. (Kashyap. N, 2006)

The file compression associations are frequently used in the operations of the Hash since they work with two inputs. This process is called chaining value and depends on the previous steps, and (b-bit) where the (n-bit) output is generated from the process, the chaining value has an initial value given by the used algorithm, mostly n<b so .The compression of the bits can be abstract as given below

CV0 = IV = initial value …….... (6)

CVi = f (CVi-1, Yi-1) 1 ≤ i ≤ L ……..... (7)

H (M) = CVL …… (8)

Where the input message M is divided into blocks Y0, Y1…YL-1, as shown in figure 2.1.

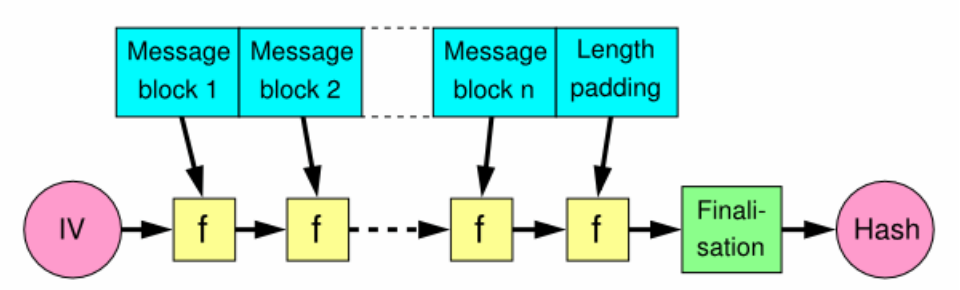


Figure 2.1 hash function (Kashyap. N, 2006)

2.4.1 Application of hash functions

Hash functions are used in various contexts; such as digital signatures, password protection, message authentication codes and as pseudo-random number generators.

2.4.1.1 Digital Signature

Digital signatures are used for several purposes including the following:

To ensure authenticity: The recipient is assured that the message was really sent by the demanded sender.

To avoid repudiation: The sender cannot pretend that he did not sign that message.

Generally, a message is first hashed before signed so as to reducing the size of the signature. (Thomsen. S, 2005).

2.4.1.2 Password Protection

Passwords are stored after hashing instead of plaintext for a clear reason; whenever a password is typed, the afresh computed hash is compared with the existing hash and if it matches, then the password is declared correct. (Thomsen. S, 2005)

2.5 Description of MD5

The MD5 algorithm is an extension of the MD4 message digest. Accordingly, MD5 algorithm is designed to be quite fast on 32-bit computers; because the MD5 algorithm does not require any huge substitution technique, MD5 is slower than MD4 (Rivest.R, 1991). But the MD5 is more conservative in design, where MD5 was designed because of MD4 which was perhaps being designed to be exceptionally fast, but it was risky felt in term successful cryptanalytic attack as shown in figure 2.2. (Naito.Y, Sasaki.Y & Kuniniro.N, Ohta.K, 2005).

The MD5 algorithm has a b-bit message as input, the MD5 algorithm calculated to find message digest, b is an arbitrary non-negative integer, b can be zero, and it need not be a multiple of eight. In new MD5, imagine the bits of the message written as:

M_0 M_1…………………………………………………. $M_{\{b-1\}}$

There are four steps to perform the computation of message digest in MD5:

Step 1. Append padding bits.

Step 2. Append length.

Step 3. Initialize MD buffer.

Step 4. Process message in 16-word blocks.

Step. 1 Append padding bits:

The message must be padded so that the length will be in bit equal to 448 module 512 so that the length of the padding (length = 448 mod 512). So message in 64 bits less than an integer multiple of 512 bits. If the message already has or doesn't have the desired length, the padding is always added as shown in figure 2.2. (Stallings.W, 1999).

Step. 2 Append length:

Before padding, in step one, 64-bit representing from the original message length, so that

In case the length in step 1 is greater than 2^{64}, these bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions. Then, the low-order of the 64 bit will be used. As a result, the outcomes of the two steps describe a message that is an integer equal to 512 bits in length as shown in figure 2.2. (Stallings.W, 1999).

Step. 3 Initialize MD buffer:

This step is standing on build a 128-bit buffer, which is used to hold intermediate and final results of the hash function. This buffer will be splitting into four 32-bit registers (A, B, C, and D), they are represented in hexadecimal values, and these values are represented in little-Indian format as shown in figure 2.2 (Kahate. A, 2008).

Step. 4 Process message in 16-word blocks:

The main work and the implementation of the goal of this study is in this step. This step is a compression algorithm that consists of four-round processing. The fundament of the four rounds that have a similar structure, but each round has a different primitive logical function which refers to as F, G, H, and I in the specification.

Each round takes the 32-bit buffer value ABCD and the current block 512-bit, and each round consists of a 64-element from the lookup table. After all these steps the output is a 128-bit message digest as shown in figure 2.2 (Kahate.A, 2008).

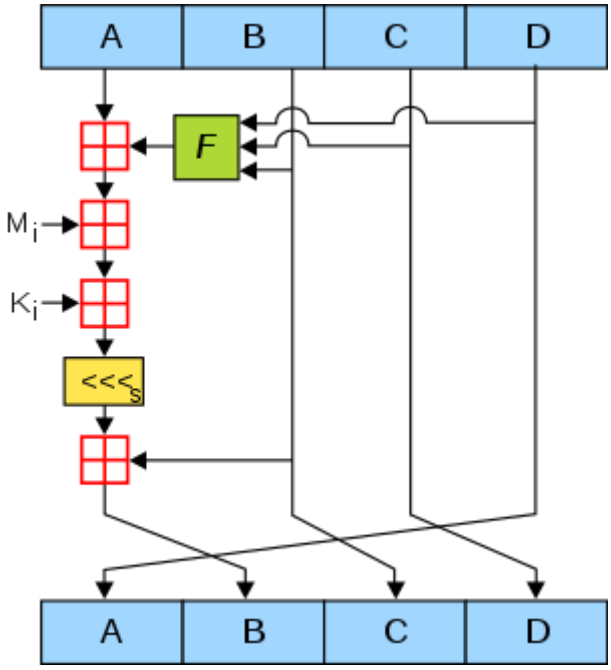


Figure 2.2 the discerption of the MD5 (Stallings. W, 2014)

2.6 MD5 collision attacks

MD5 has been prevailed in a wide variety of security applications, and also is used to check the integrity of files. In 1993, B. den and A. Bosselaers showed a weakness in MD5 by finding a collision attack for consisting messages (the same hash value for two messages with different initial content for each). MD5 compression function (IHV) is an intermediate hash value IHV = (a, b, c, d) and a 512-bit message block (Dobbertin. H, 1996).

It is now proved by Wang and others researchers that MD5 hash is no more secure after they proposed an attack that generates two message-digests, giving the a same MD5 hash value. With a different content for the file, Vlastimil Klima then proposed a more active and fast mechanism to apply this attack, K. Vlastimil used this mechanism to create a collision attack and then used this collision to apply

meaningful collision by making two different files that give congruent MD5 hash value, but each file gives out different contents.

Wang's attack was based on two parts of implementation, the first part to find the first block and implement a Klima's algorithm (Thomsen.S, 2005), and the second part to find the second block using the approach of Wang (Klima.V, 2005). Wang assumed that the two blocks can be found by different mechanism since they are independent of each other, the second value can be evaluated during the first iteration.

Wang's attack used Java language to build a program was run on a desktop computer with AMD 64 3000+ (1.83 GHz) on Windows XP as well as a virtual machine with fedora core 4 on the same computer. Wang found 25 collisions in less than 10 hours with averages out to 24 minutes for each collision (Stevens. M, 2006).

One of the solutions that the researcher proposed is a mechanism to choose the optimal input difference for generating MD5 collision attacks, with the degree of difficulty to satisfy the condition. Second, by utilizing the weaknesses of compression function of MD5. Third, there should be no difference scaling after state word with the distribution of strong conditions for each input with different pattern. Finally, they choose the input difference with the least number of strong conditions and the most number of free message words. (Xie. T, Liu. F & Feny. D, 2013). Some study based on finding a fast attack algorithm to find two – block collision of MD5 hash function, the attack algorithm is based on two-blocks with the differential path. The derived conditions for the desired differential path which it did not hold by using the attack algorithm, it can represent to speed the attack of efficiently up, the MD5 collision attacks can be accomplished within 5 Hours using the computer with Pentium 4 and 1.70 GHz (Lai. X, Liang. J, 2005).

In March 2005, Xiaogang Wang and Hongbo Yu of Shandong University in China found and published an algorithm that can find two different sequences of 128 bytes within the same MD5 hash value. So, the cryptographic MD5 hash function has been broken as the following example; One of the examples on the collision attack:

d131dd02c5e6eec4693d9a0698aff95c2fcab5**8**712467eab4004583eb8fb7f89

55ad340609f4b30283e4888325**7**1415a085125e8f7cdc99fd91dbd**f**280373c5b

d8823e3156348f5bae6dacd436c919c6dd53e2**b**487da03fd02396306d248cda0

e99f33420f577ee8ce54b67080**a**80d1ec69821bcb6a8839396f965**2**b6ff72a70

And

d131dd02c5e6eec4693d9a0698aff95c2fcab5**0**712467eab4004583eb8fb7f89

55ad340609f4b30283e4888325**f**1415a085125e8f7cdc99fd91dbd**7**280373c5b

d8823e3156348f5bae6dacd436c919c6dd53e2**3**487da03fd02396306d248cda0

e99f33420f577ee8ce54b6708**2**80d1ec69821bcb6a8839396f965**a**b6ff72a70

Each of these blocks has MD5 has a same hash value = 79054025255fb1a26e4bc422aef54eb4 with different content but same hash value (Selinger. P, 2011).

2.7 Description of MD4

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest of the input. The MD4 algorithm is intended for digital signature application, where a large file must be “compressed” in a secure manner

before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA (Rivest. R, 1992).

The MD4 algorithm works as an MD5 algorithm, but the MD5 has a step more than the MD4:

Step 1. Append padding bits.

Step 2. Append length.

Step 3. Initialize MD buffer.

Step 4. Process message in 16-word blocks.

Step 5. Output.

The main different between MD5 and the MD4 algorithms, in Step 4 process message, i.e. in 16-word blocks where MD4 defines three auxiliary functions that each takes as input the 32-bit word and produces as output 32-bit word (F, G, H), where the MD5 algorithm defines four auxiliary functions with the same size of bit word input and output.

The MD4 algorithm message digest produced as output A, B, C, D. that begins with low-order byte A, and end with the high-order byte of D, with message digest of size 128-bit.

The major difference between MD4 and MD5 algorithm:

MD4 consists of three passes for each 16-byte chunk of the message. MD5 makes four passes for each 16-byte chunk, the functions are slightly different.

MD4 has two constants, one in passes 2, and another in passes 3. MD5 uses a lookup table (Ti) a different constant, for each message word on each pass (Wiesner. K, 2009).

(Rivest. R, 1992); The MD4 algorithm introduced by Rivest, its algorithm is defined as an iterative application of a three-round compress function. But according to the unpublished attack on the first two rounds of MD4 due to Merkle, and an attack against the last two rounds by Den Boer and Bosselaers (Verlag. S, 1995), Rivest introduced the MD5 as the extension of the MD4 (Rivest. R, 1992).

All hash functions should be secure and fast at the same time. Therefore, the MD4 was a significant contribution introduced by Ron Rivest's in 1990. A short time after MD4 has been introduced, some weaknesses became apparent, and thus he introduced MD5 in 1991 explaining his reasons in:

- The MD5 algorithm is an extension of the MD4 message-digest algorithm.
- MD5 is slightly slower than MD4 but is more "conservative" in design.
- MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risky successful cryptanalytic attack. (Rivest. R, 1992).

2.8 Data integrity checksum

The most significant vulnerabilities in maintaining data integrity occur when being transferred.

The checksum techniques are used in the integrity of the data as it is transferred from its original form, to a new environment that introduces unknown changes to the data.

The user can ensure that the data is correct compared with its original source at its final transformed location.

The checksum is basically a small computing of information about a digital data, usually a file, where checksum is also used to check data after being stored. All this is to verify that the information is still the same as it was before. The computation used to compute the checksum is referred to as the checksum algorithm such in the building of the MD5 algorithm (McClelland. M, 2018).

2.9 Fingerprint recognition

The fingerprint is a vital part for the users who are currently relying heavily on their respective classification processes and verifying user identity biology. (Tulyakov. S, Farooq. F, Govindaraju.V, 2005).

The fingerprint is used by following specific algorithms to detect the number of identical spellings for the user to identify the identity and purpose of the authentication. The values that are produced in the condition of the user re-emphasizing his/her identity, where the new values are compared according to the values previously stored for the same user to give validity to the user's login (Tulyakov. S, Farooq. F, Govindaraju.V, 2005).

The fingerprint biology enables the user to solve the problem of users' lost or forgotten passwords. The fingerprint is one of the oldest biometrics used to prove its power and documentation in data protection. In this research, the fingerprint is treated in terms of digital signature to determine the values in the process of constructing the

hash value, as the repetition of the finger or studying in detail is not relevant to this subject.

In the phase of dealing with password protection, the value of the fingerprint is stored as well as the name of the user in the template, so that the comparison between them occurs when re-logging, since the systems that used in all the experimental work allows the user to repeat the fingerprint in case of rejection, since the values stored in the template answers that match of fingerprint tried.

2.10 Comparative study of Message Digest 5(MD5) and SHA algorithm

Because of the nature of an open document, the integrity of the information as a content of the document is not preserved, which means the document contents can be read and modified by many parties, so that the integrity of a document should be kept. To maintain the integrity of the data, it needs to create a mechanism which is called a digital signature, where there are many hash functions; Two of them are message digest 5 (MD5) and SHA256.

Both algorithms certainly has advantages and disadvantages. The purpose of this section is to define the algorithms as well as the different features between them. The parameters used to compare the two algorithms are the running time, and complexity. The research results obtained from the complexity of the Algorithms MD5 and SHA256 is the same, i.e., O (N), but regarding the speed is obtained that MD5 is better compared with SHA256 (Rachmawawati.D, Tarigan. J, 2018).

2.10.1 Definition of SHA256 algorithm

SHA256 algorithm is one of the successful hash functions to SHA-1, and is one of the robust, hash functions available. SHA-256 is not much more complex to be coded than SHA-1. The 256-bit key makes it a good partner-function for AES "which is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001" as shown in figure 2.1, figure 2.2 and figure 2.3 (Thomas. C, Jose. R, 2015).

Table 2.1 Comparison between MD5 and SHA (Rachmawawati.D, Tarigan. J, 2018)

Features	MD5	SHA 256
Security	Less Secure than SHA	More Secure
Length Of Message Digest	128 Bits	160 Bits
No. Of Attacks Needed To Find Original Message	2^{128} bit operations Required	2^{160} bit operations required
Attacks to try and find two messages producing the same MD	2^{64} bit operations Required	2^{80} bit operations required
Speed	Faster, 60 iterations	Slower, 80 iterations
Successful attacks so far	Attacks reported some extend	No such attack reported

Table 2.2 Similarities between MD5 and SHA Algorithms (Rachmawawati.D, Tarigan. J, 2018)

Similarities	MD5	SHA
Padding	✓	✓
Message bits	✓	✓
Members of hash family	✓	✓
Resource utilization(same)	✓	✓
Fingerprint	✓	✓

Table 2.3 Comparison between MD5 and SHA hash algorithm on general properties basic (Jose. R, Thomas. G, 2015)

Name of the algorithm	Size of the output	Rounds	Collision Status
MD5	128	60	YES
SHA	160	80	YES

This comparative helped us understand that the SHA algorithm plays a very important role in comparison to MD5 because SHA algorithms' performance rate is comparatively better than other cryptographic hash algorithm functions. The question is: Will the results be similar when the improved Protocol is applied?

2.11 Related works

- (Black. J, Cochran. M & Highland. T, 2006) this study supports one of the techniques that are generated for enhancing the MD5 as the (MD5 Toolkit), this tool is generated to deal with the weaknesses that found by Wang and others in the MD5 protocol, they referenced that the MD5 still used in various applications including SSL/TLS, IPSec, and many other protocols, also they pointed out that the several destinations, are still working by using the MD5 protocol such as in:

- Implementation of timestamping mechanisms.
- Commitment schemes.
- Integrity-checking applications for online software.
- Distributed file system.
- Random-number generation.

It is even used by the Nevada State Gaming Authority to ensure slot-machine ROMs would have not been modified with any modification.

- (Kasgar. A, Agrawal. I & Sahu. S, 2012) the study works on increasing hash code length up to 256, the study assumed to make the hash protocol stronger against collision attested, and works in a combination of some functions to reinforce the hash functions.

This study dealt with a wide range of differences between MD5 and SHA family, the main different conclusion at this point is that the MD5 and the newer one SHA-256 is equal in the complexity of both algorithms, and the value is O (N), but the running time of the MD5 is faster than SHA-256 as shown in figure 2.4.

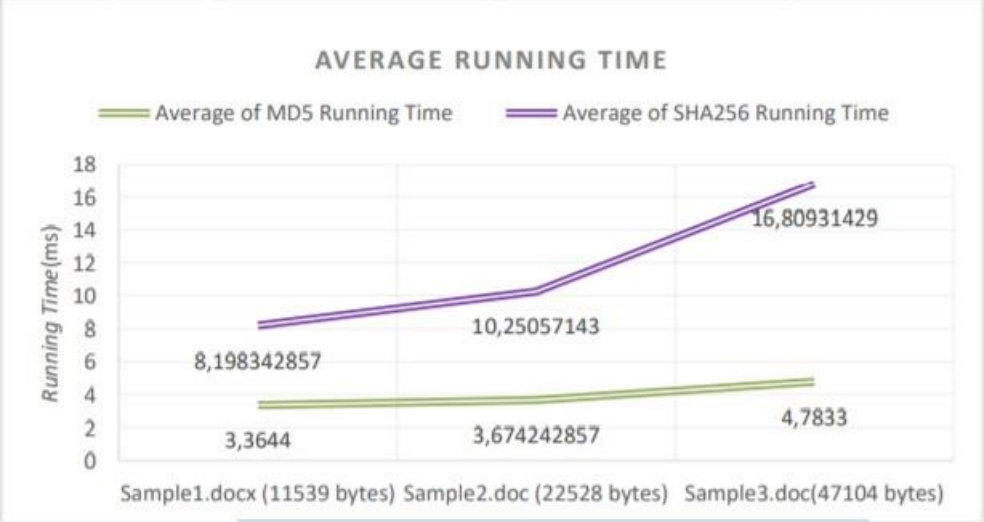


Figure 2.4 Graph about average running time MD5 and SHA256 (Rachmawati. D, Tarigan. J & Ginting. A, 2018).

- (Libed. J, Sison. A & Dr.Medina. R, 2018) the study explains a new mechanism to enhance the MD5 and to protect the file integrity by a new method for the padding process of the original message and to apply a technical solution, as in the additional operations on the internal processes that are implemented.

Some of the solutions that the study suggests:

• The result of the computing simulation indicates the extension of the message block from 512 to 1024-bit block.

• Expansion of the length of the resulting value, per round from 32 to 64 bits together with added operations, increases the security of the modified message digest hash function.

It was also noted that many studies have aimed at many solutions. This study, while changing the size of the data used to suit the required function away from the existing weakness, has carried out several tests to verify the reliability of this process and the results were somewhat satisfactory, also the test was for the evaluation of the produced hash value has been conducted using the avalanche effect test that resulted to a value of " 56.91 and randomness test to assess the randomization value into which a remarkable output of 56.45 and 55.93 respectively have been obtained into which it has considerably been attested.

- (Maliberan. E, Sison. A & Medina. R, 2018) the study works on enhancing the MD5 algorithm by developed expanding the hash value up 1280-bits from the original size of 128-bit using XOR and AND operators, and by using an available source to check the security of the new algorithm they have as:

• Powerful brute force.

• Dictionary.

• Cracking tools.

• Rainbow table.

• Cracking station.

• Hash cracker.

• Cain and Able.

• Rainbow crack.

They found that the hash value of the modified algorithm was not cracked or hacked during the experiment and testing, using the above sources and comparing with the results from the original MD5.

This study relied on existing techniques to prove its new technique, but did not take into account that the hackers can choose mechanisms that are not complex or may be computational such as Wang's attack technique, and exploited to prove the existence of a collision or to have two identical Hash values with the difference of original blocks.

- (Karani. K, Aithal. S, 2018) one of the studies that approved the efficacy of using the fingerprint with MD5 protocol, they take a fingerprint image hash code based on the MD5 algorithm and Freeman chain code calculated on the binary image.

The study supposed that the hash code alone is not sufficient for verification or authentication purpose. They used multifactor security MATLAB2015a. The study shows how fingerprint hash code uniquely identifies a user or acts as index-key.

The main use of the fingerprint hashing in the new technique is to perform identification based on fingerprint simultaneously hiding or keeping the fingerprint information secretly.

The study shows that using the fingerprint alone is not enough, and the user needs to use a good password or a special word because the fingerprint can easily be mimicked

by a fraud or an intruder, as the fingerprint does not get matched when the finger has some changes as surgeries, because the fingerprint is working as identity or index-key and not as a full security feature.

This study has proved the effectiveness of using fingerprint in the systems of authentication and credibility by the user, and noted that using the values within the fingerprint is effective and highly protected; this study relied on specific uses for the purpose of documentation.

Up to now this method stays not suitable for solely security purpose unless the user takes some security measures to protect static fingerprint image.

2.12 Summary

The MD-Message-Digest protocol is easy to implement, while the original MD5 collision attacks, in some cases, make it weak. Many companies, that are still working with MD5 and Cisco routers, are working on "enabling the secret" (Cisco, 2007), and applications such as password protection, sending and receiving files and others in the retail application. It is supposed that the difficulty of sending two messages with the same message number is in the order of 2 ^ 64 transactions, and that the difficulty of accessing any message with a specific message digest is within 2 ^ 128 operations. Through studies of the MD5 collision, the algorithm is viewed as prone to attacks and therefore cannot be considered safe and secure enough.

That is the cause this proposed work is presented, taking into account the need for a higher security technique to support the study, consider comparisons with existing protocols currently adopted.

Chapter Three

Methodology and the Proposed Work

CHAPTER THREE

Methodology and the Proposed Work

3.1 Introduction

This chapter introduces the proposed modification scheme of the MD5 algorithm since the main purpose of the amendment is to increase the efficiency of the algorithm and make it safer against collision attacks. The original MD5 is explained first, and then a simple experiment is performed showing the results of using the original protocol in its fragmental operations, as well as identifying the main weaknesses in MD5, and finally followed by a description of the proposed protocol, where each step in the proposed modification is explained, and presented.

3.2 The methodology

After studying the original MD5 and identifying its weaknesses, the most important of which is the collision attack, the MD5 does not seem to be considered safe enough. The main source of weaknesses of the protocol is using fixed values in step 4 when the hash value is being built. The new technique uses the user's fingerprint instead of the fixed values in step 4, which aims to make the new protocol safer.

The reason of making the hash value more secure is that the main dependence will be on generating the hash value of the general adoption of the protocol building values on the user's fingerprint, which makes the values difficult to be predicted, the fingerprint eventually is read by using the fingerprint scanner.

This chapter contains two parts; The first part is to investigate the original MD5 algorithm, and to identify the weaknesses in detail to produce a hash value with a simple protocol experiment to be compared with the new protocol results later, while the second part offers the proposed version of modified MD5 protocol to improve its reliability and integrity for hash value and validation. The values of the test are assumed by repeating the same experiment in the first part using the modified protocol (the results are presented in chapter four).

3.3 Investigation of the original MD5 algorithm

As noted in chapter two, the weakness in the protocol is based on the fourth step, specifically the fixed values that make the MD5 weak since the values are known, which facilitates the process of predicting the value of message digest, where many studies refer, as previously described, to the existence of a collision attack which is about making two files have an inordinate length holding the same hash value, as this vulnerability is unacceptable and considered as a vulnerability and a flaw in the safety of the used protocol.

Here is a simplified explanation of the fourth step in the original MD5, and places of the existence of fixed values, as well as the mechanism of definition when working to extract the value of the message digest (the original equations):

```
#define S11 7
#define S12 12
#define S13 17
#define S14 22
```

```
#define S21 5

#define S22 9

#define S23 14

#define S24 20

#define S31 4

#define S32 11

#define S33 16

#define S34 23

#define S41 6

#define S42 10

#define S43 15

#define S44 21
```

```
 /* Round 1 */

 FF (a, b, c, d, x [0], S11, 0xd76aa478); /* 1 */

 FF (d, a, b, c, x [1], S12, 0xe8c7b756); /* 2 */

 FF (c, d, a, b, x [2], S13, 0x242070db); /* 3 */

 FF (b, c, d, a, x [3], S14, 0xc1bdceee); /* 4 */

 FF (a, b, c, d, x [4], S11, 0xf57c0faf); /* 5 */

 FF (d, a, b, c, x [5], S12, 0x4787c62a); /* 6 */

 FF (c, d, a, b, x [6], S13, 0xa8304613); /* 7 */

 FF (b, c, d, a, x [7], S14, 0xfd469501); /* 8 */

 FF (a, b, c, d, x [8], S11, 0x698098d8); /* 9 */

 FF (d, a, b, c, x [9], S12, 0x8b44f7af); /* 10 */
```

```
FF (c, d, a, b, x [10], S13, 0xffff5bb1); /* 11 */

FF (b, c, d, a, x [11], S14, 0x895cd7be); /* 12 */

FF (a, b, c, d, x [12], S11, 0x6b901122); /* 13 */

FF (d, a, b, c, x [13], S12, 0xfd987193); /* 14 */

FF (c, d, a, b, x [14], S13, 0xa679438e); /* 15 */

FF (b, c, d, a, x [15], S14, 0x49b40821); /* 16 */


/* Round 2 */

GG (a, b, c, d, x [1], S21, 0xf61e2562); /* 17 */

GG (d, a, b, c, x [6], S22, 0xc040b340); /* 18 */

GG (c, d, a, b, x [11], S23, 0x265e5a51); /* 19 */

GG (b, c, d, a, x [0], S24, 0xe9b6c7aa); /* 20 */

GG (a, b, c, d, x [5], S21, 0xd62f105d); /* 21 */

GG (d, a, b, c, x [10], S22, 0x2441453); /* 22 */

GG (c, d, a, b, x [15], S23, 0xd8a1e681); /* 23 */

GG (b, c, d, a, x [4], S24, 0xe7d3fbc8); /* 24 */

GG (a, b, c, d, x [9], S21, 0x21e1cde6); /* 25 */

GG (d, a, b, c, x [14], S22, 0xc33707d6); /* 26 */

GG (c, d, a, b, x [3], S23, 0xf4d50d87); /* 27 */

GG (b, c, d, a, x [8], S24, 0x455a14ed); /* 28 */

GG (a, b, c, d, x [13], S21, 0xa9e3e905); /* 29 */

GG (d, a, b, c, x [2], S22, 0xfcefa3f8); /* 30 */

GG (c, d, a, b, x [7], S23, 0x676f02d9); /* 31 */

GG (b, c, d, a, x [12], S24, 0x8d2a4c8a); /* 32 */
```

```
/* Round 3 */
HH (a, b, c, d, x [5], S31, 0xfffa3942); /* 33 */
HH (d, a, b, c, x [8], S32, 0x8771f681); /* 34 */
HH (c, d, a, b, x [11], S33, 0x6d9d6122); /* 35 */
HH (b, c, d, a, x [14], S34, 0xfde5380c); /* 36 */
HH (a, b, c, d, x [1], S31, 0xa4beea44); /* 37 */
HH (d, a, b, c, x [4], S32, 0x4bdecfa9); /* 38 */
HH (c, d, a, b, x [7], S33, 0xf6bb4b60); /* 39 */
HH (b, c, d, a, x [10], S34, 0xbebfbc70); /* 40 */
HH (a, b, c, d, x [13], S31, 0x289b7ec6); /* 41 */
HH (d, a, b, c, x [0], S32, 0xeaa127fa); /* 42 */
HH (c, d, a, b, x [3], S33, 0xd4ef3085); /* 43 */
HH (b, c, d, a, x [6], S34, 0x4881d05); /* 44 */
HH (a, b, c, d, x [9], S31, 0xd9d4d039); /* 45 */
HH (d, a, b, c, x [12], S32, 0xe6db99e5); /* 46 */
HH (c, d, a, b, x [15], S33, 0x1fa27cf8); /* 47 */
HH (b, c, d, a, x [2], S34, 0xc4ac5665); /* 48 */

/* Round 4 */
II (a, b, c, d, x [0], S41, 0xf4292244); /* 49 */
II (d, a, b, c, x [7], S42, 0x432aff97); /* 50 */
II (c, d, a, b, x [14], S43, 0xab9423a7); /* 51 */
II (b, c, d, a, x [5], S44, 0xfc93a039); /* 52 */
II (a, b, c, d, x [12], S41, 0x655b59c3); /* 53 */
II (d, a, b, c, x [3], S42, 0x8f0ccc92); /* 54 */
```

```
II (c, d, a, b, x [10], S43, 0xffeff47d); /* 55 */
II (b, c, d, a, x [1], S44, 0x85845dd1); /* 56 */
II (a, b, c, d, x [8], S41, 0x6fa87e4f); /* 57 */
II (d, a, b, c, x [15], S42, 0xfe2ce6e0); /* 58 */
II (c, d, a, b, x [6], S43, 0xa3014314); /* 59 */
II (b, c, d, a, x [13], S44, 0x4e0811a1); /* 60 */
II (a, b, c, d, x [4], S41, 0xf7537e82); /* 61 */
II (d, a, b, c, x [11], S42, 0xbd3af235); /* 62 */
II (c, d, a, b, x [2], S43, 0x2ad7d2bb); /* 63 */
II (b, c, d, a, x [9], S44, 0xeb86d391); /* 64 */

State [0] += a;
State [1] += b;
State [2] += c;
State [3] += d;
```

- Where A, B, C, and D are fixed values that are already defined by the MD5 to output the Digest.

- X [16] is the value of Shared Key accessed by an MD5 Protocol user.

- S11 to S44 are also fixed values that are defined by the MD5 algorithm.

- The values 0xf4292244 and 0xeb86d391 are static values passed through the MD5 Protocol.

3.3.1 Simple test on MD5

This simple test has been used to encode the current protocol from an open source on the Internet, based on MD5 for different files:

Table 3.1 result table

File name	File size	Hash value
Document 1	11 bytes	3f80c5aaaae973738a52fc1e0507bbd1
Document 2	11 bytes	b6624d8dc379ee5810d32ad1e7ddc833
Pdf 1	265 KB	f6ab5b67c316fc90a12056b0257fb99f
Pdf 2	345 KB	0d0ac21344537b9e3bfff1caa86c71bc
Word 1	12.2 KB	5294e064fc021ecd5d618b696558b30e
Word 2	12.2 KB	72a88edf67a16edd7af7e03a653971be
Document 3	111 bytes	4146e3a8cfac929eb7fdbf1bf845d5cb

These states demonstrate a simple implementation of the original MD5 on a simple set of files, as at this stage the probability of a collision attack is weak, but with the experience, it becomes available, as mentioned in previous studies listed in the previous chapter.

3.4 The proposed MDM protocol

The proposed protocol represents an extension of the original MD5 protocol and henceforth will be referred to as a new name the MDM (Message Digest Modification) protocol. The MDM message-digest-modification protocol takes as input a message of inordinate length and produces as output a 128-bit "message digest" of the input. It is assumed that it is computationally infeasible to produce two

messages having the same message digest, or to produce any message has a given predefined target. The MDM protocol is intended for digital signature applications, where a large file must be "compressed" in a secure case before being encrypted with a private (secret) key using a public-key cryptosystem such as RSA. Otherwise, the protocol can be used in password protection operations by using the same fingerprint and merging it with the username to produce the Hash value.
The structure and the method of operation of the MDM protocol will be described in the following sections.

3.5 The proposed changes

The aim of the new protocol is to strengthening the mechanism of the old protocol work to become effective and reliable, so that the work is done in the same way as discussed in chapter two, where step 1 and step 3 have the same structure. But the mechanism of symmetric handling does not alter any modifications, since the adjustment lies within step 4 as the construction and use are similar to the MD5 protocol. The difference lies within the process of constructing hash within step 4 where the mechanisms of calculation in the fourth step of compatibility side by side with the mechanism of using the fingerprint. so that, the fixed values in the previous section has been changed to consider the values of a variable extracted from the fingerprint, thus, the fingerprint was used to read 1024 Bit of the final points of the fingerprint, and the last 512 bits were used in the process of Hash. Also the last-second section was used 512 bits to be considered among the equations responsible for extracting a special value for comparison when receiving the extracted value.

Here is a simplified explanation of the fourth step in the new MDM protocol and places of the existence of fixed values and mechanism of definition when working to extract the value of the message digest (the transformation equations that have been modified):

#define S11 7

#define S12 12

#define S13 17

#define S14 22

#define S21 5

#define S22 9

#define S23 14

#define S24 20

#define S31 4

#define S32 11

#define S33 16

#define S34 23

#define S41 6

#define S42 10

#define S43 15

#define S44 21

```
/* Round 1 */
FF (a, b, c, d, Sharedkey, S11, Fingerprint); /* 1 */
FF (d, a, b, c, Sharedkey, S12, Fingerprint); /* 2 */
FF (c, d, a, b, Sharedkey, S13, Fingerprint); /* 3 */
FF (b, c, d, a, Sharedkey, S14, Fingerprint); /* 4 */
FF (a, b, c, d, Sharedkey, S11, Fingerprint); /* 5 */
FF (d, a, b, c, Sharedkey, S12, Fingerprint); /* 6 */
FF (c, d, a, b, Sharedkey, S13, Fingerprint); /* 7 */
FF (b, c, d, a, Sharedkey, S14, Fingerprint); /* 8 */
FF (a, b, c, d, Sharedkey, S11, Fingerprint); /* 9 */
FF (d, a, b, c, Sharedkey, S12, Fingerprint); /* 10 */
FF (c, d, a, b, Sharedkey, S13, Fingerprint); /* 11 */
FF (b, c, d, a, Sharedkey, S14, Fingerprint); /* 12 */
FF (a, b, c, d, Sharedkey, S11, Fingerprint); /* 13 */
FF (d, a, b, c, Sharedkey, S12, Fingerprint); /* 14 */
FF (c, d, a, b, Sharedkey, S13, Fingerprint); /* 15 */
FF (b, c, d, a, Sharedkey, S14, Fingerprint); /* 16 */

/* Round 2 */
GG (a, b, c, d, Sharedkey, S21, Fingerprint); /* 17 */
GG (d, a, b, c, Sharedkey, S22, Fingerprint); /* 18 */
GG (c, d, a, b, Sharedkey, S23, Fingerprint); /* 19 */
GG (b, c, d, a, Sharedkey, S24, Fingerprint); /* 20 */
GG (a, b, c, d, Sharedkey, S21, Fingerprint); /* 21 */
GG (d, a, b, c, Sharedkey, S22, Fingerprint); /* 22 */
```

```
GG (c, d, a, b, Sharedkey, S23, Fingerprint); /* 23 */

GG (b, c, d, a, Sharedkey, S24, Fingerprint); /* 24 */

GG (a, b, c, d, Sharedkey, S21, Fingerprint); /* 25 */

GG (d, a, b, c, Sharedkey, S23, Fingerprint); /* 26 */

GG (c, d, a, b, Sharedkey, S24, Fingerprint); /* 27 */

GG (b, c, d, a, Sharedkey, S24, Fingerprint); /* 28 */

GG (a, b, c, d, Sharedkey, S21, Fingerprint); /* 29 */

GG (d, a, b, c, Sharedkey, S22, Fingerprint); /* 30 */

GG (c, d, a, b, Sharedkey, S23, Fingerprint); /* 31 */

GG (b, c, d, a, Sharedkey, S24, Fingerprint); /* 32 */


/* Round 3 */

HH (a, b, c, d, Sharedkey, S31, Fingerprint); /* 33 */

HH (d, a, b, c, Sharedkey, S32, Fingerprint); /* 34 */

HH (c, d, a, b, Sharedkey, S33, Fingerprint); /* 35 */

HH (b, c, d, a, Sharedkey, S34, Fingerprint); /* 36 */

HH (a, b, c, d, Sharedkey, S31, Fingerprint); /* 37 */

HH (d, a, b, c, Sharedkey, S32, Fingerprint); /* 38 */

HH (c, d, a, b, Sharedkey, S33, Fingerprint); /* 39 */

HH (b, c, d, a, Sharedkey, S34, Fingerprint); /* 40 */

HH (a, b, c, d, Sharedkey, S31, Fingerprint); /* 41 */

HH (d, a, b, c, Sharedkey, S32, Fingerprint); /* 42 */

HH (c, d, a, b, Sharedkey, S33, Fingerprint); /* 43 */

HH (b, c, d, a, Sharedkey, S34, Fingerprint); /* 44 */

HH (a, b, c, d, Sharedkey, S31, Fingerprint); /* 45 */
```

HH (d, a, b, c, Sharedkey, S32, Fingerprint); /* 46 */

HH (c, d, a, b, Sharedkey, S33, Fingerprint); /* 47 */

HH (b, c, d, a, Sharedkey, S34, Fingerprint); /* 48 */

/* Round 4 */

II (a, b, c, d, Sharedkey, S41, Fingerprint); /* 49 */

II (d, a, b, c, Sharedkey, S42, Fingerprint); /* 50 */

II (c, d, a, b, Sharedkey, S43, Fingerprint); /* 51 */

II (b, c, d, a, Sharedkey, S44, Fingerprint); /* 52 */

II (a, b, c, d, Sharedkey, S41, Fingerprint); /* 53 */

II (d, a, b, c, Sharedkey, S42, Fingerprint); /* 54 */

II (c, d, a, b, Sharedkey, S43, Fingerprint); /* 55 */

II (b, c, d, a, Sharedkey, S44, Fingerprint); /* 56 */

II (a, b, c, d, Sharedkey, S41, Fingerprint); /* 57 */

II (d, a, b, c, Sharedkey, S42, Fingerprint); /* 58 */

II (c, d, a, b, Sharedkey, S43, Fingerprint); /* 59 */

II (b, c, d, a, Sharedkey, S44, Fingerprint); /* 60 */

II (a, b, c, d, Sharedkey, S41, Fingerprint); /* 61 */

II (d, a, b, c, Sharedkey, S42, Fingerprint); /* 62 */

II (c, d, a, b, Sharedkey, S43, Fingerprint); /* 63 */

II (b, c, d, a, Sharedkey, S44, Fingerprint); /* 64 */

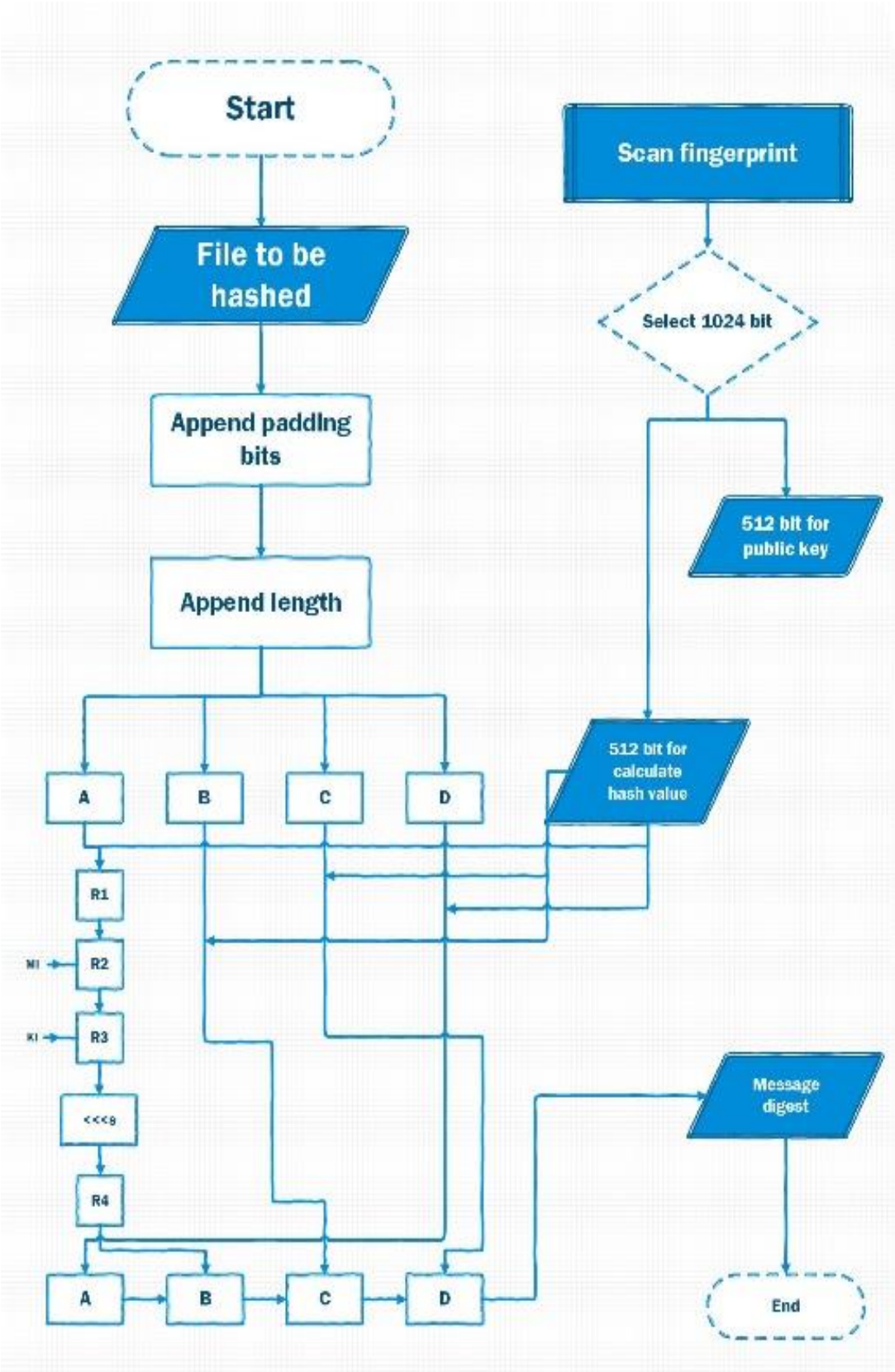


Figure 3.1 Create message digest using new MDM

3.5.1 The method of operation

The work in this protocol is similar to the original MD5 protocol as mentioned in the previous chapter, where the steps from 1 to 3 are exactly the same as the original protocol as they have a full explanation in chapter two, where Step 1 is Append Padding Bits, Step 2 is Append Length, and Step 3 is Initialize MD Buffer, and Step 4. Process Message in 16-Word Blocks, where step 3, when the protocol gets started to initialize the MD buffers the values, it will be changed to be as the new protocol MDM work , the values will become from the fingerprint to be initialized to being used in step 4 in the transformation step. The main work and the implementation of the goal of this study will be in this step. This step is a compression algorithm that consists of four rounds of processing. The fundamental of the four rounds have a similar structure, but each round has a different primitive logical function and specification, refers to as F, G, I
+.and H. Each round takes the 32-bit buffer value ABCD and the current block 512-bit and each round consists of a 64-element from the lookup table. The changes addressed here are to change the inputs to the rounds and function to be merged with the fingerprint value. After all these steps they output 128-bit message digest as shown in figure 3.1.

In figure 3.1, the protocol starts by reading the fingerprint of the fingerprint scanner. The fingerprint is recognized as a picture and converted to a byte matrix. It is split into an array of bits and the protocol takes the last 1024 bits of the array to be prepared in step 3, and then used in step 4 to build the message digest value.

These 1024 bits are used in the protocol structure whereas the user file or the text used in the Hash process is processed by the first steps as mentioned earlier:

Step .1 Append Padding Bits

The message is "padded" (lengthened) so that its length (in bits) is corresponding to 448, modulo 512. That is, the message is lengthened so it is just 64 bits shy of being a multiple of 512 bits long. Even if the length of the message is already corresponding to 448, modulo 512, padding is always executed. Where it is performed as the following: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. Totally, at least one bit and at most 512 bits are appended.

Step .2 Append Length

A 64-bit representation, of b (the length of the message before the padding bits were added), is appended to the result of the previous step. as it is unlikely event that b is greater than 2^64, so only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.) At this point, the resulting message (after padding with bits and with b) has a length that is a specific multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M [0 ... N-1] point out the words of the resulting message, where N is a multiple of 16.

Step 3. Initialize MD Buffer

A four-word buffer (A, B, C, and D) is used to compute the message digest. Here, each of A, B, C, and D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first:

Word [A] = 0x67452301;

Word [B] = 0xefcdab89;

Word [C] = 0x98badcfe;

Word [D] = 0x10325476;

Step .4 Process Message in 16-Word Blocks

We first define four auxiliary functions that each takes as input three 32-bit words and produces as output one 32-bit word.

F(X, Y, Z) = XY v not(X) Z

G(X, Y, Z) = XZ v Y not (Z)

H(X, Y, Z) = X xor Y xor Z

I(X, Y, Z) = Y xor (X v not (Z))

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X) Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, each bit of F(X, Y, and Z) will be independent and unbiased. The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner, that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X, Y, Z), H(X, Y, Z), and I(X, Y, Z) will be independent and unbiased. Note that the function H is the bit-wise "xor" or "parity" function of its inputs.

The four words (A, B, C and D) each word will be entered in the four rounds to calculate the message digest in this protocol, the values will be computationally computed with the value of 512 bits taken from the fingerprint with the four words to be introduced, and to be aggregated and then to give the value of the message digest.

In the right side of the figure3.1, there is a 512 bit still unused will be used in the next step.

3.5.2 Create checksum

In the new protocol, more values to be dealt with than the old MD5 protocol. We need to extrapolate values in the comparison process to the message digest to calculate the checksum. This value is extracted by converting the 512 bits before the last one from hexadecimal to the Decimal and also inserting values in the assembly frame to produce a valid number that can be used, as well as the same steps are executed on the value of message digest and are converted in the same way, so as to deal with the resulting numbers of this process and the application of the mathematical equation Least Common Multiple (LCM) to find the value of the division, and to be sent attached to the value of message digest as shown in figure 3.2.

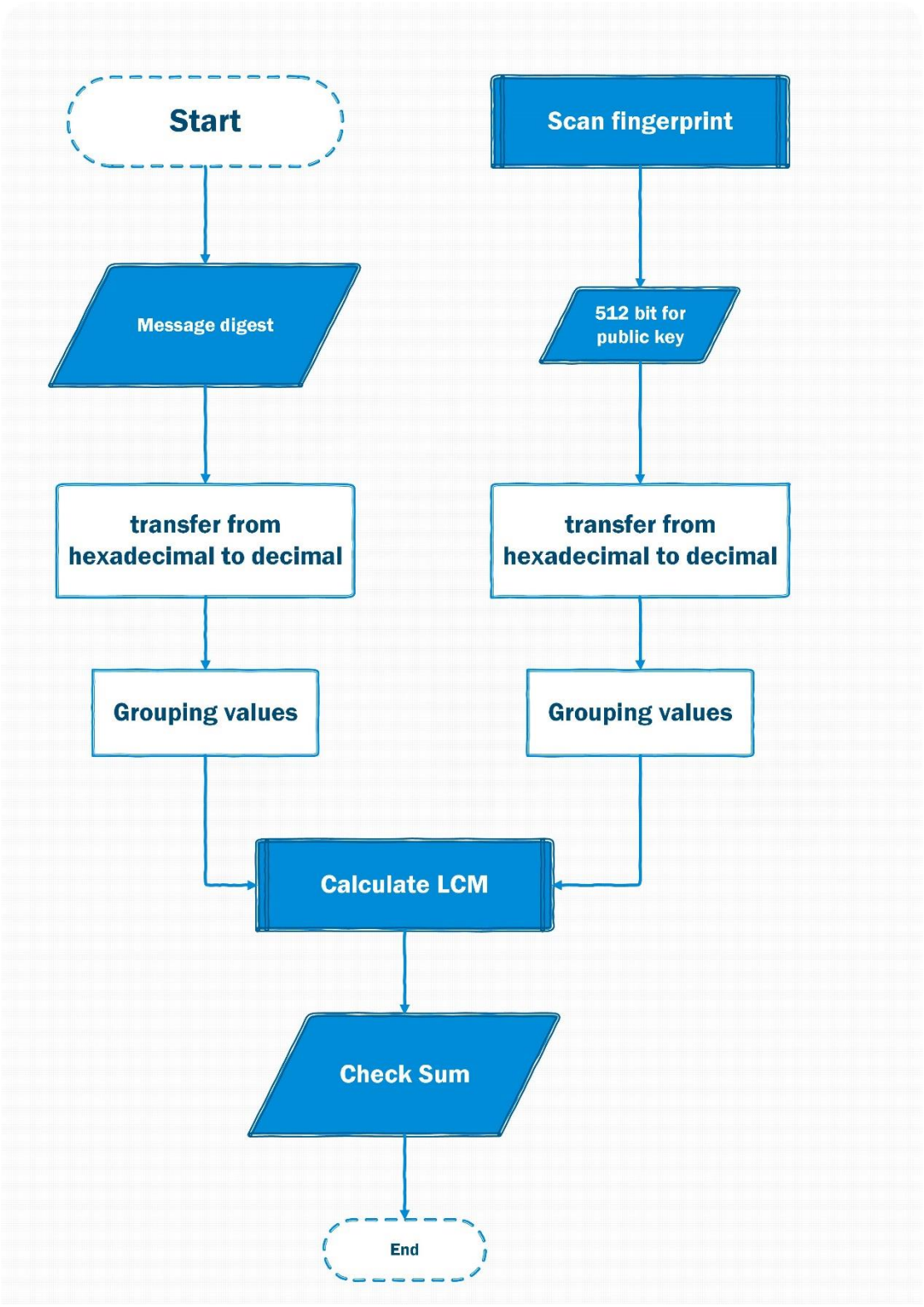


Figure 3.2 Create checksum value using new MDM

All of these changes are made in the sender side, and here is an explanation of the side of the recipient.

3.5.3 Decode Step

In this step, the integrity of the files is checked and it is manipulated by delivering three values: message digest, public key, and checksum.

The receiver converts the value of the message digest and also the public key from the hexadecimal to the decimal, grouping the values, inserts them into the equation LCM to extract their values and compare them in the scaling. If the values match, the file is intact and it has not been manipulated, but unless the value matches, the file is not correct and has been manipulated.

The process of sending the public key and the checksum for verification purposes such as digital signature and in the case of password protection to give permission to enter in the system as show in figure 3.3.

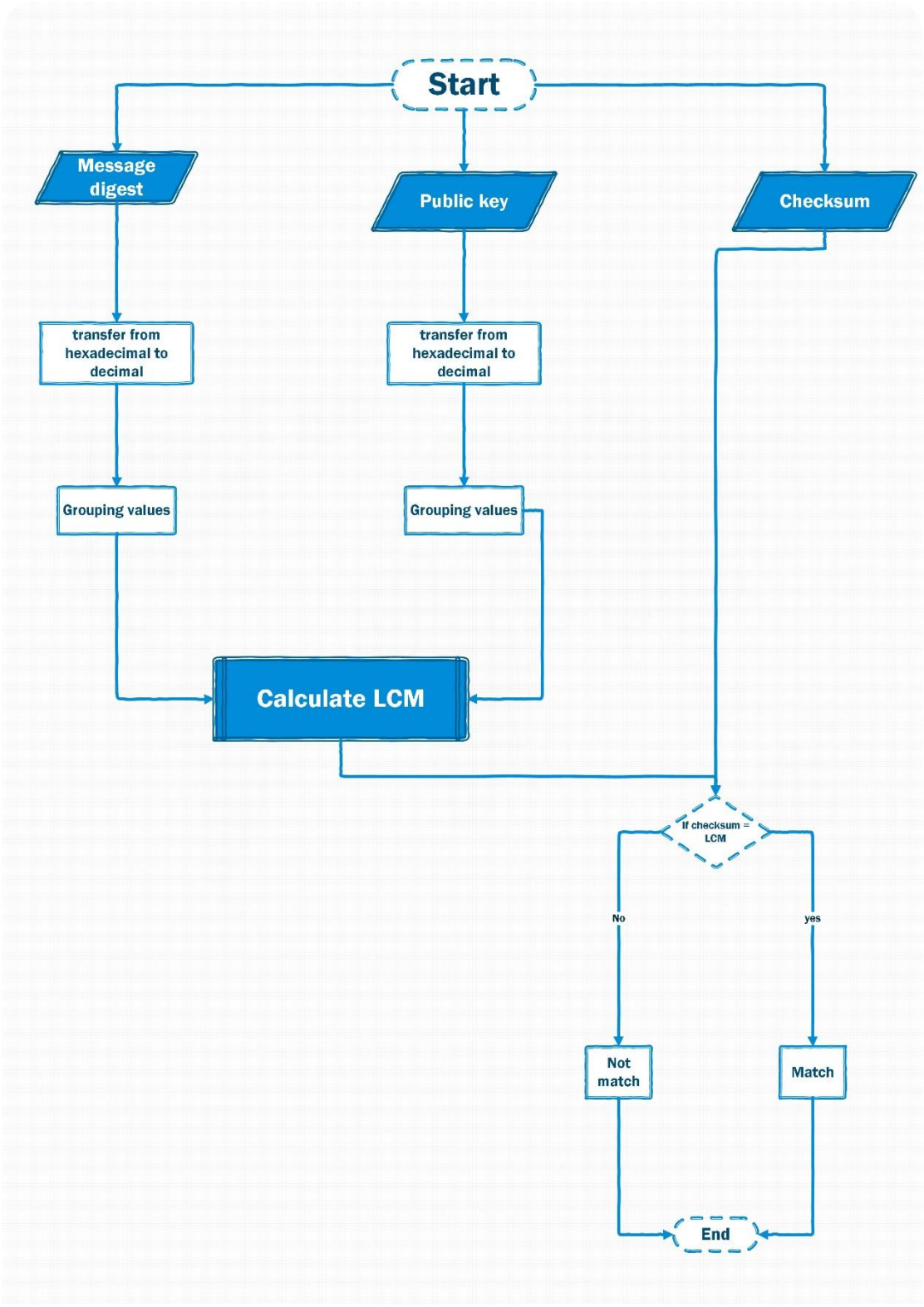


Figure 3.3 Decode step using the new MDM

3.6 Proposed Measurements

Two methods of measurement will be used:

1. Security tests will be conducted to provide a verification of the proposed protocol.

2. A comprehensive comparison of the suggested scheme will be conducted with SHA and traditional MD5.

3.7 The method of evaluation

To prove the strength of the new MDM protocol, the new algorithm must deal with the weaknesses of the original MD5, the collision attacks.

a) Test the new protocol in a local system contains server/client to ensure that sending and receiving of the data by using the new protocol is working, and that the way of hashing the password protection before being loaded in the database of the server, is applicable.

b) Efficacy of the MDM

The purpose of the efficacy test is to demonstrate the practical side of using the new MDM in a work environment like (Microsoft product daily use and others) to get feedback from the users about the use of the product.

Chapter Four

Implementation and Results

Chapter four

Implementation and Results

4.1 Introduction

The implementation of the proposed MDM algorithm is introduced in this chapter, in addition to clarifying a researcher-specific program that has been applied in several companies to check the applicability of algorithm in practical application using Microsoft Visual Studio build with C # language, to build a special chat system and to implement the MDM algorithm for protecting passwords as well as creating the digital signature processes, Then , the comparison between SHA256, MD5, and MDM in terms of the required time difference to execute the algorithm on files of different sizes, Finally, the results of the use of MDM on the same files that were used in a simple experiment on MD5 to know the value of Hash in chapter 3, which expresses the results of the breaches on the MDM algorithm

4.2 MDM implementation

The new algorithm was implemented by constructing three special systems for the researcher; the first system is based on a special program for recording users according to their username and fingerprints, the possibility of having conversations and sending files which are all directed to the new system algorithm for password protection operations. The user and the installation of

values for his fingerprint are handled in storage places and application algorithm to the process of Hash and retain its value until the request for retrieval at the time of login.

The second system is based on the possibility of an application for digital signature so that a file is selected and attached to the user's request to put the fingerprint to use the values in the Hash process since the electronic signature is linked to the RSA algorithm to complete the hash process correctly.

The third system is based on the possibility of extracting the value of Hash files, and clarifying the process of comparison through it with most of the Hash algorithms adopted and the most popular; MD5 and SHA family.

The following is an illustration of the software screens that were used to implement the practice in a real work environment for some companies that deal with office files and need more security for data.

And also an explanation of the screens used for each system and the correct and incorrect input results:

4.2.1 System one (chat system consists of server/client)

This system consists of a server and a client where a direct connection is opened between them and recording the data that transfers to the server when the registration process to the client and the retention of data to be retrieved, as this system proved efficient to maintain the user's data, and inability to have them penetrated by breakthroughs, which will be mentioned later in this chapter as shown in figure 4.1, figure 4.2, figure 4.3, figure 4.5 and figure 4.6.

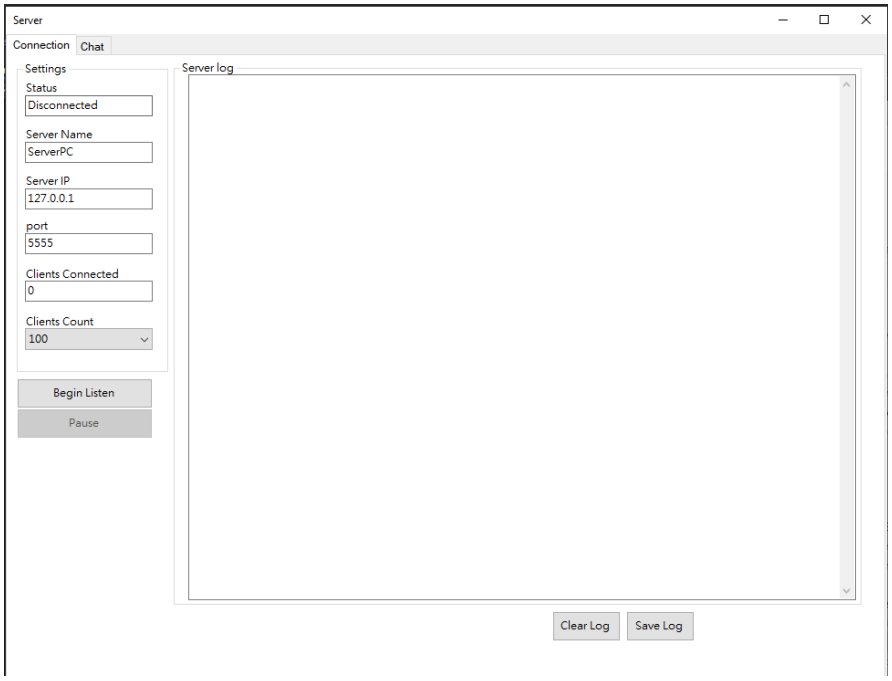


Figure 4.1 the server display: is waiting to open a direct connection with the client.

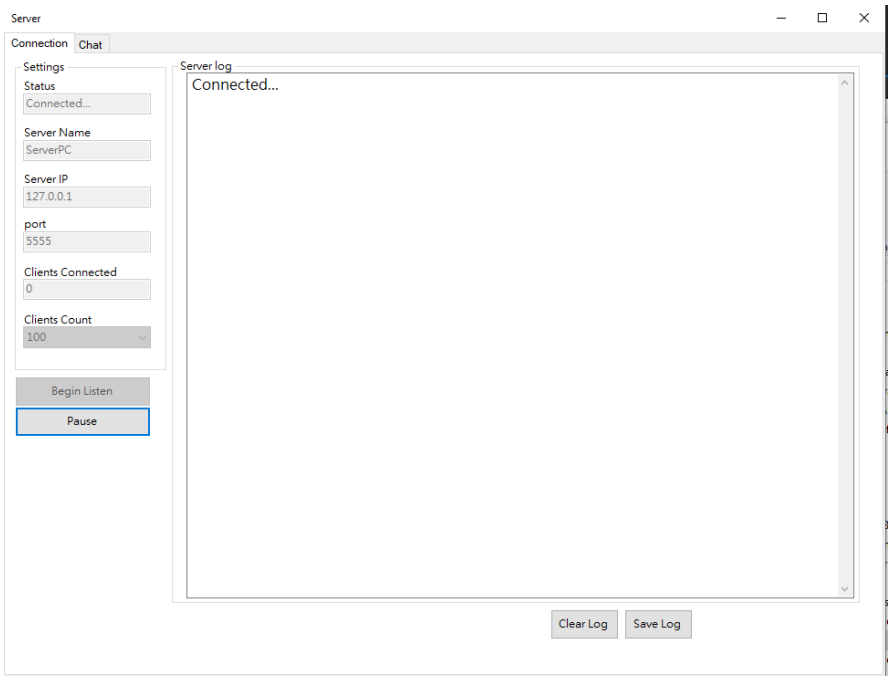


Figure 4.2 open the connection

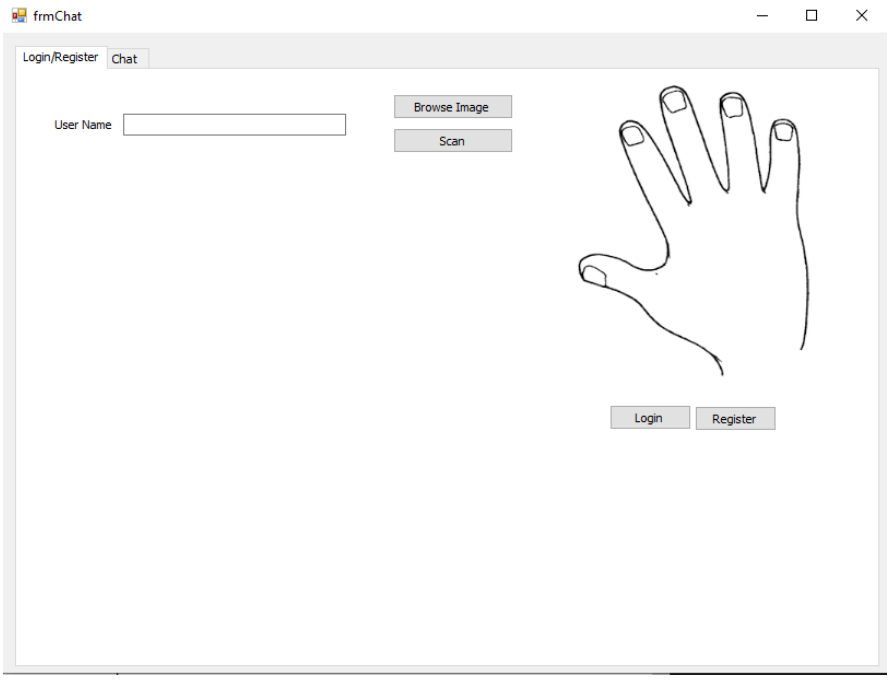


Figure 4.3 the client screen

where the system is waiting to enter the user name and the position of the fingerprint to be compared to its value in the database, and compare the results to give the permission to enter or reject, also some common errors not to accept the fingerprint by the first time, so the system allows the possibility of restoring the fingerprint to allowed the user to enter as shown in figure 4.3.

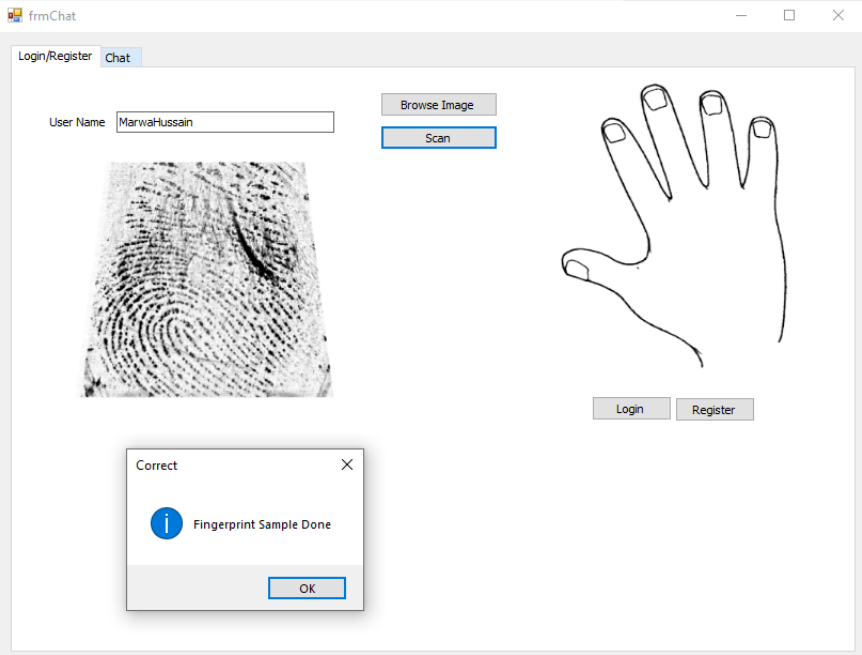


Figure 4.4 Confirm that the fingerprint entered correctly gives the permissions to enter the system.

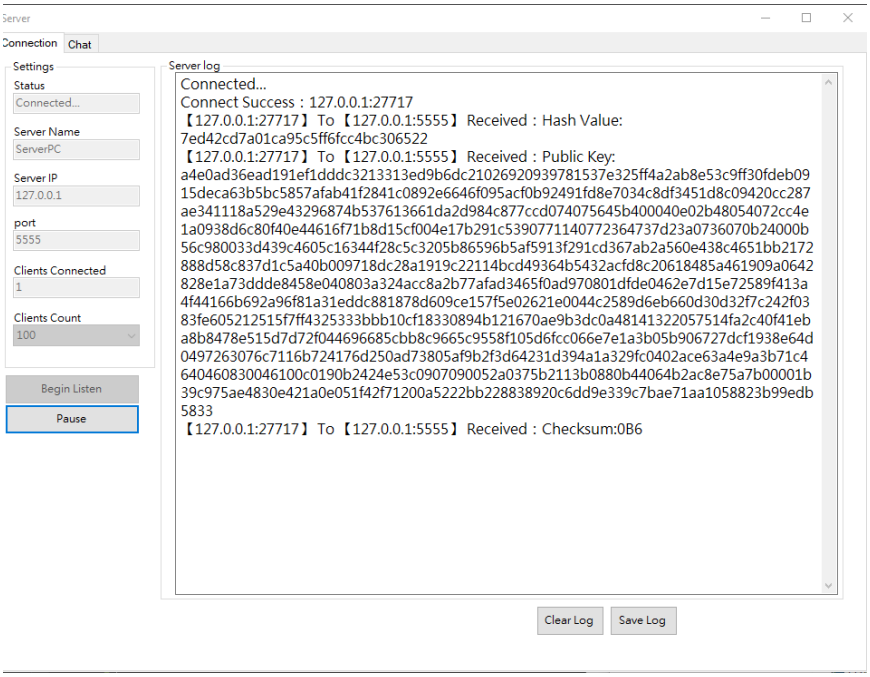


Figure 4.5 the server log

That records the calculated data for the value of the Hash as explained in the first chapters and which is (Hash value or the message digest, public key the 512 bit that we use, and the checksum that has been calculated with the LCM equation) as shown in figure 4.5.

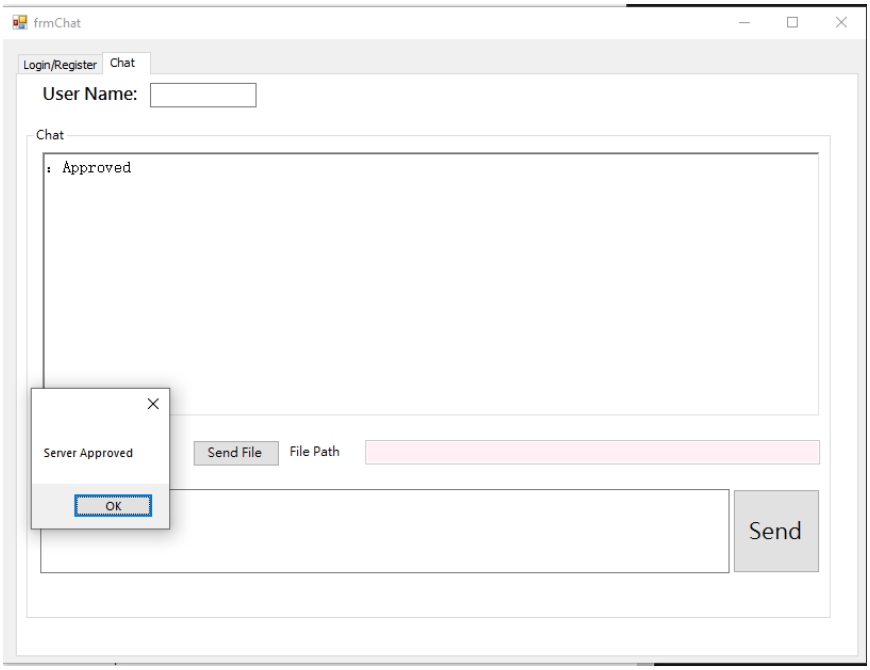


Figure 4.6 after authentication, the use of the services available within the system and after the user's verification is enabled.

4.2.2 System two (digital signature generator)

This system is used to insert files with different measurements and is authenticated with a digital signature using the proposed protocol in dealing with the RSA algorithm. As Hash processes use this algorithm in the process of creating the digital signature, this system includes a screen to insert the file and request the use of the fingerprint to find the special values in it. Checking the signature and not having any modification to the file are explained in the screens as shown in figure 4.7, figure 4.8, figure 4.9 and figure 4.10.

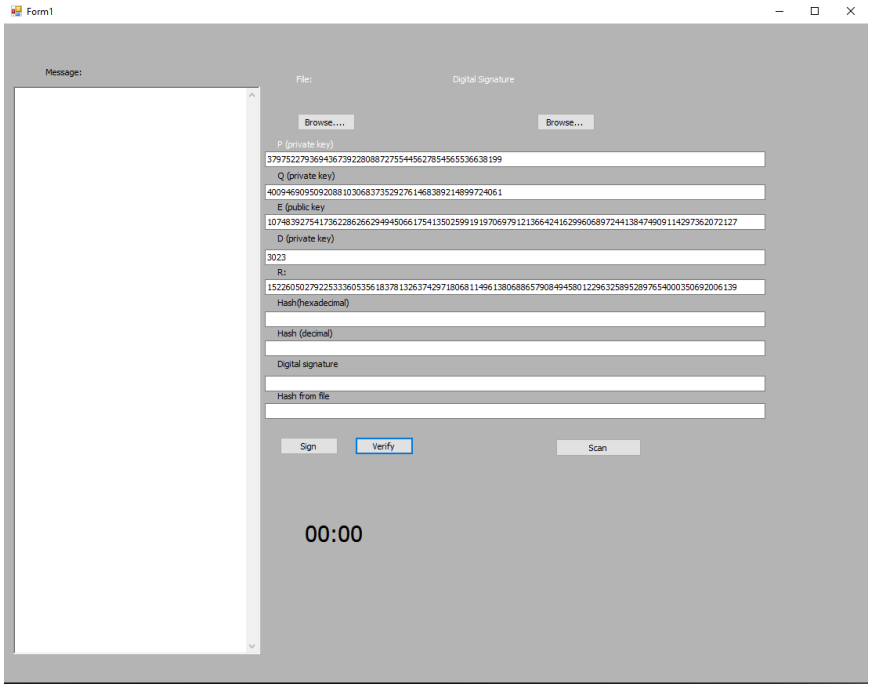


Figure 4.7 the first screen of the program

As shown in figure 4.7 in the left inside the white space where the text or file is converted to hexadecimal for easy handling of content, also clarifying the right values used by RSA and the location of the fingerprint, as well as the possibility of creation and verification of the digital signature.

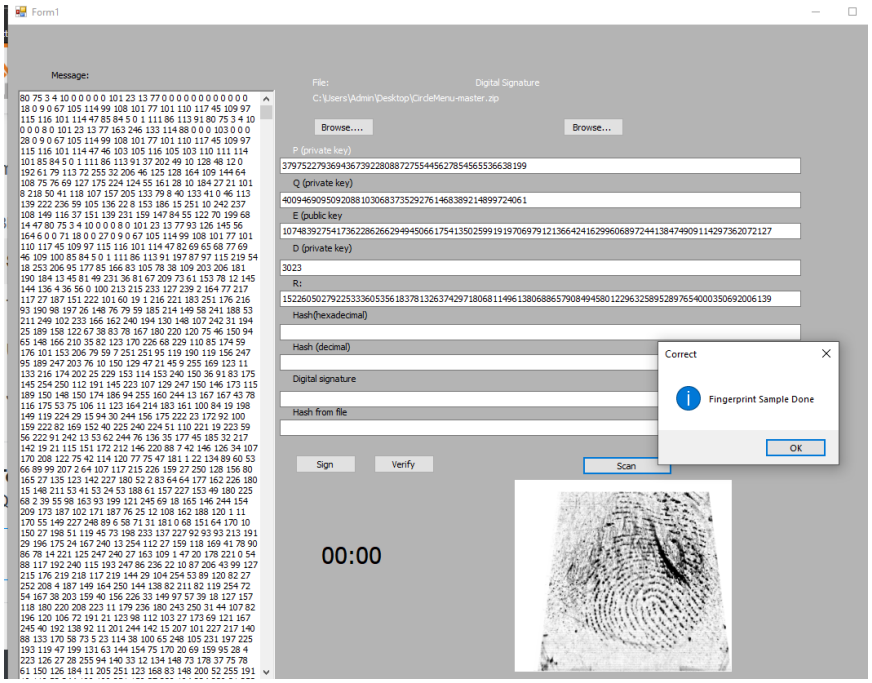


Figure 4.8 the process of inserting the file and entering the fingerprint

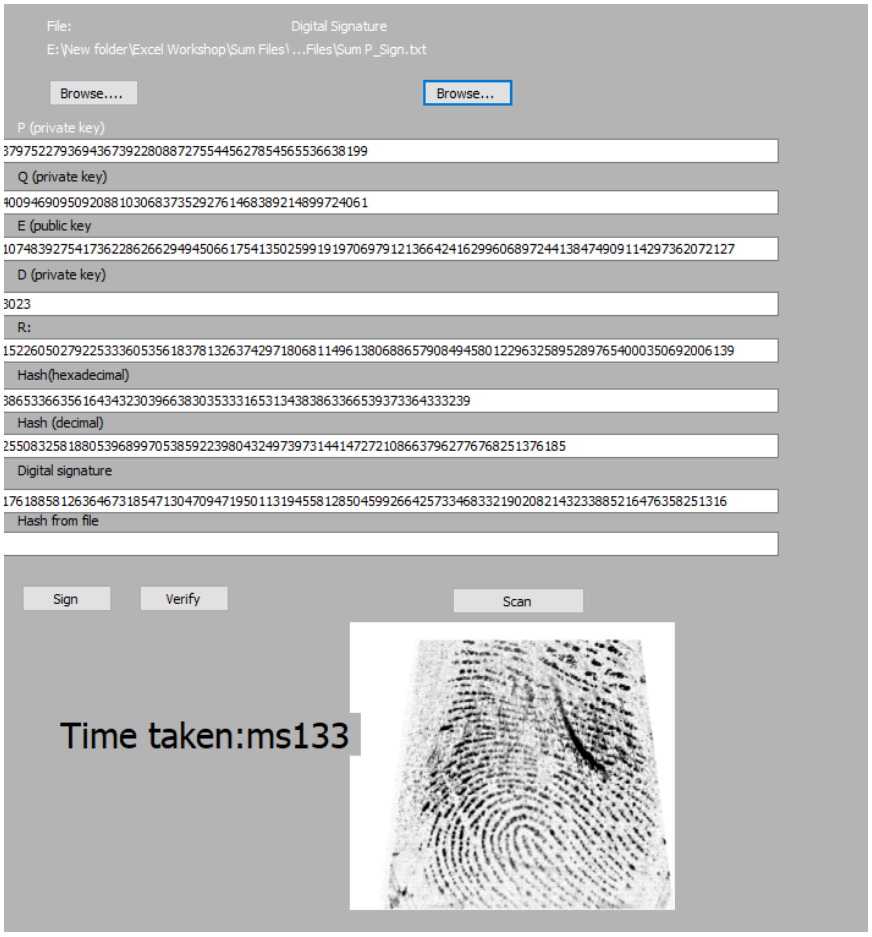


Figure 4.9 the process of creating a digital signature is based on the value of the fingerprint where the Pressing on sign is for calculating the message digest or hash value.

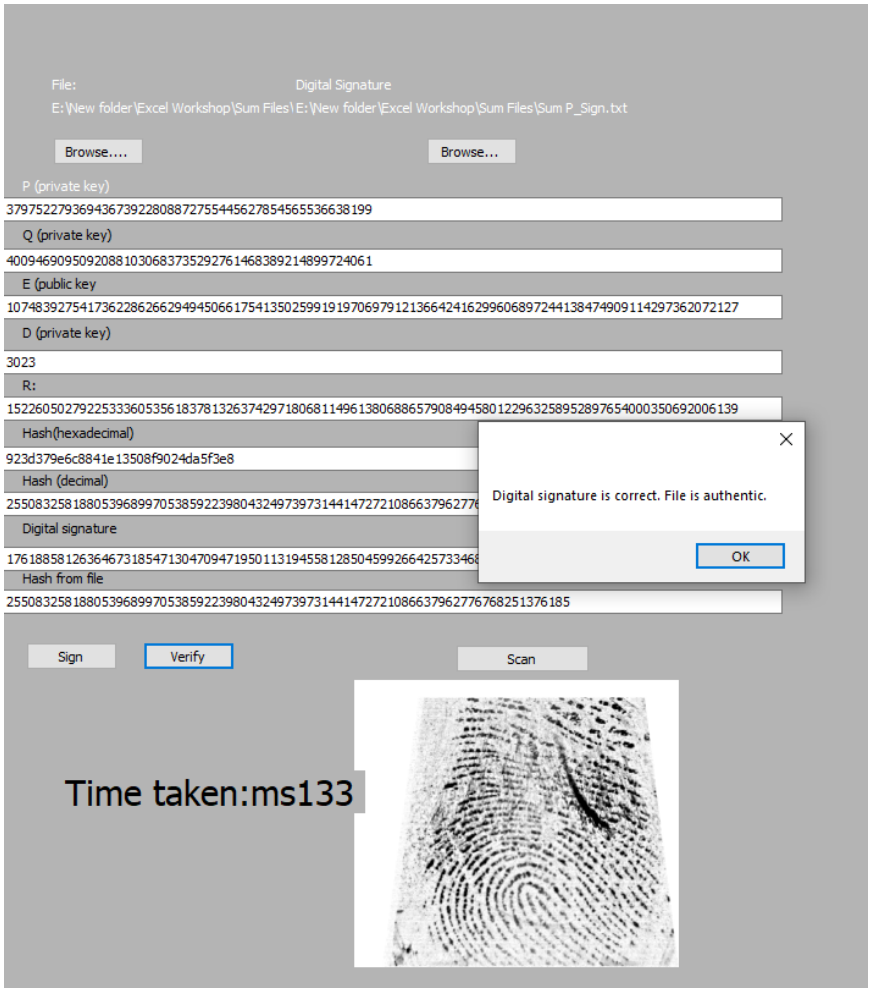


Figure 4.10 This screen enables the user to verify the file if it has been modified or not by using the administrator to verify the signature and compare the existing file with the original signature and give the confirmation or rejection of the file.

4.2.3 System three (Comparing system)

The function of this system is to calculate the Hash value for a given file, while the value is stored in its own file, the possibility of retrieving the value and verifying that if the file is modified or not. This system provides the possibility of using more than MDM Hash algorithms

such as (SHA family, MD5). This program was used to build comparison tables within this thesis as shown in figure 4.11, figure 4.12, and figure 4.13 until figure 4.19, as illustrated by the screens:

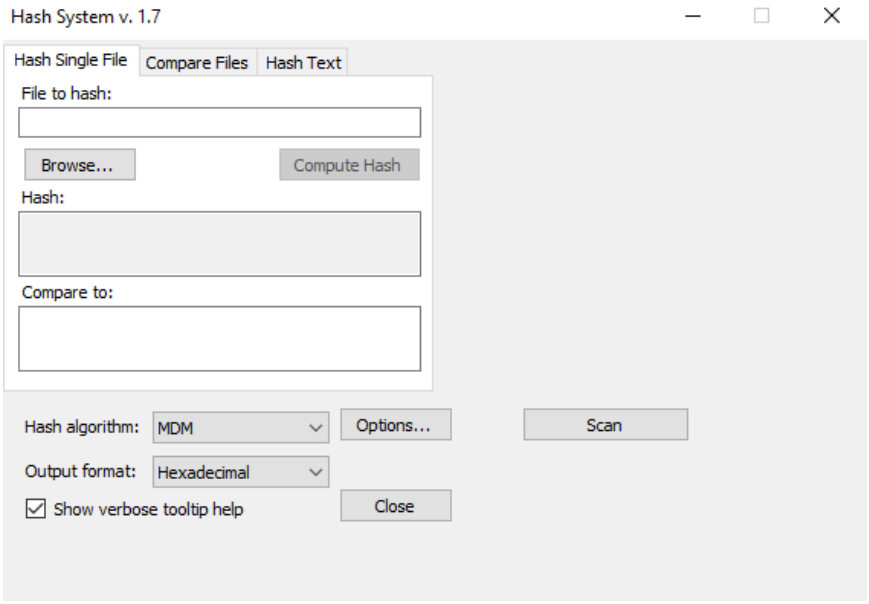


Figure 4.11 the first screen of the comparison program, and choosing the file to be hashed, where it is determined by clicking on browse the file and choosing to work Hash.

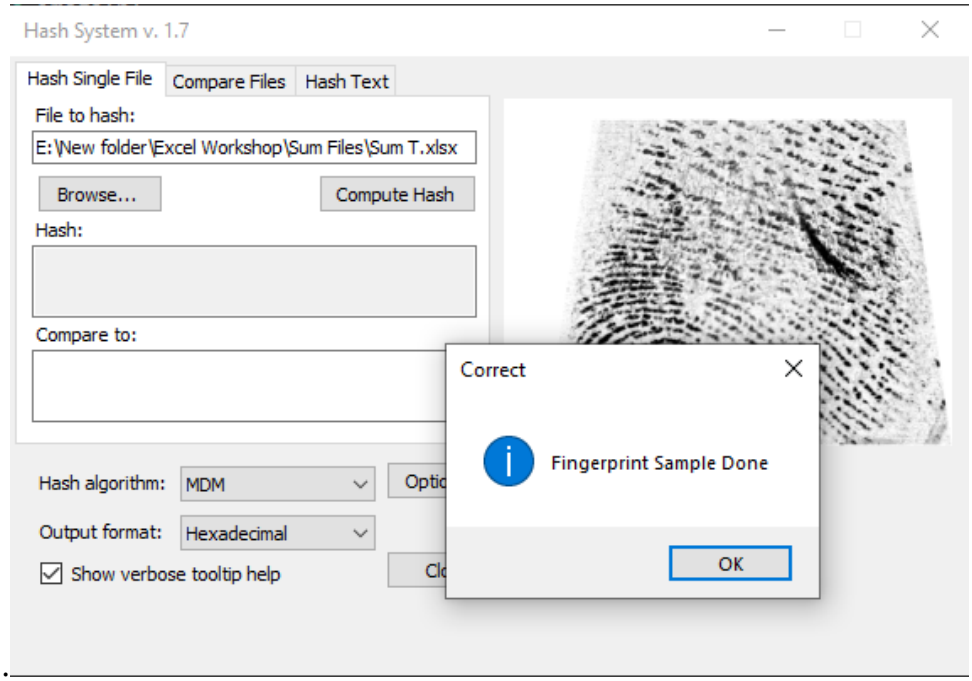


Figure 4.12 after selecting the file, the fingerprint is used to determine the values to work.

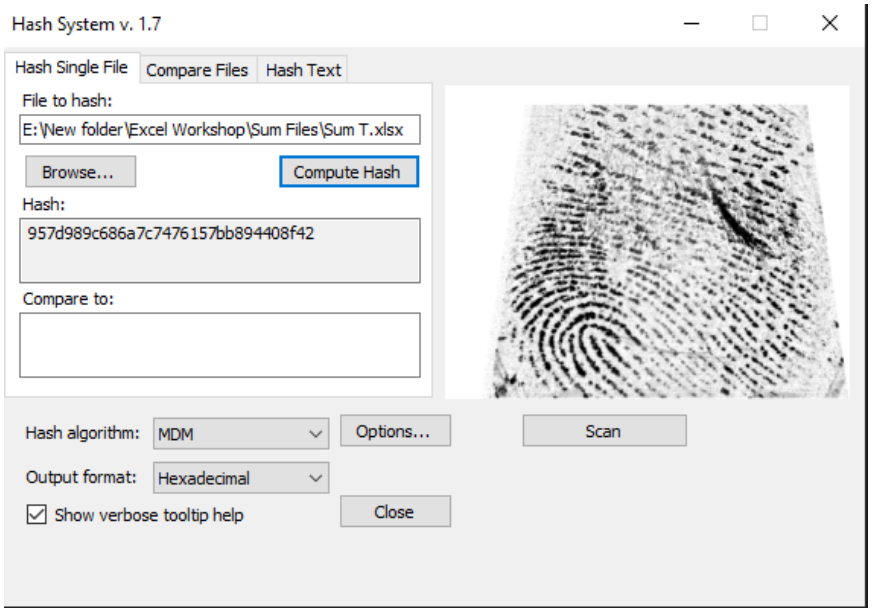


Figure 4.13 the Hash value is determined

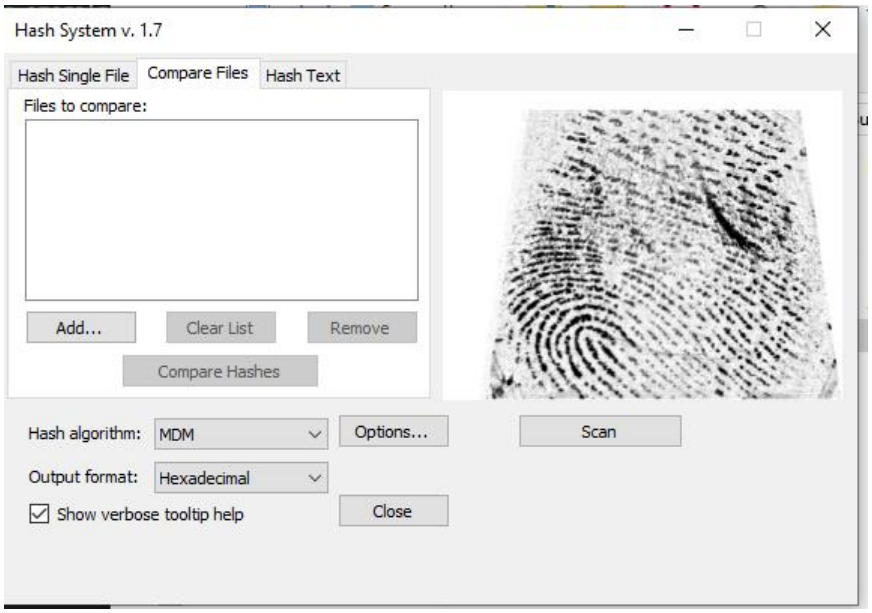


Figure 4.14 choose a comparison command to check the integrity of the file.

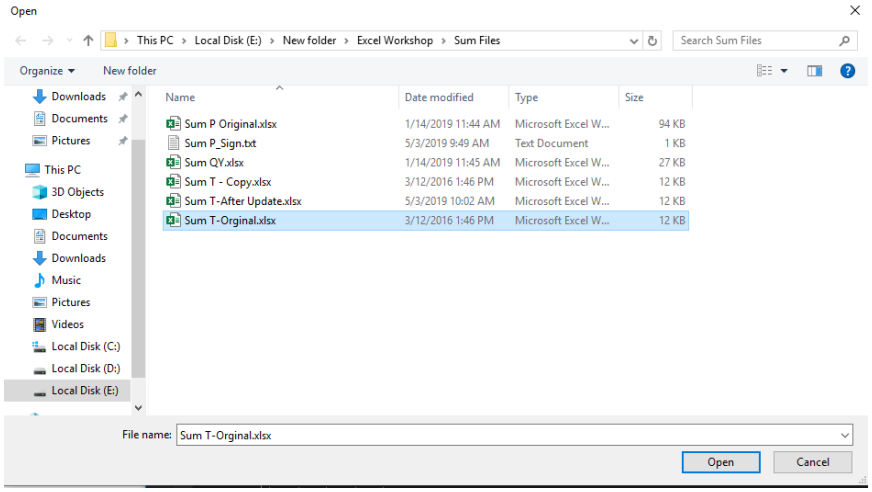


Figure 4.15 the file to be compared is chosen from the defined storage locations.

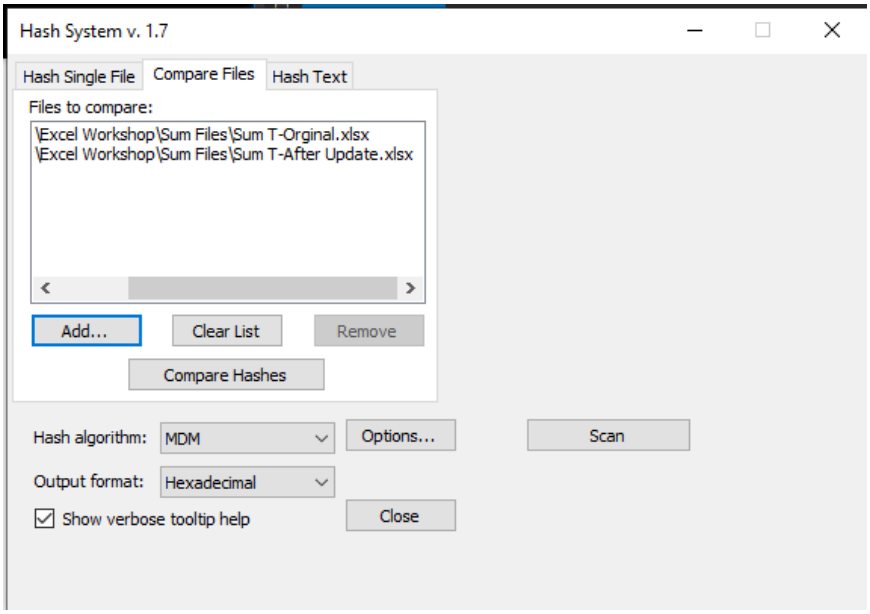


Figure 4.16 an additional command is pressed to confirm the file.

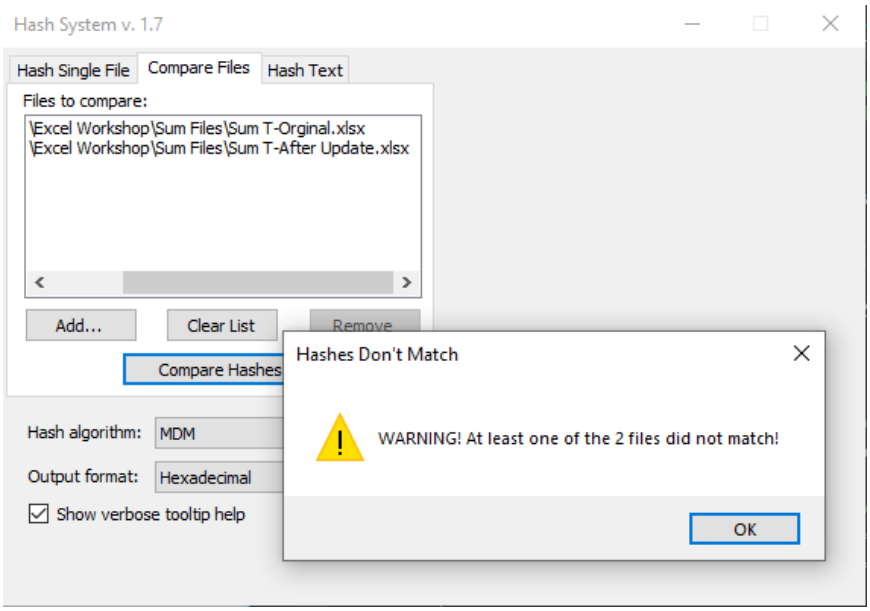


Figure 4.17 when the file has been manipulated, the response is rejected and the result of the comparison is negative if the file has been modified.

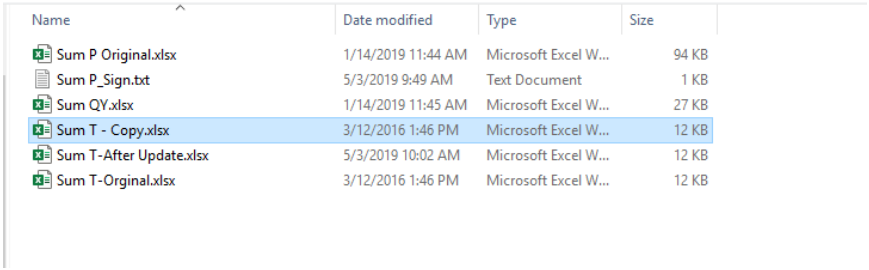


Figure 4.18 choose the proper file for comparison.

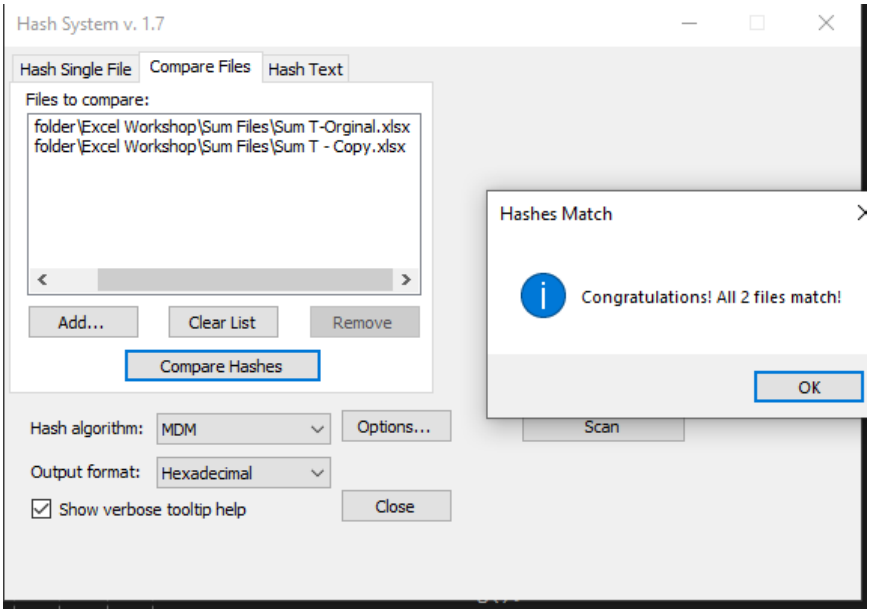


Figure 4.19 the results of the comparison are true and that the file is not modified.

4.3 Comparative study on MDM

An illustrative table for new algorithm implementations and calculation of the time taken to implement, and comparing results with MD5 and SHA256, These results were selected through

application in different companies for the purposes of checking the security of the algorithm and the possibility of dealing with them in the real environment, and the results were satisfactory to a large extent.

Table 4.1 Comparative time of execution measured by (ms).

File size	MDM	MD5	SHA256
1 KB	62	185	563
1038 KB	190	230	365
11 bytes	85	100	153
1114 KB	153	194	271
131 KB	190	209	257
144 KB	180	212	230
1551 KB	211	222	230
175 KB	185	279	236
17678 KB	345	387	663
2 KB	171	220	200
2118 KB	180	212	335
265 KB	199	203	220
345 KB	163	279	247
4 KB	63	166	215
5 KB	185	190	212
56 KB	181	189	247
7131 KB	91	241	817

868 KB	163	238	286
93587 KB	1056	1080	2425
2662688 KB	6323	20926	70710
Average	518.8	1298.1	3944.1

MDM/MD5 = 39.97%

MDM/SHA = 13.15%

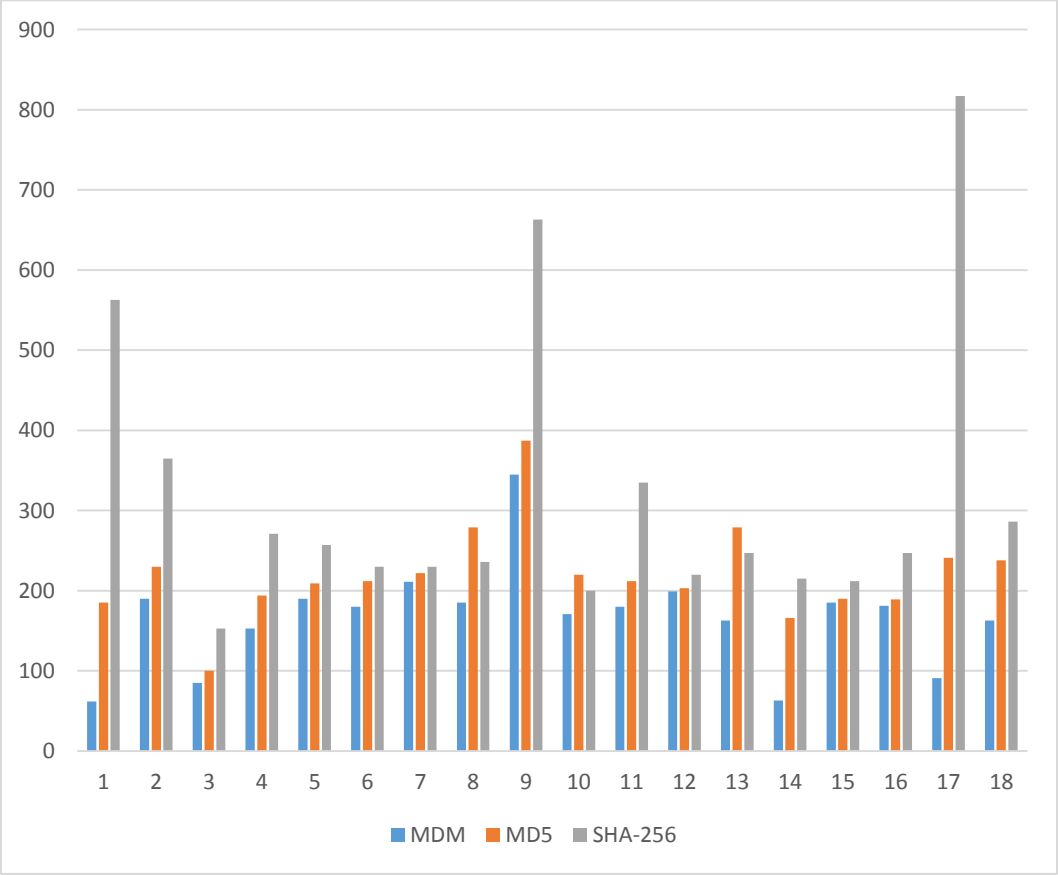


Figure 4.20 Comparative time of execution measured by (ms).

Table 4.3.1 demonstrates and as the graph 4.3.2 analyzes that the time taken by the new algorithm is significantly lower than the time of the other algorithms which means the new algorithm is faster than the others.

4.4 Test result

The study has made a small experiment to apply the MD5 in Chapter III and it has been reported that the same files will be tested but on the new algorithm; here are the results.

Table 4.2 MDM results

File Name	File size	Hash value
Document 1	11 bytes	ab5bee7e546c530af02b7091d2ad50c5
document 2	11 bytes	ab5bee7e546c530af02b7091d2ad50c5
pdf 1	265 KB	225f4dc9810272e6f27fe8356ade6f47
pdf 2	345 KB	170ebcecbf1e4c2ada657b9e2a3782c5
word 1	12.2 KB	f8cc92cdf4343549b34173bff8891955
word 2	12.2 KB	44701676e0d19feed4c2ab05786d4a16
Document 3	11 bytes	ef31ff889c7dc647b7e162b5a067018c

4.5 Proving attacks

This section presents some of the hacking attempts on the proposed algorithm, then compared with MD5, where the programs that have been used to find the original files of the MD5, cannot be enabled to penetrate the new algorithm MDM. The programs that are used are Ophcrack, Cain and able. The word Cisco was used to do the Hash and the original word was searched within these programs when applied to the value of the MD5 it is found, but when searching after the application the MDM value it could not be found as shown in figure 4.21 until figure 4.30.

Cisco = > MD5=> dfeaf10390e560aea745ccba53e044ed

Cisco => MDM => 5553cc030cd6a34ee9ef93784c72bfe5

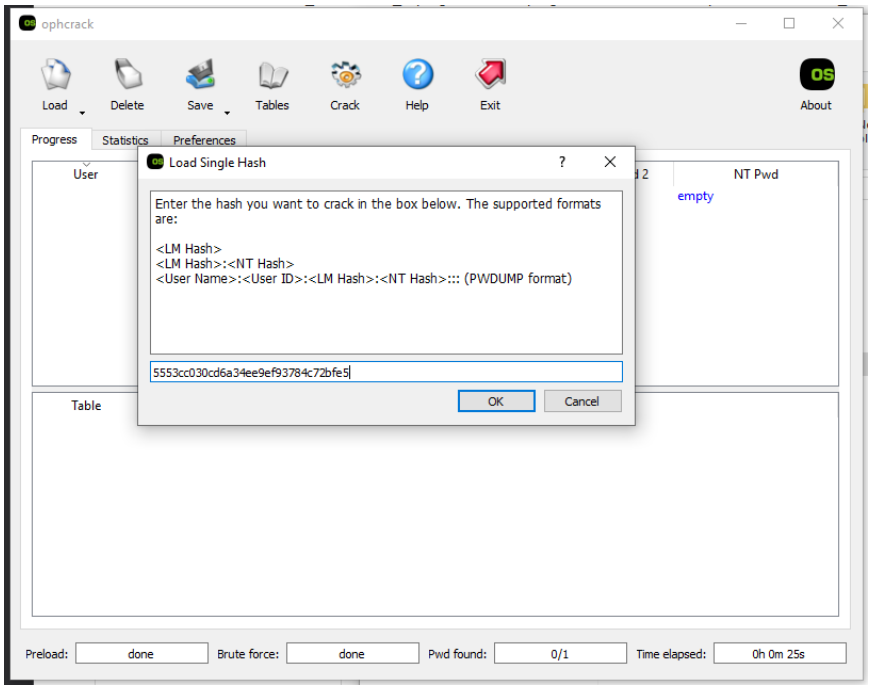


Figure 4.21 Ophcrack on MDM

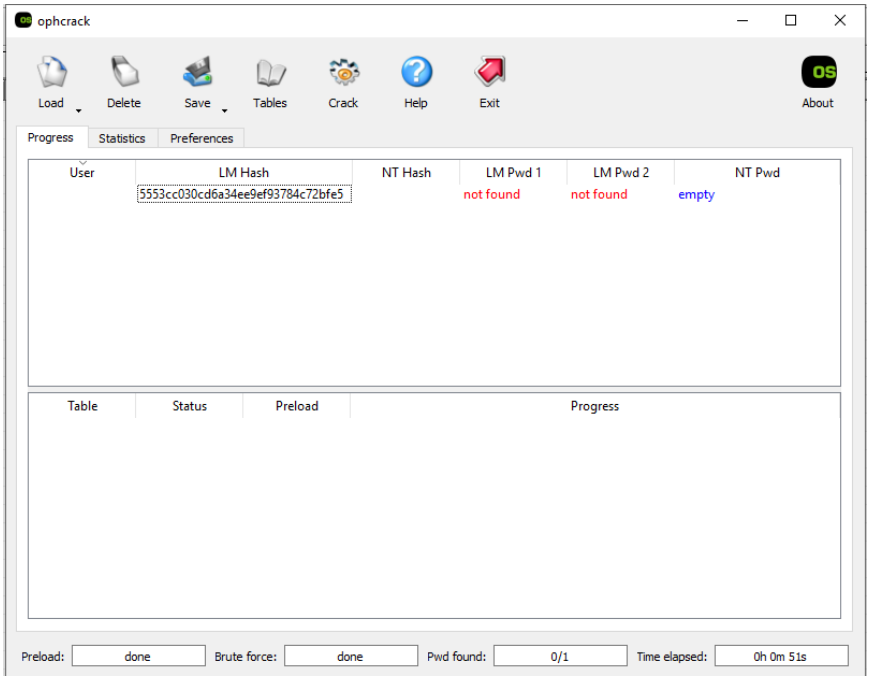


Figure 4.22 Ophcrack on MDM

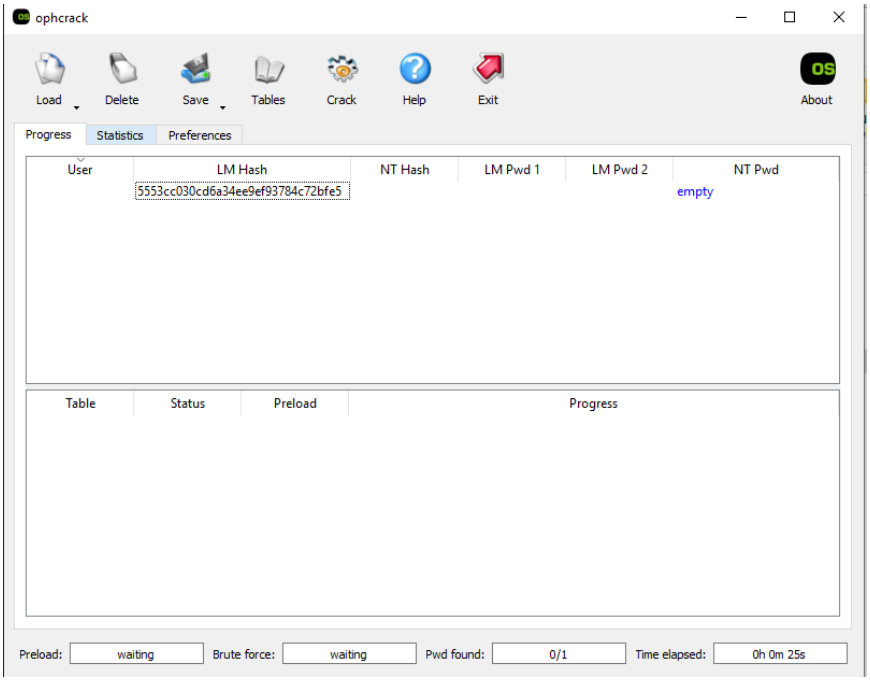


Figure 4.23 Ophcrack on MD5

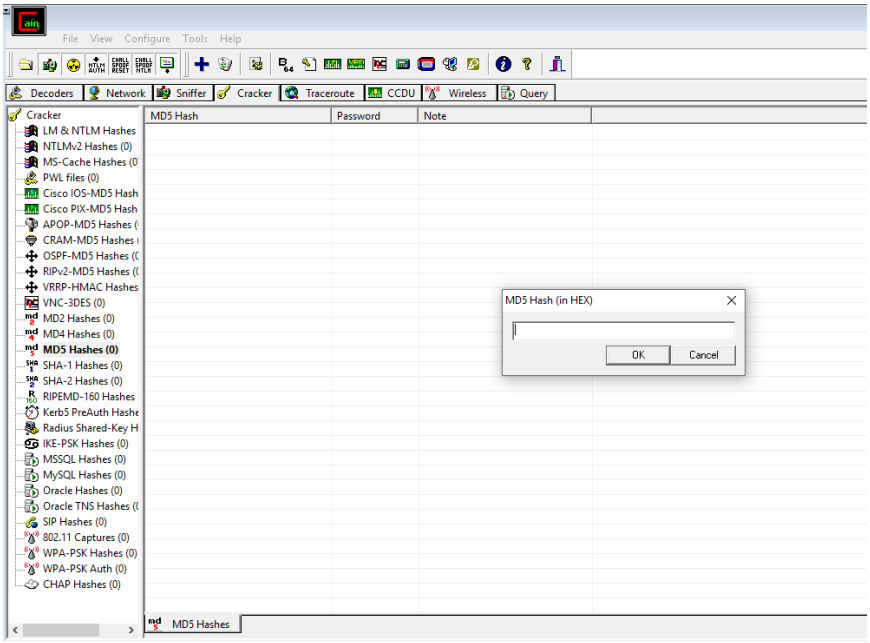


Figure 4.24 Cain and able on MD5

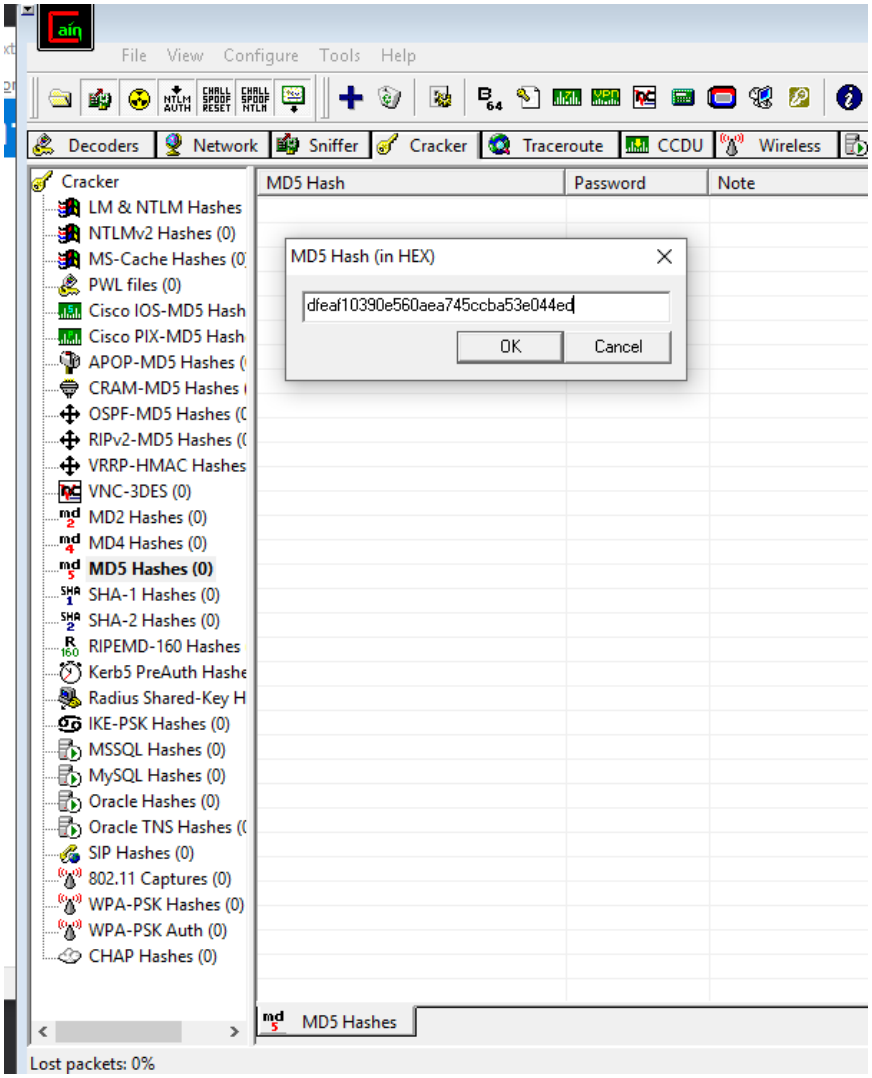


Figure 4.25 Cain and able on MD5

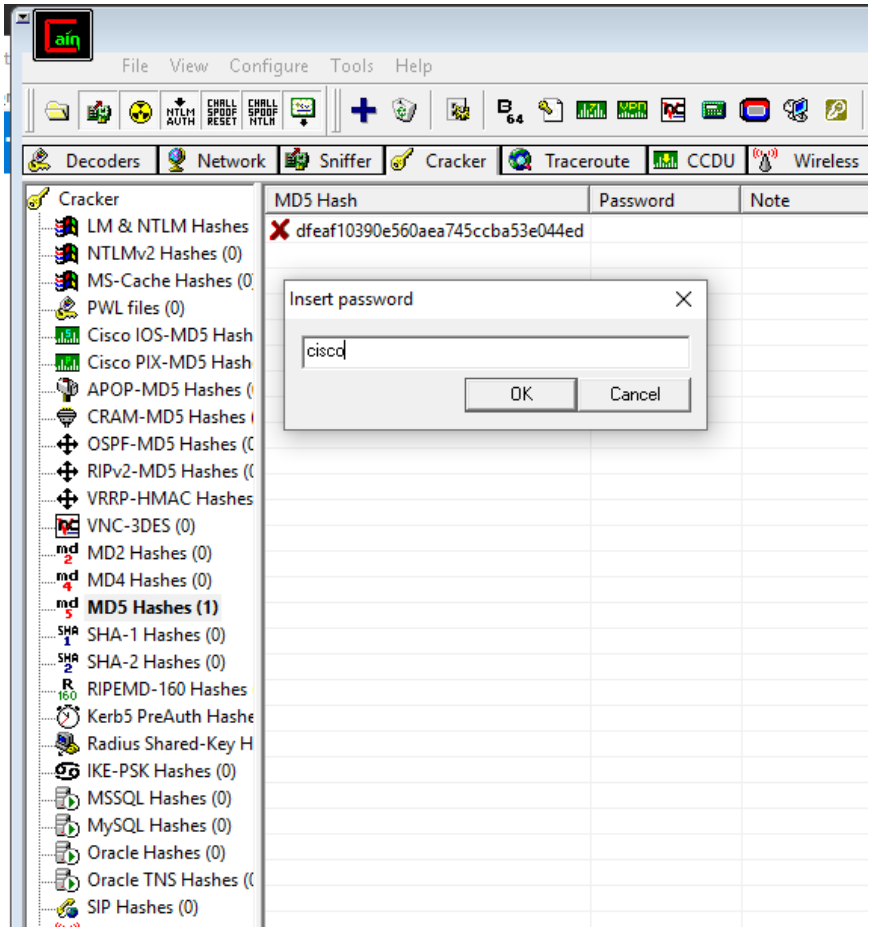


Figure 4.26 Cain and able on MD5

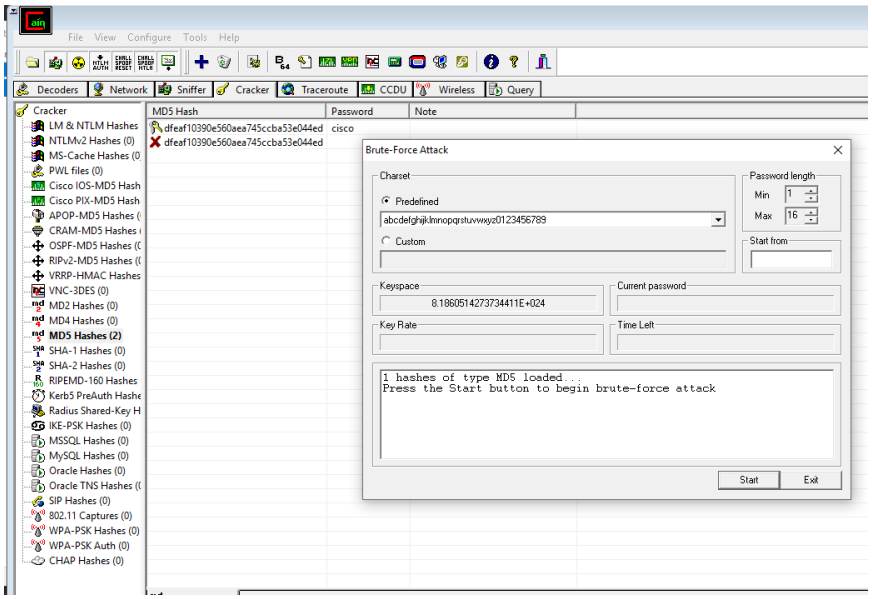


Figure 4.27 Cain and able on MD5 where the name is found

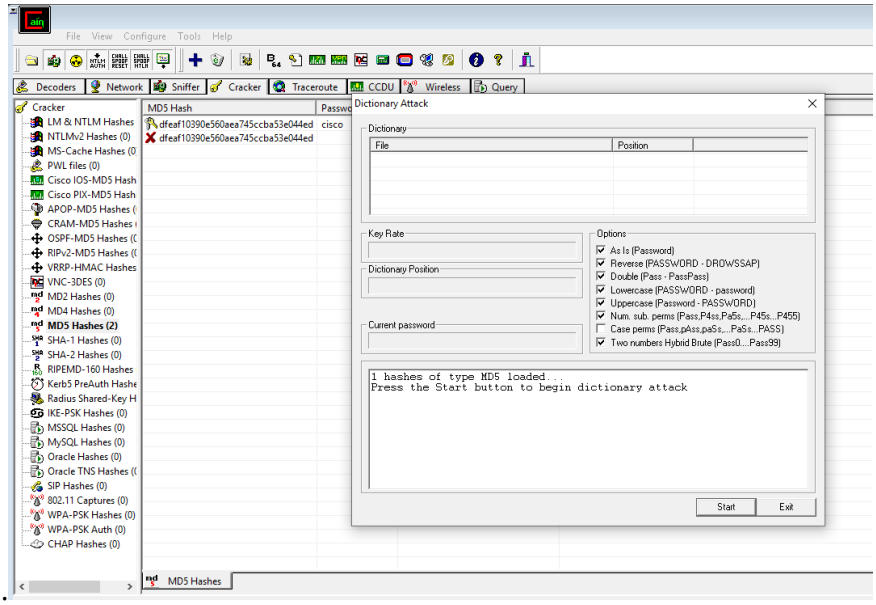


Figure 4.28 Cain and able on MDM

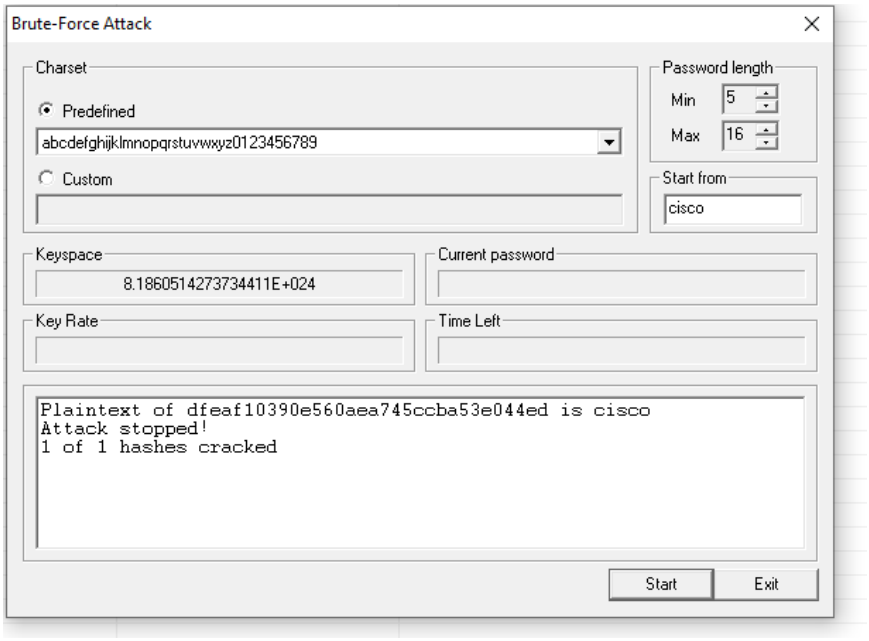


Figure 4.29 Cain and able on MD5

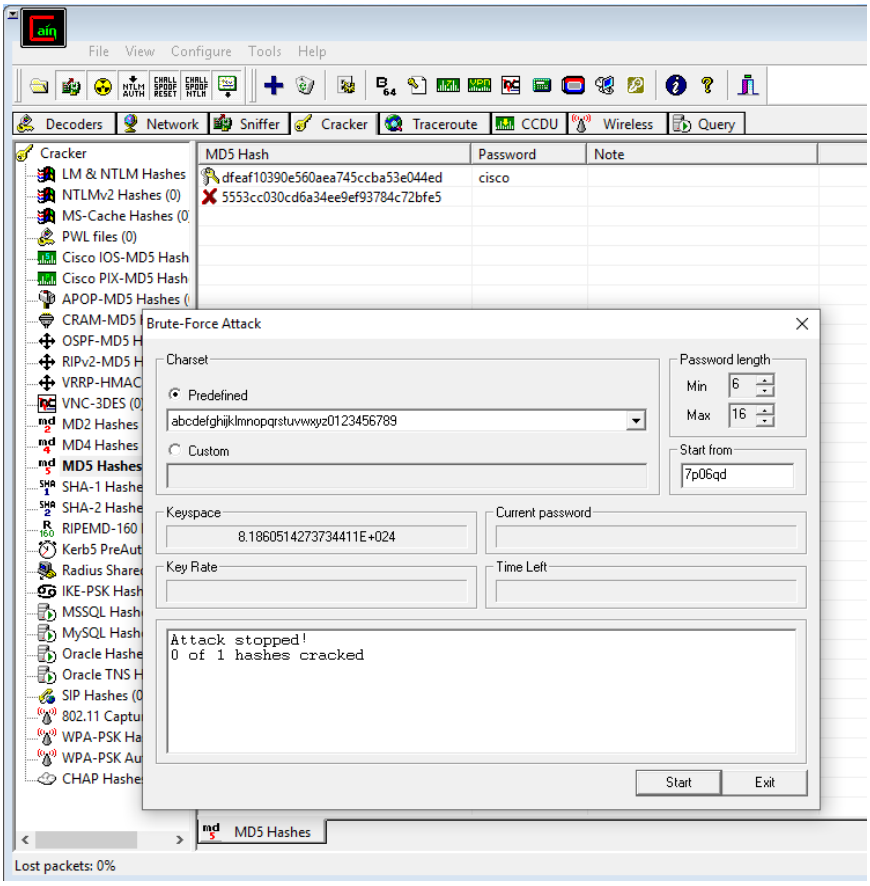


Figure 4.30 Cain and able on MDM

Chapter Five

Conclusions and Future Work

Chapter Five

Conclusion and Future Work

5.1 Conclusion

- The goal of this research thesis is to improve and strengthen the performance of the MD5 algorithm and to solve the problem of collision attack, through changing the constant values used in the original algorithm. The enhancement has been achieved through replacing the constant values with variable values obtained from user's fingerprint, with retention of the structure of the original algorithm. Based on the practical experience of modifying the MD5 algorithm and testing the new algorithm (MDM) , the following conclusions can be drawn:
- Obtaining a new algorithm that is strengthened by the fingerprint technology that supports the integrity of files, and is one of the first technologies that use the asymmetric key in the Hash operations.

- Obtaining a Secure digital signature.
- The digital signature mechanism provides authentication for the sender's identity in terms of the integrity of the data sent by the other party, even if the communication channel is unsafe.
- Getting an algorithm that supports maintaining passwords, without fear of hacking.

- Providing an algorithm that is clearly much faster compared with the original algorithm and the SHA algorithm.
- The results showed that the attack software was not able to penetrate the proposed algorithm.
- The results showed the strength and security of the proposed algorithm to protect the user's data.
- The study obtained a good deal of reliability feedback from companies that used the MDM algorithm, the application was made through them to ensure the effectiveness of the algorithm.

5.2 Future work

Work in this algorithm may extend to current problems:

- Use more non-fingerprinting techniques such as eye iris and retina.
- Target samples for further study.
- Use the algorithm within larger software to activate more of security.
- Applying the algorithms with higher cost techniques such as iris with fingerprint.

References

- *(PDF) A Study on Fingerprint Hash Code Generation Based on MD5 Algorithm and Freeman Chain code*. Available from: https://www.researchgate.net/publication/322465126_A_Study_on_Fingerprint_Hash_Code_Generation_Based_on_MD5_Algorithm_and_Freeman_Chain_code.

- Alok kumar Kasgar, Jitendra Agrawal and Santosh Sahu, (2012). New Modified 256-bit MD5 Algorithm with SHA Compression Function, International Journal of Computer Applications (0975 – 8887) Volume 42– No.12, March 2012, Available: https://www.researchgate.net/publication/258650505_New_modified_256-bit_MD5_Algorithm_with_SHA_Compression_Function.

- Boer, B. and Bosselaers, A. (1994) 'Collisions for the compression function of MD5', in *EUROCRYPT '93*, *Lecture Notes in Computer Science*, Vol. 765, pp.293–304.

- C.G Thomas, Thomas.Ro.Jo,(2015), *A Comparative Study on Different Hashing Algorithms,* International Journal of Innovative Research in Computer And Communication Engineering, **ISSN (Online): 2320-9801.**

- Cisco, (2007). Cisco IOS software integrity assurance- Cisco.com. (on-line) Available: https://tools.cisco.com/security/center/resources/integrity_assurance.html.

- collision attack for MD5?'. *Cryptology ePrint Archive*, Report 2008/391, available at http://eprint.iacr.org/2008/391.

- D Rachmawati*, J T Tarigan1* and A B C Ginting, (2018). A comparative study of Message Digest 5(MD5) and SHA256 algorithm, 2nd International Conference on Computing and Applied Informatics 2017 IOP Publishing IOP Conf. Series: Journal of Physics: Conf. Series 978 (2018) 012116 doi :10.1088/1742-6596/978/1/012116, Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia *Email: dian.rachmawati@usu.ac.id, jostarigan@usu.ac.id, realardi@gmail.com, Available: file:///C:/Users/mero/Desktop/MD5/Rachmawati_2018_J._Phys.__Conf._Ser._978_012116.pdf.

- D. Florencio and C. Herley, ―A large-scale study of web password habits, ‖ Proceedings of the 16thInternational conference on the World Wide Web, 2007. *(PDF) Fingerprint Recognition System: Design & Analysis*. Available: https://www.researchgate.net/publication/247773759_Fingerprint_Recognition_System_Design_Analysis .

- Dobbertin. Ha (1996). Cryptanalysis of MD4. German information security agency.

- Esmael V. Maliberan, Ariel M. Sison, Ruji P. Medina, (2018). A New Approach in Expanding the Hash Size of MD5, Graduate Programs, Technological Institute of the Philippines, Quezon City, Philippines, Available: file:///C:/Users/mero/Downloads/3292-6925-1-PB.pdf.

- John Black, Martin Cochran and Trevor Highland, (2006). A Study of the MD5 Attacks: Insights and Improvements, University of Colorado at Boulder, USA

www.cs.colorado.edu/~jrblack, ucsu.colorado.edu/~cochranm jrblack@cs.colorado.edu,

cochranm@cs.colorado.edu 2 University of Texas at Austin, USA

trevor.highland@gmail.com, Available:

https://link.springer.com/content/pdf/10.1007%2F11799313_17.pdf.

- K. Krishna Prasad & P. S. Aithal. A Study on Fingerprint Hash Code Generation Based on MD5 Algorithm and Freeman Chain Code. International Journal of Computational Research and Development, Volume 3, Issue 1, Page Number 13-22, 2018.

- K. Wiesner, M. Foth, M. Bilandzic, and H. Krcmar. (2009). Restrictions and Constraints in mobile narratives for place-based community engagement. In *Community Practices and Locative Media Workshop, MobileHCI*, University of Bonn, Bonn, 2009.

- Kahate. At (2008). Cryptography and network security (Second edition), India.

- Klima, V. (2005). Finding MD5 collisions on a notebook PC using multi-message Modifications. *Cryptology ePrint Archive*, Report 2005/102, available at http://eprint.iacr.org/2005/102.

- Klima, V. (2005). Finding MD5 collisions on a notebook PC using multi-message Modifications. *Cryptology ePrint Archive*, Report 2005/102, available at http://eprint.iacr.org/2005/102.

- Klima, V. (2006). Tunnels in hash functions: MD5 collisions within a minute. *Cryptology ePrint Archive*, Report 2006/105, available at http://eprint.iacr.org/2006/105.

- Klima, V. (2006). Tunnels in hash functions: MD5 collisions within a minute.*Cryptology ePrint Archive*, Report 2006/105, available at http://eprint.iacr.org/2006/105.

- Kumar. Sa, Gupta. Pi, (2018), A Comparative Analysis of SHA and MD5 Algorithm, Piyush Gupta et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4492-4495.

- Libed. JA, Sison. AR &Dr.Medina. Ru. (2018). Enhancing MD5 Collision Susceptibility.

  Conference: 4th International Conference on Industrial and Business Engineering

Proceedings, At University of Macau. Available: https://www.researchgate.net/publication/328879023_Enhancing_MD5_Collision_Susceptibility.

- Libed. JA, Sison. AR &Dr.Medina. Ru. (2018). Improved MD5 through the extension of 1024 message input block. Conference: International Conference on Machine Learning and Machine Intelligence, At Hanoi, Vietnam. Available: https://www.researchgate.net/publication/328879020_Improved_MD5_through_the_extension_of_1024_Message_Input_Block.

- M. Stamp and R. M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Wiley 2007.

- Maltoni. D, Maio. D, Jain. A. K., & Prabhakar. S. (2009). Handbook of Fingerprint Recognition, Springer-Verlag, London, XVI, 494.

- McClelland. Me (2018). Data integrity checksum. (on-line) available: http://www.versity.com/blog/data-integrity-checksums.

- Naito. Yu, Kunihiro. No & Ohta. Ka, (2005). The University of Electro-Communications, Japan *f*tolucky, yu339, kunihiro, ota*g* @ice.uec.ac.jp Available: https://www.semanticscholar.org/paper/Improved-Collision-Attack-on-MD5-Sasaki-Naito/06036d71d60fd4a8a002f50bf2524ef9c3540717.

- Narayana. D. Ka, (2006). **A Meaningful MD5 Hash Collision Attack,** Master thesis, San Jose State University, San Jose, California.

- Piyush. GU, Sandeep. Ku, (2019), *A Comparative Analysis of SHA and MD5 Algorithm*, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4492-4495,

- Rachmawati. D, Tarigan. A B C Ginting, (2018*). A comparative study of Message Digest 5(MD5) and SHA256 algorithm*, 2nd International Conference on Computing and

Applied Informatics 2017, IOP Publishing. IOP Conf. Series: Journal of Physics: Conf. Series **978** (2018) 012116.

- Rivest, R. (1992) 'The MD5 Message-Digest Algorithm', Internet Engineering Task Force, Request For Comments, RFC 1321, April, (on-line) available at http://www.ietf.org/rfc/rfc1321.txt.

- Robshaw. Ma, (1994). Message Authentication with MD5, Burt Kaliski and Matt Robshaw RSA Laboratories 100 Marine Parkway, Suite 500 Redwood City, CA 94065 USA burt@rsa.com matt@rsa.com , Available: https://pdfs.semanticscholar.org/cad3/6d5c4fdf768154b7bfafa5e1a33a1abf0062.pdf.

- S. Thomsen, (2007). *The Grindahl Hash Functions*, Conference paper, Available: https://link.springer.com/chapter/10.1007/978-3-540-74619-5_3 .

- Sasaki. Y, Naito. Y & Kunihiron. N, Ohta, K, (2005). Improved collision attack on MD4. The University of Electro-Communications, Japan {tolucky, yu339, kunihiro, ota} @ice.uec.ac.jp, Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=B8DED5D7D255B36A2CA82B5D6D5EFA71?doi=10.1.1.59.5096&rep=rep1&type=pdf.

- Schaettgen. Ni, Levy. Di & Schelnast. Ju, Socol. So, Digital signatures paving the way to a digital Europe (on-line) available: http://www.adlittle.fr/sites/default/files/viewpoints/ADL_2014_Digital-Signatures.pdf.

- Selinger. Pe, (2011). MD5 collision demo (on-line) Available: https://www.mscs.dal.ca/~selinger/md5collision/.

- Stallings, W. (2006). Cryptography and Network Security: Principles and Practice.

- Stallings. Wi (2004). Cryptography and network security (fourth edition), Prentice Hall publication.

- Stevens, M. (2006). Fast collision attack on MD5. *Cryptology ePrint Archive*, Report 2006/104, available at http://eprint.iacr.org/2006/104.

- Stevens, M. (2007). On collisions for MD5. TU Eindhoven MSc thesis, June. available at http://www.win.tue.nl/hashclash/On%20Collisions%20for%20MD5%20-%20M.M.J.%20Stevens.pdf.

- Stevens, M. (2012). Attacks on hash functions and applications. Universiteit Leiden PhD thesis, to appear, available at http://marc-stevens.nl/research.

- Stevens, M., Lenstra, A.K. and de Weger, B.M.M. (2007). Chosen-prefix Collisions for MD5 and colliding X.509certificates for different identities. In *EUROCRYPT 2007, Lecture Notes in Computer Science*, Vol. 4515, pp.1–22.

- Stevens, M., Lenstra, A.K. and de Weger, B.M.M. (2007). Chosen-prefix collisions For MD5 and colliding X.509 certificates for different identities. in *EUROCRYPT 2007, Lecture Notes in Computer Science*, Vol. 4515, pp.1–22.

- Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A. and De Weger, B.M.M. (2009). Short chosen-prefix collisions for MD5 and the creation of A rogue CA certificate. in *CRYPTO 2009, Lecture Notes in Computer Science*, Vol.5677.

- Thomas. Co (2009). Introduction to algorithm (third edition). The MIT press. Lai. Xu, Liang. Ji, (2005). Improved collision attacks on hash function MD5. Department of Computer Science and Engineering Shanghai Jiao Tong University Shanghai 200240, China Email: luckyaa@sjtu.edu.cn.

- Tulyakov. Se, Farooq. Fa, Govindaraju. Ve, (2005), *Symmetric Hash Functions for Fingerprint Minutiae*, SUNY at Buffalo, Buffalo NY 14228, USA.

- Wang, X. and Yu, H. (2005). How to break MD5 and other hash functions. In *EUROCRYPT 2005, Lecture Notes in Computer Science*, Vol. 3494, pp.19–35.

- Wang, X., Feng, D., Lai, X. and Yu, H. (2004). Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. *Cryptology ePrint Archive*, Report 2004/199, Available at http://eprint.iacr.org/2004/199.

- Wang, X., Yao, A. and Yao, F. (2005a). New collision search for SHA-1. *CRYPTO 2005 Rump session*, available at http://www.iacr.org/conferences/crypto2005/r/2.pdf.

- Wang, X., Yin, Y.L. and Yu, H. (2005). Finding collisions in the full SHA-1. In *CRYPTO 2005, Lecture Notes in Computer Science*, Vol. 3621, pp.17–36.

- Xie, T., Liu, F. and Feng, D. (2008). Could the 1-MSB input difference be the fastest collision attack for MD5?. *Cryptology ePrint Archive*, Report 2008/391, available at http://eprint.iacr.org/2008/391.

- Xie, T., Liu, F. and Feng, D. (2008). Could the 1-MSB input difference be the fastest Collision attack for MD5?. *Cryptology ePrint Archive*, Report 2008/391, available at http://eprint.iacr.org/2008/391.

- Xie, T., Liu, F. and Feng, D. (2008). Could the 1-MSB input difference be the fastest collision attack for MD5?. *Cryptology ePrint Archive*, Report 2008/391, available at http://eprint.iacr.org/2008/391.

- Xie, T., Liu, F. and Feng, D. (2013). Could the 1-MSB input difference be the fastest

Appendix A

Contract

Performance Improvement Plan (PIP)

Confidential

TO: Marwa Hussein Al- Awawdeh
FROM: Sondos Market Research
DATE: 3/1/2019
RE: Performance Improvement Plan (PIP)

The purpose of this Performance Improvement Plan (PIP) is to define serious areas of concern, gaps in your work performance, and allow you the opportunity to demonstrate improvement and commitment. This plan proposed for work in master thesis for **The Middle East University.**

Areas of Concern:

Registration of sales staff, documenting special files, contracts, offers and formal transactions of registrants on the company's system.

Observations, Previous Discussions or Counseling:

SHA256 was used, and now the new protocol was adopted after the experiment to be installed on the company's system.

Step 1: Improvement Goals: These are the goals related to areas of concern to be improved and addressed:

1.	**Ensure information security**
2.	**The quality of the protocol and the accuracy of its performance compared to previous protocols**
3.	**Easy to use and cost-effective**

Step 2: Activity Goals: Listed below are activities that will help you reach each goal:

Goal #	Activity	How to Accomplish	Start Date	Projected Completion Date
1.	Apply the protocol to the files to use the digital signature	Apply on PDF, DOCx, Excel sheets, any file need to be hashed, and power point.	3/15/2019	4/28/2019
2.	Apply the protocol to employee registration and protect passwords	Apply on the system for registration.	3/15/2019	4/28/2019
3.	News accuracy preserves data by working hacking attempts	Apply by using techniques Which was launched in the thesis.	3/15/2019	4/28/2019

Step 3: Resources: Listed below are resources available to you to complete your Improvement activities (may include other people's time or expertise, funds for training materials and activities, or time away from usual responsibilities.)

1.	Fingerprint scanner with its own SDK system
2.	Install the DLL file protocol on the company system
3.	Follow-up performance of the owners of experience in the company

Step 4: Expectations: The following performance standards must be accomplished to demonstrate progress towards achievement of each Improvement goal:

1.	Get a high degree of security for a digital signature on files used
2.	Having tightened security to keep passwords for employees
3.	Lack of penetration within the system by all techniques used
4.	Obtain the trust of users to support the trusted protocol
5.	Obtain a recommendation from the company proposed to support the Master's study

Step 5 Progress Checkpoints: The following schedule will be used to evaluate your progress in meeting your Improvement activities.

Goal #	Activity	Checkpoint Date	Type of Follow-up (memo/call/meeting)	Progress Expected	Notes
1.	Definition of supervisors of the proposed system	3/1/2019	meeting	Approval	Start work
2.	Explanation mechanism for employees	3/2/2019	meeting	Understanding with the correct experience	Apply the protocol on the system
3.	Follow up work weekly	3/10/2019	meeting	Work is going well	Make notes
4.	Take a recommendation and rely on the new protocol to work	4/28/2019	meeting	Approval	----------

Follow-up Updates: You will receive feedback on your progress according to the following schedule:

Date Scheduled	Activity	Conducted By	Completion Date
3/10/2019	Follow up work weekly	[IT / Manager]	3/11/2019
3/24/2019	Follow up work weekly	[IT / Manager]	3/25/2019
4/21/2019	Follow up work weekly	[IT / Manager]	4/22/2019

Timeline for Improvement, Consequences & Expectations:

Effective immediately, you are placed on a **(50)**-day PIP. During this time you will be expected to make regular progress on the plan outlined above. Failure to meet or exceed these expectations, or any display of gross misconduct will result in further disciplinary action, up to and including termination. In addition, if there is no significant improvement to indicate that the expectations and goals will be met within the timeline indicated in this PIP, your employment may be terminated prior to **(50)** days. Furthermore, failure to maintain performance expectations after the completion of the PIP may result in additional disciplinary action up to and including termination.

The PIP does not alter the employment-at-will relationship. Additionally, the contents of this PIP are to remain confidential. Should you have questions or concerns regarding the content, you will be expected to follow up directly with me.

We will meet again on as noted above to discuss your Performance Improvement Plan. Please schedule accordingly.

Signatures:

Researcher Name: Marwa Hussein Al- Awawdeh

Researcher Signature:

Date: 3/1/2019

Supervisor/Manager Name: Osama Abushahadeh

Date: 3/1/2019

IN the Name of God the Compassionate the Merciful

Sondos Market Research No:152/2019

Personnel, IT Department Date:30/4/2019

Certificate of recommendation and accreditation

The Department of Technology in **Sondos Market Research** hereby certifies that it has entered into a contract with Ms. Marwa Hussein Issa Al-Awawdeh has applied study for the purposes of graduate studies - Master of Computer Science - on the date of 3/1/2019 until 4/28/2019 and accordingly has adopted its technology for the protocol supported to be studied and applied to the systems of the company from the digital signature and the protection of employees data for passwords, and this certificate has been given upon request to document the work within the study ,The management has to adopt the work with the above mentioned in terms of its confidentiality of work within the framework of the company.

Date: 30/4/2019

بسم الله الرحمن الرحيم

Sondos Market Research
الإداره العامه / قسم التكنولوجيا

الرقم :152/2019
التاريخ:30/4/2019

شهادة التوصية والاعتماد

يشهد قسم التكنولوجيا في **Sondos Market Research** بأن السيده مروة حسين عيسى العواوده قد تقدمت بطلب تطبيق لدراستها لغايات الدراسات العليا – ماجستير في علم الحاسوب – في تاريخ 3/1/2019 وحتى تاريخ 4/28/2019 وعليه قد تم اعتماد تقنيتها لبروتوكول المدعم لدراستها وتطبيقه على انظمة الشركه من توقيع الكتروني و حماية بيانات الموظفين لكلمات المرور , وتعطى هذه الشهاده بناء على طلبها لتوثيق العمل داخل الدراسه, وموافقه من الإداره على اعتماد العمل مع السيده اعلاه ضمن شروطها المتمثله بسرية العمل ضمن اطار الشركه.

التاريخ :30/4/2019