جامعة الشرق الأوسط
MIDDLE EAST UNIVERSITY

A New Technique to Protect the Ownership of Images in Social Network using Encryption and Steganography Algorithms

تقنية جديدة لحماية ملكية الصور في الشبكات الاجتماعية باستخدام خوارزميات التشفير واخفاء المعلومات

Prepared by

Imad Shafi Khuffash

Supervisor

Dr. Hebah H. O. Nasereddin

A Thesis Submitted in Partial Fulfillment of the Requirements for Master Degree in Computer Information System

Department of Computer Information System

Faculty of Information Technology

Middle East University

Amman- Jordan

January, 2016

Authorization Statement

I, Imad Shafi Wadi Khuffash, authorize the Middle East University to supply a copy of my thesis to libraries, establishments or individuals upon their request.

Name :Imad Shafi Wadi Khuffash

Date : 3/11/2016........

Signature: [illegible]

Middle East University

Examination Committee Decision

This is to certify that the thesis entitled **"A New Technique to Protect the Ownership of Images in Social Networks using Encryption and Steganography Algorithms"** was successfully defended and approved in 2016.

Examination Committee Members	**Signature**
Dr. Hebah H. O. Nasereddin (Supervisor & Member) Associate Professor, Department Of Computer Information Systems Middle East University (MEU)	[illegible]
Dr. Mudhafar Al-Jarrah (Chairman) Assistant Professor, Department of Computer Science Middle East University (MEU)	[illegible]
Dr. Hisham Abu-Saymeh (External Member) Associate Professor. Applied Science University	[illegible]

Acknowledgment

My special appreciation and thanks are to my supervisor Professor. Hebah H. O. Nasereddin who has been a tremendous mentor for me. I would like to thank him for supervising my research and for allowing me to grow as a researcher. His comments, guidance, and advice on my research have been helpful.

Special thanks also are given to my parents. Words cannot express how grateful I am to my parents for all the sacrifices that they have made on my behalf.

Dedication

To my father and my mother

To my brothers and my sister

To my friends

No words can make me express my gratitude and love.

To the martyrs' souls of Palestine.

الاهداء

يا من أحمل اسمك بكل فخر
يا من أفتقدك منذ الصغر
يا من يرتعش قلبي لذكرك
يا من أودعتني لله

أبي

إلى من بها أكبر وعليها أعتمد .. إلى شمعة متقدة تنير ظلمة حياتي..
إلى من بوجودها أكتسب قوة ومحبة لا حدود لها..
إلى من عرفت معها معنى الحياة
أمي الحبيبة

إلى القلوب الطاهرة الرقيقة والنفوس البريئة إلى رياحين حياتي (اخوتي وأصدقائي)

إلى من علمنا التفاؤل والمضي إلى الأمام، إلى من رعانا وحافظ علينا، إلى من وقف إلى جانبنا عندما صعبت علينا الطريق
الدكتورة هبة حسن ناصر الدين

التي نقول لها بشراك قول رسول الله صلى الله عليه وسلم :
"إن الحوت في البحر ، والطير في السماء ، ليصلون على معلم الناس الخير "

ABSTRACT

A New Technique to Protect the Ownership of Images in Social Network using Encryption and Steganography Algorithms

Prepared By:

Imad Shafi Wadi Khuffash

Supervised By:

Dr. Hebah H. O. Nasereddin

Steganography is the science of data embedding, through embedding data in other multimedia in such that no one could know the embedded data, thus hiding the existence of the secret data , usually ,steganography used to embed data into multimedia (images, audio or video), one of the used algorithm in steganography is the Least significant bit (LSB) .which sometimes can be detected by robbers, where this will affect privacy of the hidden data, thus, this thesis will enhance the security of the embedding data by using LSB with secure hash function family (SHA) and advanced encryption standard (AES) which is known as Rijndael.

Secure Hash Algorithm (SHA) Advance encryption standard (AES) will be used to enhance the privacy of images in online social network (OSN), The implemented technique includes two stages, signature encryption and embedding; in signature encryption the (SHA / AES) algorithm will encrypt signature of the user to obtain encrypted value, while embedding will embed the encrypted value into the image, that's how the technique will guarantee that no signed images could be re-upload by other users

the results of the proposed methods have been compared with other study, the results indicates that the noise caused by embedding the encrypted signature is too small, this can be viewed in higher PSNR values and lower RMSE, this observation is caused by the fact that lower significant of bits are needed to be replaced in order to hide the encrypted signature.

Keywords: Steganography, Least Significant Bit, Secure Hash Algorithm, Advanced Encryption Standard, Online Social Network.

الملخص

تقنية جديدة لحماية ملكية الصور في الشبكات الاجتماعية باستخدام خوارزميات التشفير واخفاء المعلومات

إعداد : عماد شفيع وديع خفش

إشراف :د. هبة حسن عثمان ناصر الدين

تستخدم steganography لاخفاء البيانات داخل ملفات الوسائط المتعددة (ملفات الصور , الصوت او الفيديو) وتعتبر خوازمية (القيمة الاقل تأثيرا LSB) اشهرها ,ولكن في بعض الحالات قد يتم الوصول الى البيانات المخفاة , مما يؤدي لانتهاك الخصوصية لهذه البيانات ,في هذه الأطروحة، يتم اقتراح طريقة جديدة لاخفاء توقيع صاحب الصورة داخل الصورةالتقنية الجديدة ستتضمن مرحلتين اساسيتين , مرحلة تشفير التوقيع , ومرحلة ارفاق التوقيع

في المرحلة الاولى (تشفير التوقيع) سيتم استخدام خوارزمية (AES) او خورازمية (SHA) لتشفير توقيع صاحب الصورة للحصول على قيمة مشفّرة ستسخدم لاثبات ملكية الصورة في شبكات التواصل الاجتماعي, بينما في المرحلة الثانية (ارفاق التوقيع) سيتم استخدام خوارزمية (القيمة الاقل تأثيرا LSB) لارفاق التوقيع داخل الصورة

تم مقارنة النتائج مع دراسة سابقة , حيث اظهرت النتائج الحصول على تشويش طفيف على الصور المستخدمة نتيجة ارفاق التوقيع المشفر داخل الصورة , يمكن استنتاج ذلك من خلال نتائج ال (PSNR)العالية نسبيا ونتائج ال (RMSE) القليلة , وهذا يرجع الى حقيقة ان استخدام خوارزمية (القيمة الاقل تأثيرا) في عملية ارفاق البيانات داخل الصور تعطي نتائج افضل من حيث الوضوح والقيمة المتأثرة من البكسل .

الكلمات المفتاحية : اخفاء البيانات, القيمة الأقل تأثيرا, شبكات التواصل الاجتماعي .

Table of content

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Meaning
LSB	Least Significant Bit
MD5	Merkle–Damgård Hash Function
OSN	Online Social Network
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
VIP	Very Important Person

CHAPTER ONE

Introduction

1.1 Introduction

Online Social networks (OSN)s such as MySpace, Facebook, Cyworld, and Bebo have attracted many users who use these sites into their daily events , which offers various technological and support a wide range fields of interests and activities. (Boyd et al., 2007)

Millions of users worldwide share, everyday, astronomical amounts of their private information through blogs, wikis and OSNs (Squicciarini et al.2010)

Therefore, web communities, companies and governments try to provide more security and privacy and adjust these services by taking the advantages of technological evolution in big data storage, cloud computing, semantic web, mobile services, which facilitates the design and development of new social web services. (Patsakis et al., 2014).

Nowadays, technology enables us to use the Internet as a best communication tool. Users can transfer secret data over the Internet as part of the proper communication, but we cannot ignore the danger of hackers and sniphers if the secret data are sent unshielded through the internet. (Niimi et al., 1999).

The problem of authentication of published information and who can see and share it and the risk of the unwanted malicious users represents a real privacy problem for the user, therefore OSNs try to manage users' privacy by using "privacy settings" for the services in many cases, but in fact new violations ranging from identity theft up to personal information exposure are disclosed daily with the ease of re-uploading and re-publishing a user's images, without informing the real owner. (Patsakis et al., 2014).

Ownership for those who use OSNs, should not be considered in terms of copyright, but rather it should be considered as a fundamental right of user privacy, some users think that by uploading their own photos on OSNs, they can allowing access only to the users that they want, in fact the uploaded images are part of their privacy and users should be able to selectively introduce themselves to others or not. (Patsakis et al., 2014).

Information could be provided with more security by using cryptography or steganography, Cryptography focuses on the encoding of the information, while the presence of information is obvious on the other side, and steganography aims to hide the presence of the information. (Por et al., 2008).

• Steganography

The goal of steganography is to hide secret text inside other files, usually multimedia. The secret message will be embedded inside the target file, in other words the sender writes a message and conceals a secret message on the same envelope. (Niimi et al., 1999).

In ancient ages they used tricks to hide their information, such as using invisible inks, tiny pin punctures on target characters, pencil marks on typewritten characters, grilles which cover most of the message except for a few characters, and so on. (BrahmaTeja et al., 2012).

Steganography is a term of Greek origin which means 'concealed writing' from the Greek words steganos (stegano´) which mean 'protected', and graphein (grawein) meaning 'to write', Nowadays, using modern computers the encrypted letter can be decrypted more easily than before, the embedded message which is hidden with steganography might be discovered and read it easily. (Kim et al., 2010)

Nowadays both cryptography and steganography are mostly related to electronic data (multimedia), while in the past steganography was presented using invisible ink that is still used rarely, while the message being invisible with a naked eye, using chemical materials to disclose color upon special lighting or chemical equation , but the limitation of using chemical invisible ink that is all kinds of invisible ink could be discovered with application of modern techniques (Kishimura et al., 2005).

• Hash Functions

Hash function is a special kind of one-way function which may be categorized into two types: un-keyed and keyed hash function, the un-keyed hash function may be also simply classified as a keyed hash function if it uses one secret key. (Xiao et al., 2005)

The MD5 algorithm was developed by Ronald Rivest at MIT University, by taking a message input of undefined lengths and producing a 128-bit hash code value, it has been one of the most widely-used hash algorithms, but it has been discovered that there is a security threat in the MD5 algorithm. (Walia et al., 2014)

• MD5 Hash Function

The hash function MD5, which have been designed by Ronald Rivest in 1992, is a strengthened version of MD4, although some weakness has been found by B. den Boer, A. Bosselaers and H. Dobbertin since its publication. MD5 was widely used in cryptography field, not too long ago it was used in such fields like digital signatures, data integrity, user authentication, key agreement, e -cash , protocols, and also in commercial security systems and products (Liang et al., 2007)

The MD5 algorithm is the next generation of the MD4 hash algorithm; its operation and performance is slower than MD4, but it offers more security, because MD4 focuses on speed rather security (Karamjeet et al., 2014).

• Secure Hash Algorithm (SHA)

The Secure Hash Algorithm (SHA) is called secure because it is theoretically impossible to find the encrypted message which it's origin to the parent message, or to find different messages which produce the same encrypted hash value ,and any tiny change to a message will produce a totally different hash value in return (Eastlake et al., 2001).

Among all the hash functions there is the SHA type, which shares the same functional structure with some other types in the internal operations, message block size, word size, message size, number of security bits and message hash size. They are all mostly used in the same way and use one-way functions that input a value and output a hashed value (Badillo et al., 2012).

• Rijndeal Algorithm

In 2000 the US National Institute of Standards and Technology selected the Rijndael algorithm as a new Advanced Encryption Standard (AES). Rijndael is a cipher that offers a combination of security, performance, efficiency, implementability and flexibility. It has already attained considerable popularity and acceptance.

All operations in Rijndael are defined in terms of arithmetic. Apart from Rijndael, there are several other instances of the use of Galois Field arithmetic in cryptography and coding theory. (Daemen et al.,2002)

• Least Significant Bit

The simplest method that is used in steganography is the 'least significant bit' (LSB) technique, or LSB embedding. The image consists of a matrix of pixels and each pixel is usually represented by 3 bytes with each byte of an image representing a color. The bits on the right do not represent too much of hue the first, "therefore, two bytes that only differ in the last few bits can represent two colors that are virtually indistinguishable to the human eye". For example, 00100100 and 00100101 are theoretically two different color, and by changing the last bit it is very hard for the human eye to notice any change in the color. LSB embedding alters these last bits by hiding a message within them as shown in figure 1. (Liu et al., 2008)

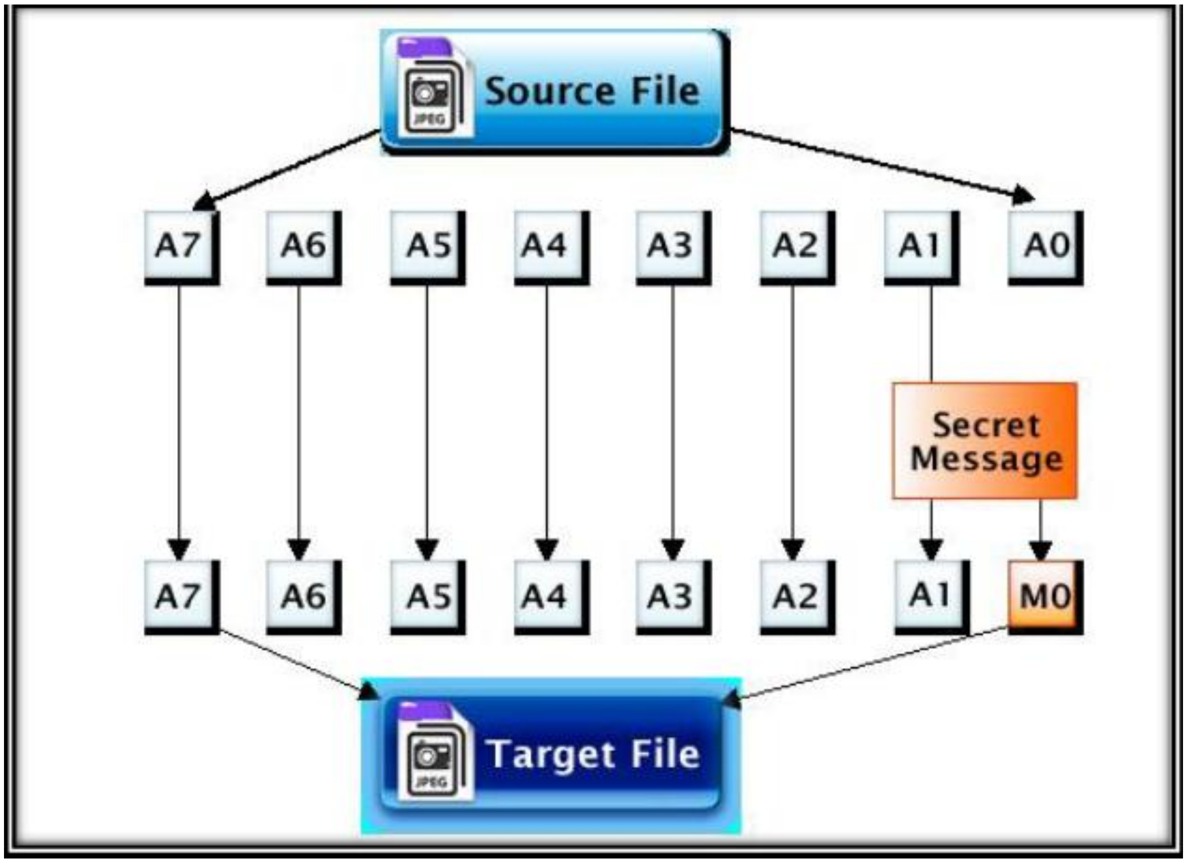


Figure 1: Represent of Least Significant Bit

Least significant bit (LSB) insertion is a common, simple approach to embed information in a cover image on the image domain. The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. When using a 24-bit image, a bit of each of the red, green and blue (RGB) color components can be used, since they 8
are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 × 600 pixel image, can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data. For example a grid for 3 pixels of a 24-bit image can be as follows: (Falih et al., 2013)

(00101101   00011100   11011100)

(10100110   11000100   00001100)

(11010010   10101101   01100011)

1.2 ***Problem Statement:***

Many people who use the OSN are photographers or designers that consider the (OSN) as a place where they can share their fantasy without any consideration of the owner's privacy. Some people try to misuse, download and re-upload images of others, finally these images are shared around the (OSN) without the real owner being identified.

This research will propose 2 different new methods to protect the images' privacy for users across (OSN) using steganography to store a unique signature encrypted by hash algorithm or Rijndael algorithm, taking into consideration that hash algorithm or Rijndael algorithm is required to encrypt the user's signature and then embed that signature inside the image.

1.3 Objectives

The main objective of this research is to create two different new methods that use encrypted signatures using hash algorithm or Rijndael algorithm, after embedding the signature inside the image by using (LSB). This technique will detect and recognize the privacy of the uploaded image, embed the signature inside an image, and detect and recognize if any uploaded image has privacy or not. The technique will reject uploading the image if it has an assigned privacy.

1.4 Questions

1. Can the proposed technique provide more privacy for images users in (OSN) s?
2. What is the effect of using hash algorithm in this technique?
3. What is the effect of using Rijndeal algorithm in this technique?
4. Will the proposed technique prevent sniffers from re-uploading the protected images?

1.5 Problem Motivation

The research addresses the images privacy in (OSN) using steganography technique, since steganography is an old technique that is sometimes used to protect privacy in multimedia by embedding a signature to verify the ownership.

Until now research has shown that there are no adopted techniques by (OSN)s that use both steganography and encrypted signature to protect the image privacy, so images will be the main concern of this research.

The new technique aspires to minimize users' privacy violation through automated procedures ,the novelty of the proposed scheme is the ability to enforce a user's privacy policies across the (OSN) without using a third party even if the technique is not applied to all (OSN) s.

CHAPTER TWO

Literature Review

This section describes some previous works in privacy in (ONS) s. Many techniques and studies have been produced to improve the protection of images privacy

The first related work, by Boyd et al. (2013), describes the architecture and the relationship in (OSN)s. This architecture is depicted as a line graph among users in Figure 2, which shows the relationships among users (A, B, C. D, E. F. G, and H) in an OSN architecture. The relationships between users are direct connections among users in the graph, (A, B). (B. C), and (G. D). Non-friend relationships are shown as indirect connections to a user in the graph. Thus, non-friend relationships are connected through a friend. (B. E: B—A—E) and (D. C: D—B—C)

The research shows that it is possible to express the overall (OSN) architecture via users as a single chart, and it is also possible to analyze the relationships between users based on the distance of separation in the object.

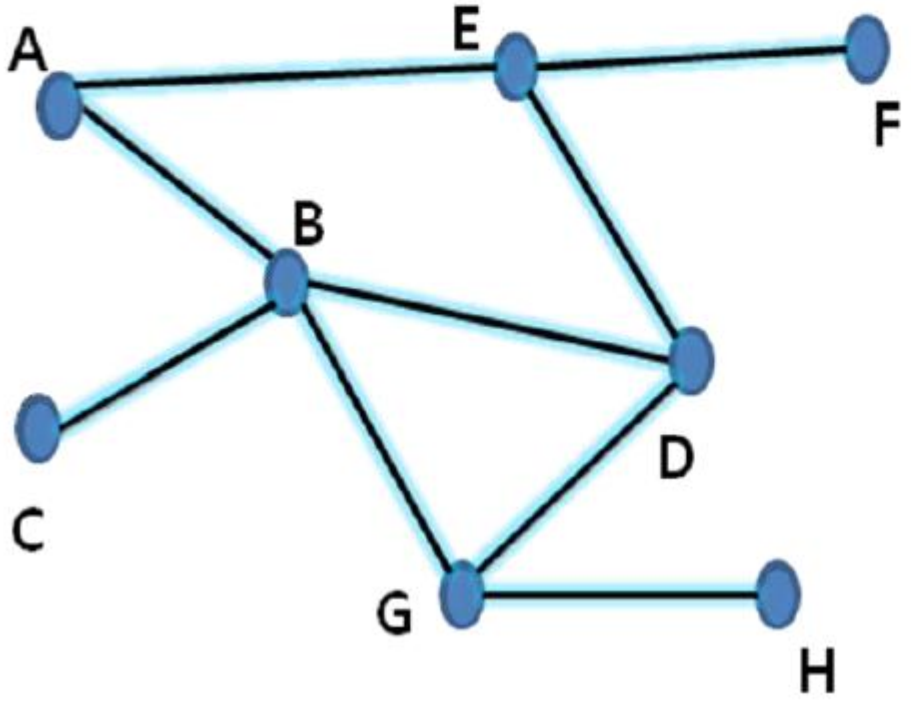


Figure 2: description of relationships in (OSN)

Patsakis et al. (2014) suggest an automated procedure that guarantees the copyright of multimedia for users across multiple (OSNs). (OSN) will not interact with any solution to protect the copyright of users' multimedia because, like any other company its' focus is to have a bigger share in the market and rules and constraints will encourage customers switch to other companies. The idea of redesigning all the (OSN) from scratch is bad idea, but all (OSN) should cooperate with each other to avoid the sniphing of multimedia and protect the copy right of their users. (OSN) should avoid using external players like governmental rules. The solution is to invent an option for the users that can be used to protect their multimedia that they share on (OSN), for example a technique which uses a watermark with encryption algorithm on the multimedia. This watermark would later be published on other (OSN) so that no one can re-upload that content on any (OSN). While Shilbayeh et al., (2014) propose a new secure architecture to find an effective solution to reduce fake pages and possibility of recognizing VIP pages on (OSN)s by the logo method which appears inside the profile photo.

This is limited to serve only the VIPs which have an effective website, apply this on Facebook, which are the most famous social-networking sites and also flexible to use a third party. Service on the FaceTrust will reports the number of fans who joined to VIPs pages that use the FaceTrust application, but the researcher suggests to depend on a third party. Figure 3 represents the main processes of FaceTrust.

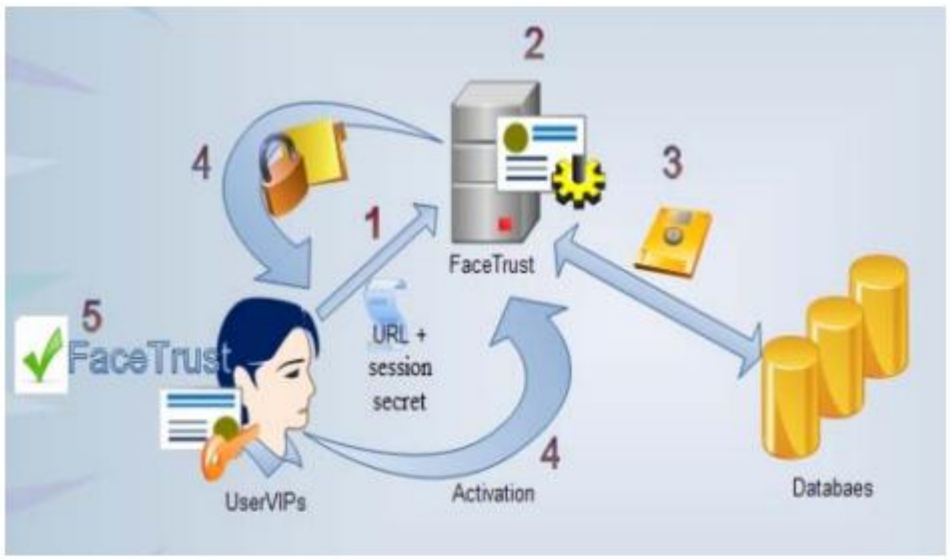


Figure 3: the main process of Facetrust architecture

Jang et al., (2013) proposed a secure tickets generator using a secure hash function. Both the Information Creator and the Information Holder validate a ticket. The users with legible tickets are allowed to access information. The Information Holders and Information Creator on both sides of the ticket validation process were configured to control access to different Information Creators and Information Holders. This prevented information leakage via authorized users. This thesis will not suggest the use of an embedded signature for any information

Figure 4 represents the process of information access.

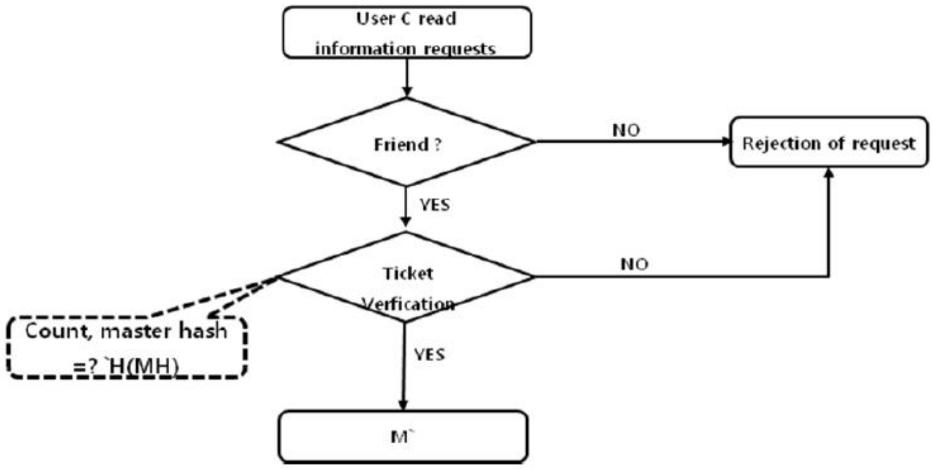

User C read information requests
Friend ?
NO
Rejection of request
YES
Ticket Verfication
NO
Count, master hash =? `H(MH)
YES
M`

Figure 4: the process of information access

Yuan et al, (2014) implement a privacy-preserving image-centric social discovery system to expand user’s friends with common interests securely, by deploying a system which uses the ‘cloud’ as an image storage back end, the system consists of secure and compact index to enable fast and scalable searches over millions of user image profiles, the evaluation demonstrates that the system is practical and efficient under a huge image dataset with 1 million users.

Figure 5 represents the proposed system architecture, but this implementation will prevent others from previewing other images and used a database and index.

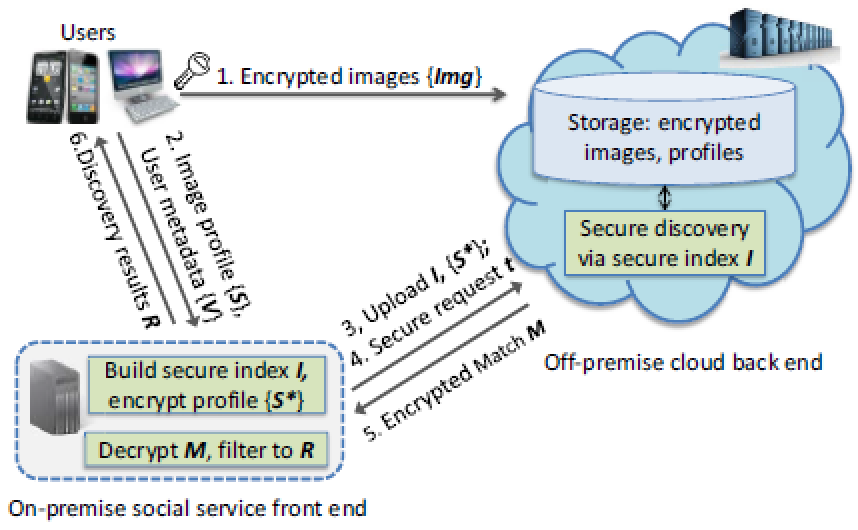

Users
1. Encrypted images {*Img*}
Storage: encrypted images, profiles
Secure discovery via secure index *I*
2. Image profile {*S*}, User metadata {*V*}
6.Discovery results *R*
3, Upload *I*, {*S**};
4. Secure request *t*
5. Encrypted Match *M*
Off-premise cloud back end
Build secure index *I*, encrypt profile {*S**}
Decrypt *M*, filter to *R*
On-premise social service front end

Figure 5: the Proposed System Architecture for Yuan et al, (2014)

Luo et al (2009) Developed FaceClock, which is a solution to provide more protection for users privacy in (OSN) s, it should be shielded from the social networking site. By allowing to the user to be selective of which information he wants to safeguard and which to leave as it was, based on his own judgment of the value of privacy, the implementation can be general enough to be applied to other (OSN)s. Using digital rights managements (DRM) in OSN, figure 6 represents the architecture of FaceClock.

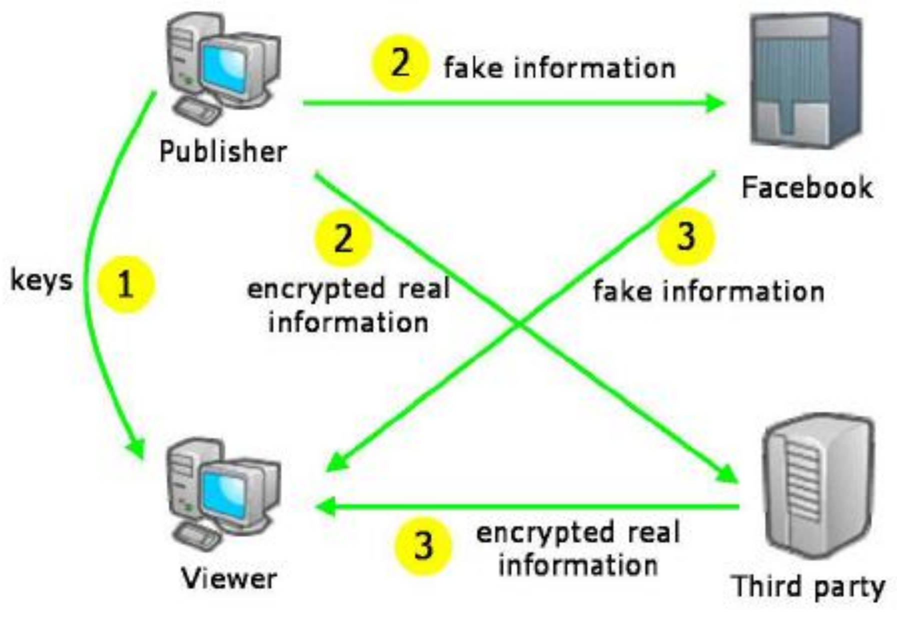
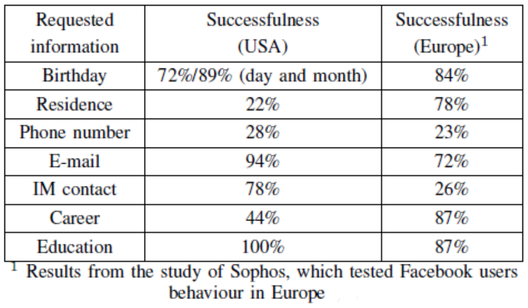


Figure 6 :FaceClock Architicture

Qin-long et al., (2014) said that the best solution to provide more secure is to use cipher text policy attribute-based encryption (CP-ABE) instead of the regular cryptography. Considering that the security techniques of (OSN) have no credibility, (OSN) should be only a mediator and a third party should handle the security and privacy.

The object of Nagy et al., (2009) work is to analyze social networks as a modern communication medium that can be misused by technique of social engineering. By taking a sample of people from Europe and America, one of the conclusions from the sample is that there is no coherence between the amount of information provided in the profile account and number of friends to the success of gain new contacts. There is also another finding which shows that it is easier to gain contacts with women than to make contact with men in the (OSN). But this finding is not a general conclusion as they tested only a small sample. The solution of privacy and security for the user by increase the awareness of using the (OSN) what the user can share and what should not share. Figure 7 show the percentage of the revealed data among the sample.



Requested information	Successfulness (USA)	Successfulness (Europe)[1]
Birthday	72%/89% (day and month)	84%
Residence	22%	78%
Phone number	28%	23%
E-mail	94%	72%
IM contact	78%	26%
Career	44%	87%
Education	100%	87%

[1] Results from the study of Sophos, which tested Facebook users behaviour in Europe

Figure 7: the Percentage of Access Information for Nagy Sample

The researchers R. Gross et. al., (2005) present a survey of more than 4,000 Carnegie Mellon University Facebook users, the researchers have quantified individuals' willingness to provide large amounts of personal information in an online social network, and show how unconcerned social network users appear in regards to privacy risks: "while personal data is generously provided, limiting privacy preferences are hardly used; only a small number of members change the default privacy preferences, which are set to maximize the visibility of users pro- files."

According to the researcher, the information that users have provide online exposes users to various physical and cyber risks, and makes it extremely easy for third parties to create overall reports of their behavior.

CHAPTER THREE

Methodology

This thesis introduces a new technique to protect the privacy of images in online social networks by using signature encryption and LSB, The new technique will use a user signature which will be encrypted by either (SHA 2 algorithm or AES algorithm). This encrypted signature value will be distributed in the target image according to the uploading time if the owner wants to apply privacy on his uploaded images.

In case of any re-uploading by others, the technique will stop and prevent the operation of re-uploading.

The research will present two methods

1. Using Hash algorithm (SHA2 256 bit) which is used in Facebook and Snapchat
2. Using Rijndeal algorithm (AES 256-CBC bit) which is used in Hotmail

Figure 8 represent the flow chart of the used technique

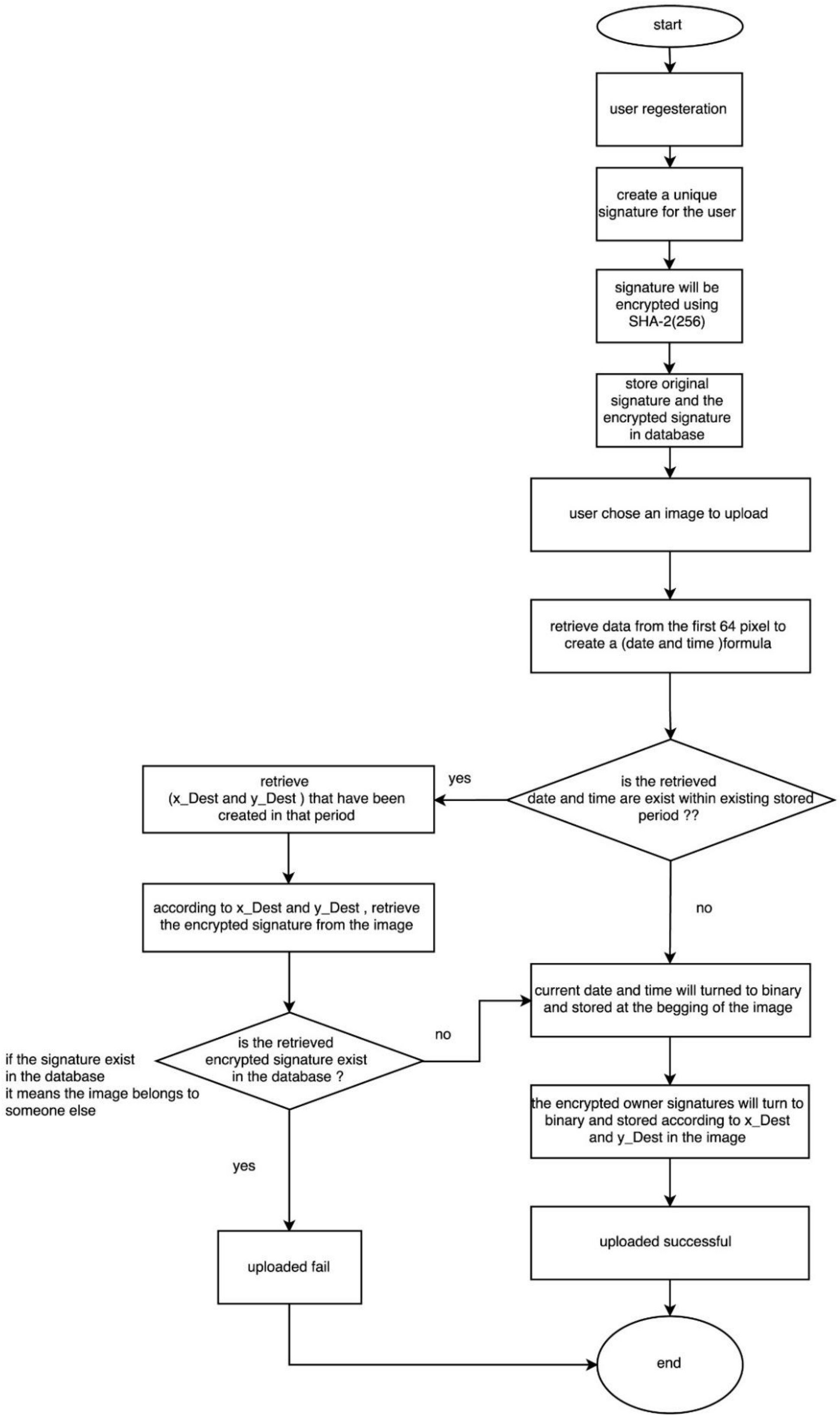
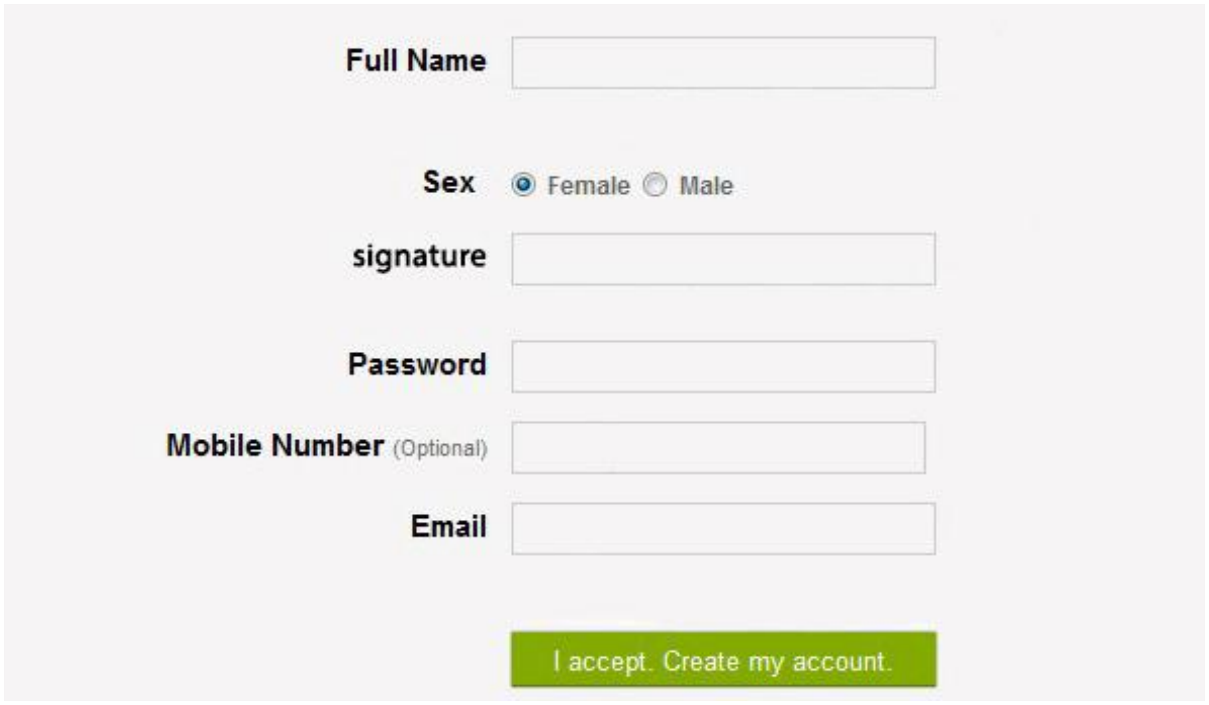

start
user regesteration
create a unique signature for the user
signature will be encrypted using SHA-2(256)
store original signature and the encrypted signature in database
user chose an image to upload
retrieve data from the first 64 pixel to create a (date and time)formula
is the retrieved date and time are exist within existing stored period ??
yes
retrieve (x_Dest and y_Dest) that have been created in that period
according to x_Dest and y_Dest , retrieve the encrypted signature from the image
is the retrieved encrypted signature exist in the database ?
if the signature exist in the database it means the image belongs to someone else
yes
uploaded fail
no
no
current date and time will turned to binary and stored at the begging of the image
the encrypted owner signatures will turn to binary and stored according to x_Dest and y_Dest in the image
uploaded successful
end

Figure 8: Flow Chart of the Used Technique

The main steps of the new technique are demonstrated below:

3.1- User Registration

Like any (OSN), registration is compulsory, the user will provide the (OSN) with his personal information in addition to a unique signature chosen by the user, as shown in figure 9



Full Name	
Sex	[● Female] ○ Male
signature	
Password	
Mobile Number (Optional)	
Email	

Figure 9: Registration Interface

3.2- Signature Creation

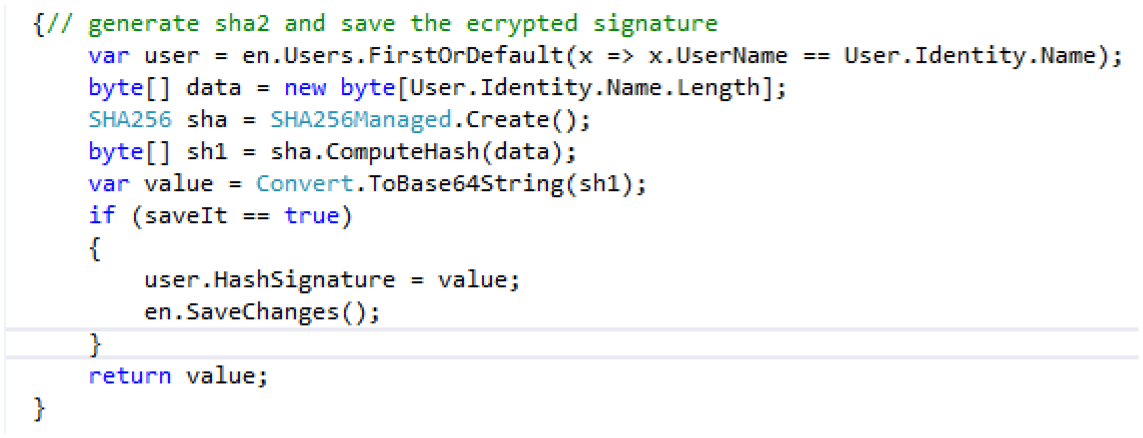
After registration, the user signature will be encrypted using either SHA-2(256) algorithm or AES (256-CBC) algorithm, the signature, which will be embedded into the target image, will be used to apply privacy on the image.

3.2.1- Using SHA-2 (256)

The signature will be encrypted using SHA-2(256 bit) algorithm and a fixed-length of 64 hexadecimal letters will be created.

Example: encrypt user signature where the signature is “hebah nasereddin”
595ad0ae2af0bfd0533aad9e48f5c1d7c7f71d951b800c5d8f3410bf0ab31b35

Both (original signature) and (fixed-length encrypted signature) will be stored in the database, figure 10 represent the process of signature creation using SHA256.



```
{// generate sha2 and save the ecrypted signature
    var user = en.Users.FirstOrDefault(x => x.UserName == User.Identity.Name);
    byte[] data = new byte[User.Identity.Name.Length];
    SHA256 sha = SHA256Managed.Create();
    byte[] sh1 = sha.ComputeHash(data);
    var value = Convert.ToBase64String(sh1);
    if (saveIt == true)
    {
        user.HashSignature = value;
        en.SaveChanges();
    }
    return value;
}
```

Figure 10 :Code for Signature Creation Using SHA2

3.2.2- Using (AES 256 CBC)

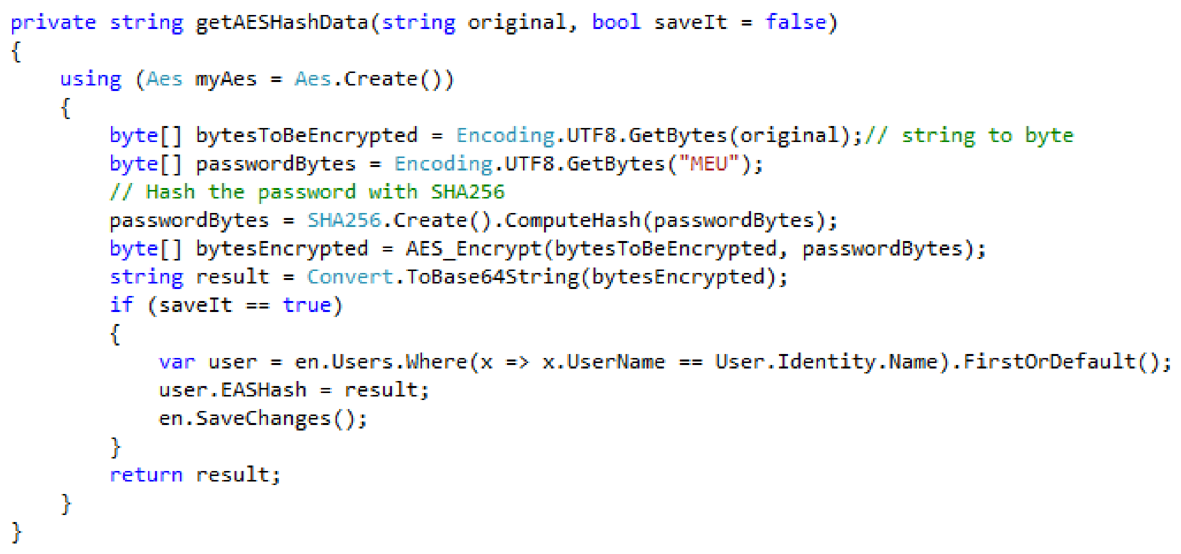
After registration, the user signature will be encrypted using Rijndeal (AES-256-CBC) algorithm, a fixed-length of 64 letter will be created.

Example: encrypt user signature where the signature is “imad” and the AES key is “Khuffash”

The result is : KPektxOSpIfFkCSdxv7hVztnu+usG6lXyDq1CKd4e4I=

The AES key was chosen to be “MEU” which will be encrypted using SHA256 to generate encrypted value which will be used to encrypt and decrypt the owner signature, then, the encrypted signature will be embedded into the image.

Figure 11 show the process of creating signature using AES encryption algorithm .



```
private string getAESHashData(string original, bool saveIt = false)
{
    using (Aes myAes = Aes.Create())
    {
        byte[] bytesToBeEncrypted = Encoding.UTF8.GetBytes(original);// string to byte
        byte[] passwordBytes = Encoding.UTF8.GetBytes("MEU");
        // Hash the password with SHA256
        passwordBytes = SHA256.Create().ComputeHash(passwordBytes);
        byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, passwordBytes);
        string result = Convert.ToBase64String(bytesEncrypted);
        if (saveIt == true)
        {
            var user = en.Users.Where(x => x.UserName == User.Identity.Name).FirstOrDefault();
            user.EASHash = result;
            en.SaveChanges();
        }
        return result;
    }
}
```

Figure 11:Code for Signature Creation Using AES

3.3- Image Uploading and Date Embedding

Like any (OSN), the user can upload any image on his profile.
Date, time and encrypted signature are important inputs for image uploading, which will be discussed below.

Whilst uploading, the image will be split into series of pixels (x columns and y rows), and the date, time and encrypted signature will be embedded into the image.

- ***Date and Time Embedding for Both Methods:***

The uploading date and time (of the image) will be transferred into binary formula.

Example: if the user uploads the image at 15/10/24; 23:17:03, the date and time will be transferred as follows
15/10/24; 23:17:03=> 00001111 00001010 00011000 00010111 00010001 00000011

The technique will reserve the first row of the image to store the uploading time and date (of the image) using (LSB) which will be discuss later in details in 3.5.1 .
After embedding date and time The new technique will use user signature which has been encrypted by either (SHA 2 algorithm which will be discussed 3.6.1 or Rijndeal algorithm which will be discussed in section 3.6.2).

3.4- Signature Embedding

The technique will embed the (the encrypted signature) which have been encrypted by either (SHA algorithm or AES algorithm)

The encrypted signature, will be stored into the image, each value will be stored in separated line according to (x_Dest and y_Dest).

(x_Dest) determine the first used pixel in storing hash value as shown in figure 12.

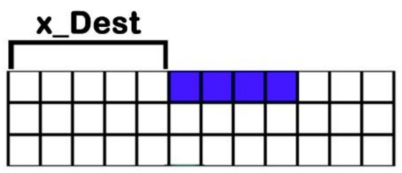


Figure 12: Using x_Dest to Determine the Destination in x-axis.

While (y_Dest) determine the number of the unused rows between each two used rows (unused rows have no embedding values). As shown in Figure 13.

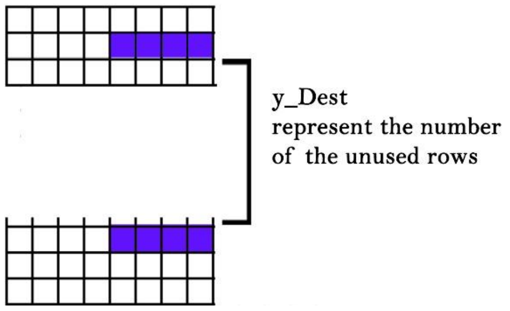
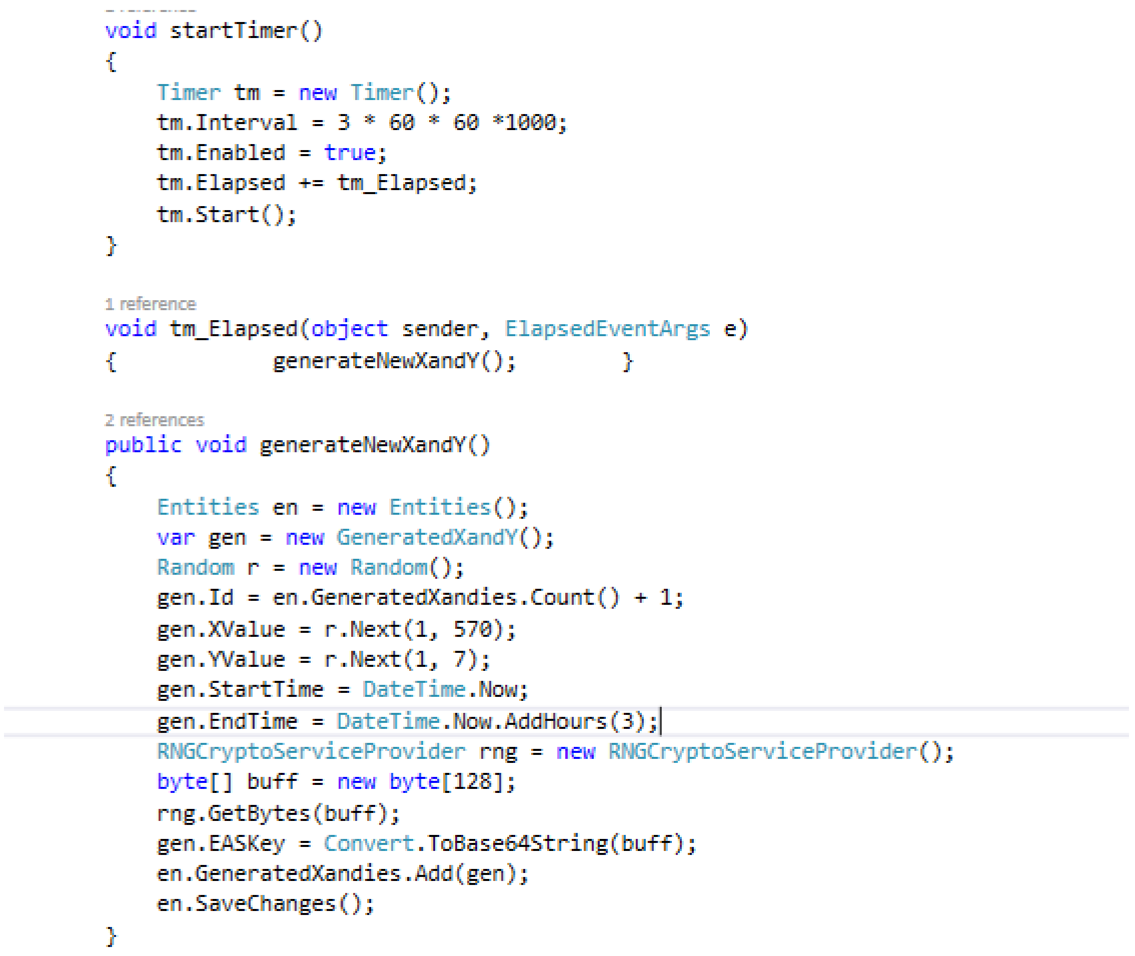


Figure 13: Using y_Dest to Determine the Unused Rows Between Each 2-Used Rows.

Both (x_Dest and y_Dest) will be stored in the database.
Figure 14 represent the used code to generate x_Dest and y_Dest



```
void startTimer()
{
    Timer tm = new Timer();
    tm.Interval = 3 * 60 * 60 *1000;
    tm.Enabled = true;
    tm.Elapsed += tm_Elapsed;
    tm.Start();
}

1 reference
void tm_Elapsed(object sender, ElapsedEventArgs e)
{            generateNewXandY();        }

2 references
public void generateNewXandY()
{
    Entities en = new Entities();
    var gen = new GeneratedXandY();
    Random r = new Random();
    gen.Id = en.GeneratedXandies.Count() + 1;
    gen.XValue = r.Next(1, 570);
    gen.YValue = r.Next(1, 7);
    gen.StartTime = DateTime.Now;
    gen.EndTime = DateTime.Now.AddHours(3);
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
    byte[] buff = new byte[128];
    rng.GetBytes(buff);
    gen.EASKey = Convert.ToBase64String(buff);
    en.GeneratedXandies.Add(gen);
    en.SaveChanges();
}
```

Figure 14:Code for Generating x_Dest and y_Dest

3.5- Others Technique for More Security

The technique will use extra techniques to improve security of the embedded data

3.5.1 Embedding Into Pixels

For both (date and time embedding) and (encrypted signature embedding), the technique will use only 1-bit from (blue) channel for each chosen pixel, according to the following:

- If the sum of (location x and location y) of the pixel, is odd, the third (LSB) will be used.
- If the sum of (location x and location y) of the pixel, is even, the second (LSB) will be used

Figure 15 show an example of the used technique ,while figure 16 show the code the used technique .

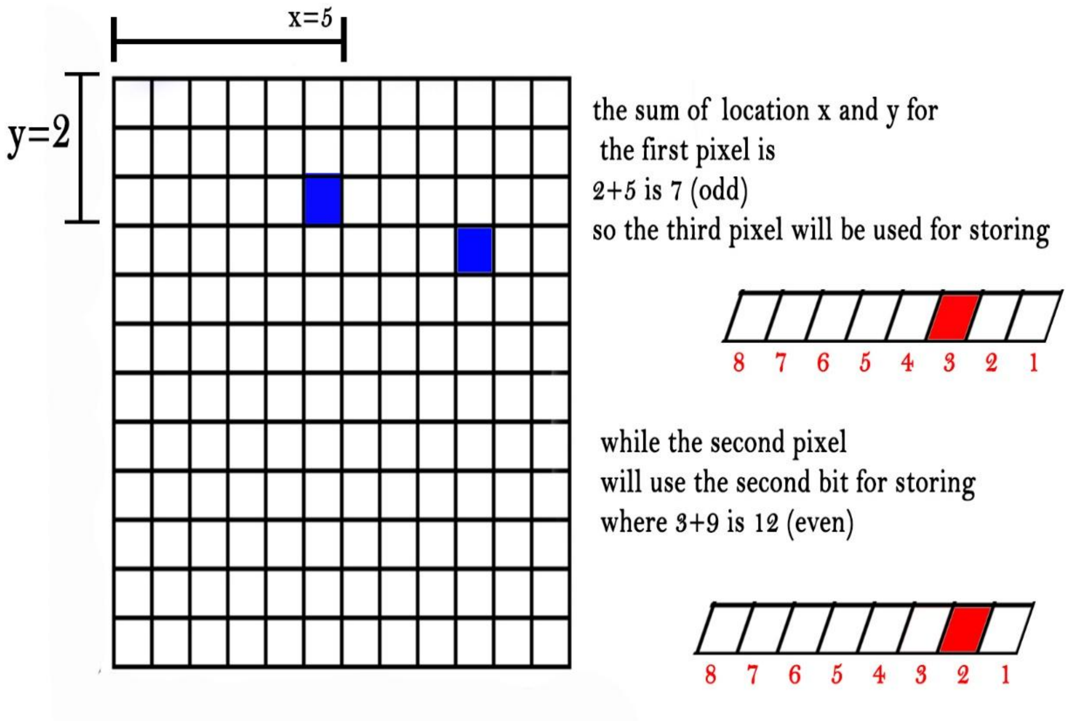
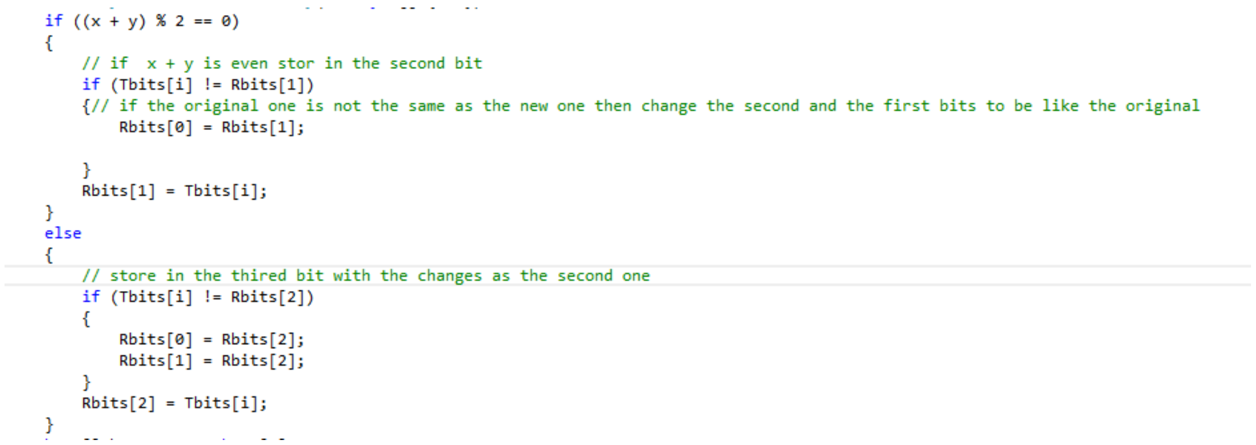


Figure 15 : Pixel Embedding Technique



```
if ((x + y) % 2 == 0)
{
    // if  x + y is even stor in the second bit
    if (Tbits[i] != Rbits[1])
    {// if the original one is not the same as the new one then change the second and the first bits to be like the original
        Rbits[0] = Rbits[1];

    }
    Rbits[1] = Tbits[i];
}
else
{
    // store in the thired bit with the changes as the second one
    if (Tbits[i] != Rbits[2])
    {
        Rbits[0] = Rbits[2];
        Rbits[1] = Rbits[2];
    }
    Rbits[2] = Tbits[i];
}
```

Figure 16:Code for Embedding Into the 2nd and 3rd (LSB)

3.5.2 (x_Dest and y_Dest) Random Generation

Every fixed amount of time (the technique used 3-hours interval, to reduce the load on server, this period could be minimized or maximized according to the server storage and performance) a new random (x_Dest) and (y_Dest) values will be generated , x_Dest , y_Dest and generation time will be store in database as shown in table 1

Table 1: x_Dest , y_Dest and Generation

genTime	**x_Dest**	**y_Dest**
2015/9/28 15:25:00	54	5
2015/9/28 18:25:00	23	2
2015/9/28 21:25:00	276	7

While uploading, the technique will use the latest generated (x_Dest and y_Dest) values, these values will determine the destination of x and y for embedding data as described before.

3.5.3 Reduce Noise Technique

This technique will try to reduce the possible noise of embedding using steganography techniques.

Tables below show the reduction of noise after using “reduce noise technique”.

In case of placing 0 with 1 as shown in table 2

Table 2: Cases of Placing 0 with 1

Original	**After placing (Without modifying)**	**Noise degree**	**After placing (modifying)**	**Noise degree (modified)**
000	**1**00	4	1**00**	4
001	**1**01	4	1**00**	3
010	**1**10	4	1**00**	2
011	**1**11	4	1**00**	1

Table 3: Cases of Placing 1 with 0

Original	**After placing (Without modifying)**	**Noise degree**	**After placing (modifying)**	**Noise degree (modified)**
100	**0**00	4	0**11**	1
101	**0**01	4	0**11**	2
110	**0**10	4	0**11**	3
111	**0**11	4	0**11**	4

3.6 in case of re-uploading the embedding image (protected image)

If others try to re-upload the protected image, the new technique will do the following:

While re-uploading , the technique will retrieve the first 64 pixels in the image, which represent the uploading date and time , a date format will be retrieved as follows (year / month / day , hour: minute: second) to use it to check the retrieved date and time from the database.

3.6.1 Hash Algorithm Method :

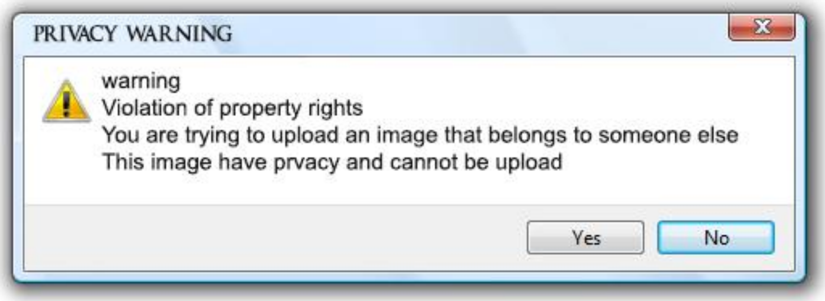
If the retrieved date and time exist within any existing stored period, then (x_Dest and y_Dest) will be retrieved.

By retrieving x_Dest and y_Dest, the map destination of the stored (encrypted signature) will be retrieved to create the hashed signature.

If the retrieved encrypted signature exists in the database, the upload will fail, unless the signature belongs to the owner.

Table 5: Retrieve x_Dest , y_Dest According to Generation Time

genTime	x_Dest	y_Dest
2015/9/28 15:25:00	54	5
2015/9/28 18:25:00	23	2
2015/9/28 21:25:00	276	7


PRIVACY WARNING

warning
Violation of property rights
You are trying to upload an image that belongs to someone else
This image have prvacy and cannot be upload

Figure 17 : (Privacy-Warning) Window in Case of Property Violation

3.6.2 Rijndeal Algorithm Method:

If the retrieved date and time are exist within any existing stored period, then (x_Dest , y_Dest and AES key) will be retrieved.

By retrieving x_Dest , y_Dest , the map destination of the stored (encrypted signature) will be retrieved to create the encrypted signature.

The AES key will be used to decrypt the retrieved signature to produce the original owner signature

CHAPTER FOUR

Experimental Results

This chapter will discuss the result of the new technique , a tool have been developed using visual C# 2013 , it will be used to evaluate the performance of the proposed technique on images, and the effect of embedding process on the secret image , the research is performed on 3 different images with different sizes .

4.1 Image Quality Test

The image quality test is a measure to test the quality of the images compared to the original images. Several images have been used to measure the performance of the new technique, this test has used DiffImg tool version 2.0.1 to compare both the original and the signed image.

Here are some statistical differences between embedding signatures that have been encrypted using either (hash algorithm or Rijendeal algorithm). This data will expose the standard deviation (SD) of Peak Signal to Noise Ratio (PSNR) and Root Mean Squared Error (RMSE) .

4.1.1 Standard Deviation (SD)

Standard deviation is a statistical equation used to represent the variation or dispersion in a set of values from the average. The standard deviation have been measured for images after signature embedding, as described in the tables below.

Standard deviation is calculated according to the following equation

$$SD= \sqrt{{}^{1}/_{n} \sum_{i=1}^{n} (X - X')^2}$$

Where:

N: the size of the sample.

X: the observed values of the sample items.

X`: the mean value of these observations.

4.1.2 Root Mean Squared Error (RMSE)

The root mean square error is an equation which is usually used to measure the differences between variables that can be predicted, in our case the RMSE variables are discussed for signature embedding into an image.

4.1.3 Peak Signal to Noise Ratio (PSNR)

PSNR is used to measure the quality metric, the PSNR here is being reviewed for embedding a signature.

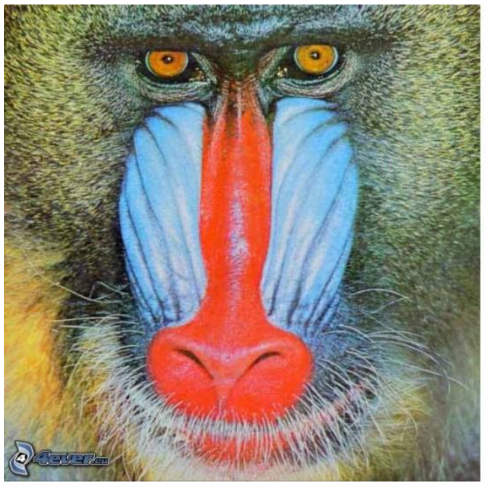
$$PSNR=20.\log_{10} \frac{255}{RMSE}$$

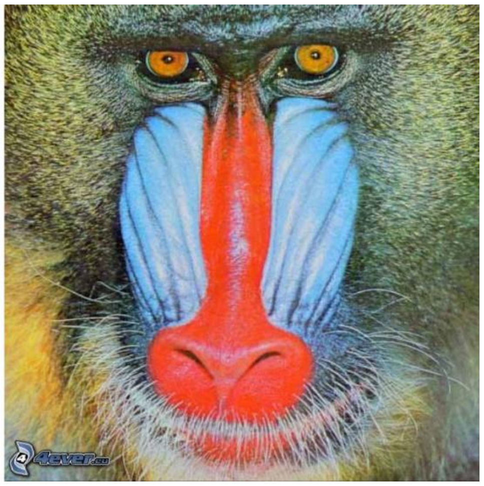
4.2 Quality Test on the Signed Images

The study is performed on 3 different images with different sizes , each image have been used but the new technique

The following tables show the effect of the signed images after being embedding , the change of quality have been measured using 3 statistical equations , Standard deviation (SD), Root Mean square error (RMSE) and Peak Signal to Noise Ratio (PSNR)

The following figures are sample images before signed and after

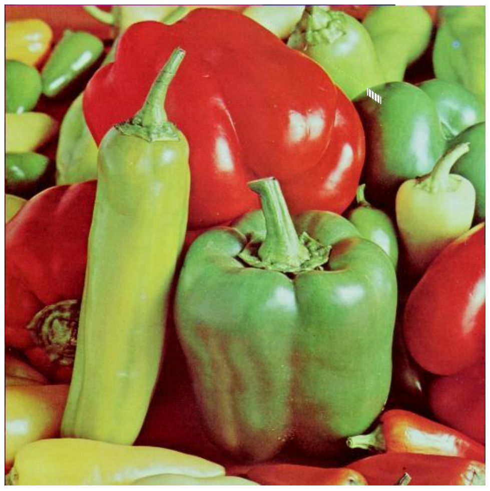




a) Baboon Before Embedding

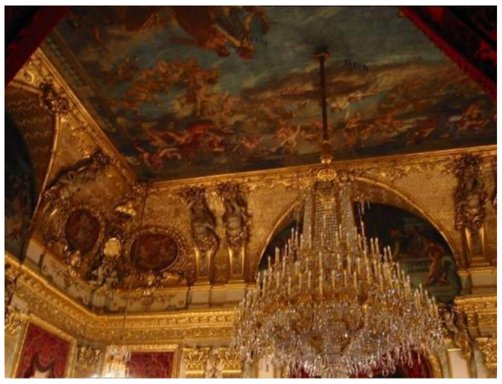
b) Baboon After Embedding

Figure 18 :Baboon Image Before and After Embedding



a) Peppers Before Embedding b) Peppers After Embedding

Figure 19: Peppers image before and after embedding



a) Ceiling Before Embedding b) Ceiling After Embedding

Figure 20 :Ceiling Image Before and After Embedding

The lack of quality after embedding will be relatively low when using the proposed technique

4.2.1 Standard Deviation

Standard deviation is an equation to describe the variation, as shown below in the tables, the standard deviation have been measured for images after being signed

Table 5: standard deviation of the signed images

	Using SHA method	Using AES method
Ceiling	0.00639	0.00756
Baboon	0.00111	0.00126
pappers	0.00764	0.00981

The results of standard deviation are very small which means that images are very similar, however, the Standard Deviation (SD) for the new technique was too low which provided a better quality after embedding the signature into the target image.

4.2.2 Root Mean Squared Error

The root mean square error is an equation which is usually used to measure the differences between variables that can be predicted, in our case the RMSE variables are discussed for signature embedding into an image, in this section the (RMSE) have been discussed for the signed images.

(Wang, et al., 2003).

Table 6 : the RMSE Results

	Using SHA method	Using AES method
Ceiling	0.00640	0.00760
Baboon	0.00111	0.00137
Pappers	0.00670	0.00789

The values of (RMSE) are preferred to be low which indicate that images are more similar, the RMSE with the new technique was too low which provide better quality.

4.2.3 Peak Signal to Noise Ratio:

Peak signal to Noise Ratio is used to measure the quality metrics that describe the quality of the image (Wang, et al., 2003).

Table below show the PSNR values of the signed image after embedding the encrypted signature, in addition to comparison with kaur technique (kaur, et al. 2013).

Table 7 the PSNR Results

	Using SHA method	Using AES method	Using Kaur technique
Lena	76.21	75.12	45.3672
Baboon	60.07	59.93	42.9277
Pappers	61.85	60.64	42.4288
Ceiling	65.65	60.54	

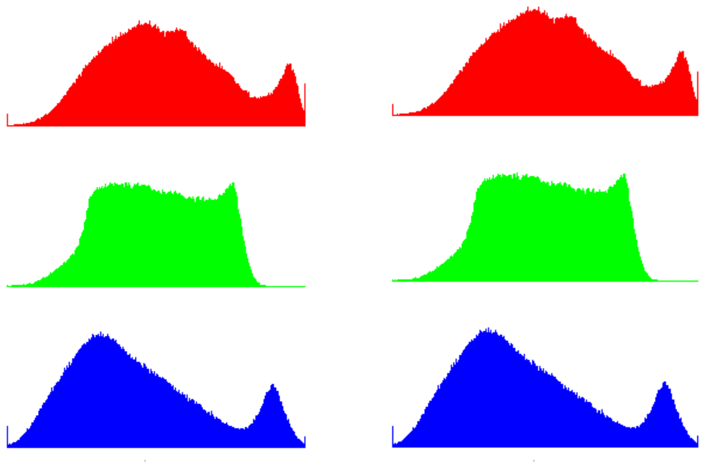
Peak Signal to Noise Ratio (PSNR) values are preferred to be higher which indicates that image quality is better, however, the PSNR for both methods were high

4.3 Histograms for Proposed Method

The following histograms represent the distribution difference between stegoimages before and after the embedding of the encrypted signature.

Figure 21 shows that there is no visual effect on the values of RGB channels for the

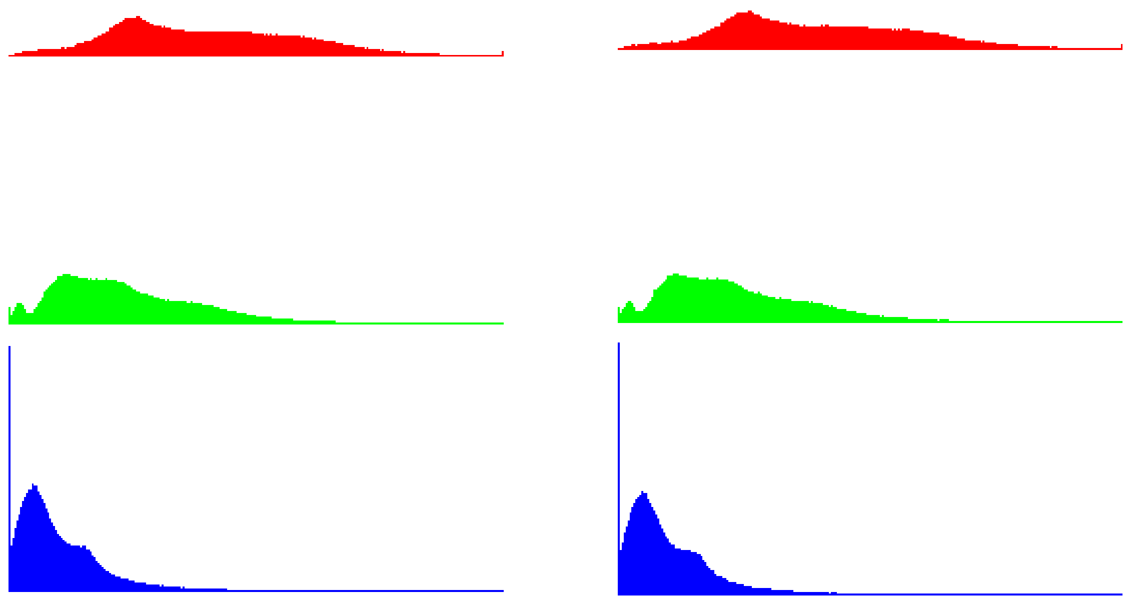
Baboon im



b) Baboon Before Embedding b) Baboon After Embedding

Figure 21 : Histogram for Baboon Signed Image Using the Proposed Method.

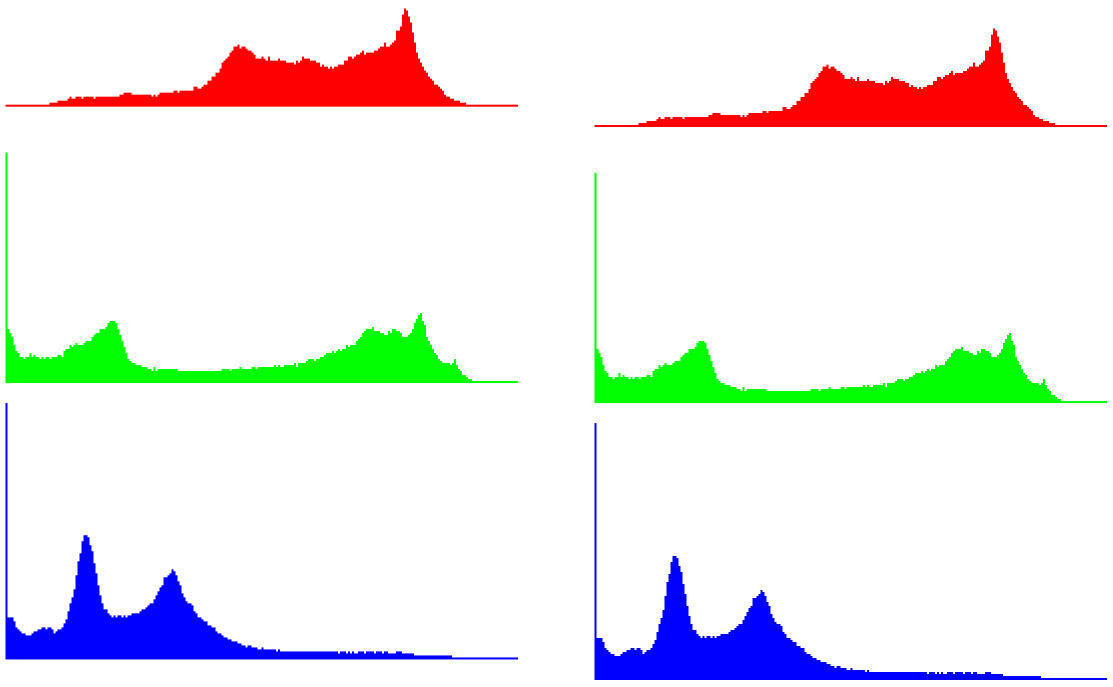
Figure 22 shows that there is no visual effect on the values of RGB channels for the ceiling image



a)Ceiling Before Embedding b) Ceiling After Embedding

Figure 22: Histogram for Ceiling Signed Image Using the Proposed Method.

Figure 23 shows that there is no visual effect on the values of RGB channels for the ceiling image



a) Peppers Before Embedding b) Peppers After Embedding

Figure 23 : Histogram for Peppers Signed Image Using the Proposed Method

4.4 Performance Test

The performance test of the technique will represent the measures of the needed time for operations, as mentioned before , the technique will create a unique encrypted signature using either (SHA algorithm or Rijendal algorithm) and embed that signature into the image , the overall operation will surely consume more time .

This section will represent the statistical values (time differences) between the regular upload and the proposed technique

Table 8 : Uploading Time Results

	Using SHA upload	AES method upload	Regular upload
ceiling	9.956	10.320	1.442
Baboon	8.91	8.95	1.39
pappers	9.59	11.86	2.14

CHAPTER FIVE

Conclusion and Future Work

5.1 Overview:

This chapter summarizes the conclusion of this thesis, the main objective of the thesis was to focus on the ownership of images in online social network, several studies were conducted to provide more ownership opportunities of images in social networks, Thus, this chapter was organized as section 5.2 to discuss the main conclusions, and section 5.3 to preview the future research works in the field of ownership of images in social network

5.2 Conclusion

In this thesis , a new technique have been proposed which suggests that the images in (OSN) are protected by embedding unique signature , that signature will be encrypted using either SHA-2(256) algorithm or AES (256-CBC) algorithm ,In case of any try to re-upload the protected image by others, the technique will prevent up-loading.

To increase PSNR value, the technique assigned some enhancement on the embedding technique, the enhancement will try to reduce the possible noise that caused by signature embedding.

The results of the proposed methods (AES and SHA) have been compared, the results indicates that the noise caused by embedding the encrypted signature is too small, this can be viewed in higher PSNR values and lower RMSE, this observation is caused by the fact that lower significant of bits are needed to be replaced in order to hide the encrypted signature.

5.3 Future work

In this thesis a new technique have been proposed for embedding encrypted signature inside image, this method can be the base for which other studies can emerge, the following list present some ideas for further studies:

1- Using different type of encryption algorithm
2- Using different embedding techniques
3- Apply the technique on different multimedia in social network(video or audio)

References

- Abdullah, Y., and Nassereddin, H., (2013). “Proposed Data Hiding Technique – Text under Text” ***American Academic and Scholarly Research Journal***, 243- 248 .

- Ahmad, F., and Nassereddin, H. (2013). “New method color image inside image steganography”.

- Algredo-Badillo, I., Feregrino-Uribe, C., Morales-Sandoval, M., and Cumplido, R. (2012). “FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256”. ***Microprocessors and Microsystems*.**

- Boyd, D., and Ellison, N. (2007). “Social Network Sites: Definition, History, and Scholarship”**. *Journal of Computer-Mediated Communication***, 210-230.

- Kaur, S. and Jindal, S. (2013). “Image Steganography using Hybrid Edge Detection and First Component Alteration Technique”. ***International Journal of Hybrid Information Technology.***

- Khufash, I., and Nassereddin, H. (2015) “New Technique to Protect the Privacy of Images in Social Network by Using Hash Algorithm and Least Significant Bit” ***international journal of advanced research in computer science and software engineering (ijarcsse)*** 672-677

- Nassereddin, H., (2011). "Digital Watermarking A Technology Overview " ***International Journal of Research and Reviews in Applied Sciences*** 89- 93.

- Nassereddin, H., Farzaeai, M., (2010). “Proposed Data Hiding Techniqie Text image Inside Image” ***International Journal of Research and Reviews in Applied Sciences*** 183- 193.

- Patsakis, C., Zigomitros, A., Papageorgiou, A., and Galván-López, E. (2014). “Distributing Privacy Policies Over Multimedia Content Across Multiple Online Social Networks”. ***Computer Networks.***

- BrahmaTeja, K., Madhumati, G., and Rao, K. (2012). “Data Hiding Using EDGE Based Steganography”. ***International Journal of Emerging Technology and Advanced Engineering,*** 285-290.

- Eastlake, D., and Jones, P. (2001). “US Secure Hash Algorithm 1 (SHA1)”. ***Network Working Group.***

- Jang, Y., and Kwak, J. (2013). “Access-control-based Efficient Privacy Protection Method for Social Networking Services”. ***International Journal of Security and Its Applications.***

- Kim, K., Bocharova, V., Halámek, J., Oh, M., and Katz, E. (2010). “Steganography and Encrypting Based on Immunochemical Systems”. ***Biotechnology and Bioengineering.***

- Liang, J., and Lai, X. (2007). “Improved Collision Attack on Hash Function MD5”. ***Journal of Computer Science and Technology.***

- Liu, Q., Sung, A., Ribeiro, B., Wei, M., Chen, Z., and Xu, J. (2008). “Image complexity and feature mining for steganalysis of least significant bit matching steganography”. ***Information Sciences,*** 21-36.

- Luo, W., Xie, Q., and Hengartner, U. (2009). "FaceCloak: An Architecture for User Privacy on Social Networking Sites". ***Cheriton School of Computer Science,*** 26-33.

- Niimi, M., Noda, H., and Kawaguchi, E. (1999). "Steganography Based on Region Segmentation with a Complexity Measure". ***Systems and Computers in Japan.***

- Por, L., and Delina, B. (2008). "Information Hiding: A New Approach in Text Steganography".***Advances on Applied Computer and Applied Computational Science,*** 685-695.

- Rivest, R. (1992). "The MD5 Message-Digest Algorithm". ***MIT Laboratory for Computer Science.***

- Schneier, B. (2015). ***Applied Cryptography Protocols, Algorithms and Source Code in C, 20th Anniversary Edition***. S.l.: John Wiley & Sons.

- Shilbayeh, N., Khuffash, S., Allymoun, M., and Al-Saidi, R. (2014). "Protecting the Privacy and Trust of VIP Users on Social Network Sites". ***World Academy of Science,*** 1488-1498.

- Singh, K., and Goel, C. (2014). "Using MD5 AND RSA Algorithm Improve Security in MANETs Systems". ***International Journal of Advances in Science and Technology (IJAST***), 48-54.

- Squicciarini, A., Shehab, M., & Wede, J. (2010). " Privacy policies for shared content in social network sites"***. The International Journal on Very Large Data Bases***.

- Vanitha, T., Souza, A., Rashmi, B., and DSouza, S. (2014). “A Review on Steganography - Least Significant Bit Algorithm and Discrete Wavelet Transform Algorithm”. ***International Journal of Innovative Research in Computer and Communication Engineering***, 89-95.

- Walia, P., and Thapar, V. (2014). “Implementation of New Modified MD5-512 bit Algorithm for Cryptography”. ***International Journal of Innovative Research in Advanced Engineering (IJIRAE)***, 87-97.

- Xiao, D., Liao, X., and Deng, S. (2005). “One-way Hash function construction based on the chaotic map with changeable-parameter”. ***Chaos, Solitons and Fractals,*** 65-71.

- Yadav, R., Saini, R., and Deep, K. (2011). “Cyclic combination method for digital image steganography with uniform distribution of message”. ***Advanced Computing: An International Journal (ACIJ)***

- Yuan, X., Wang, X., Wang, C., Squicciarini, A., and Ren, K. (2014). “Enabling Privacy-preserving Image-centric Social Discovery”. ***IEEE 34th International Conference on Distributed Computing Systems***, 198-207.

Appendix A : Software Code

The used code for the proposed technique

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Net;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace ImageSignature
{
    public class postion
    {
        public int x { get; set; }
        public int y { get; set; }
    }

    public partial class _Default : Page
    {
        Entities en = new Entities();

        string spliter = "_";

        protected void Page_Load(object sender, EventArgs e)
        {

        }


        private List<postion> getPositions(string currentDate, int
maxWidth, int maxHight, int length)
        {
            string preDate = currentDate;
            var date = DateTime.ParseExact(currentDate,
"yyyymmddHHMMss", null);
```

```
            var generatedXandY = getXandY(date);
            while (currentDate.Length < length)
            {
                currentDate += preDate;
            }
            int i = 1;
            List<postion> positions = new List<postion>();

            int prevY = generatedXandY.XValue.Value;

            foreach (var item in currentDate)
            {
                var itemInt = int.Parse(item.ToString());
                itemInt++;
                int key = prevY + (int)Math.Abs(((i + 2) * 2.5));
                prevY += i % generatedXandY.YValue.Value;
                int value = i;
                while (key >= maxHight)
                {
                    key /= 2;
                }
                if (key == 0)
                {
                    key++;
                }
                while (positions.Select(x2 => x2.x).Contains(key))
                {
                    key++;
                }

                for (int j = 0; j < 8; j++)
                {
                    while (value >= maxWidth)
                    {
                        value /= 2;
                    }
                    positions.Add(new postion() { x = key, y =
value });
                    value++;
                }
                i++;
            }
            return positions;
        }
        private string GetSHA1HashData(string str, bool saveIt =
false)
        {
            var user = en.Users.FirstOrDefault(x => x.UserName ==
User.Identity.Name);
            byte[] data = new byte[User.Identity.Name.Length];
            SHA256 sha = SHA256Managed.Create();
            byte[] sh1 = sha.ComputeHash(data);
            var value = Convert.ToBase64String(sh1);
            if (saveIt == true)
            {
                user.HashSignature = value;
                en.SaveChanges();
            }
            return value;
        }
```

```
        private string getAESHashData(string original, bool saveIt
= false)
        {
            using (Aes myAes = Aes.Create())
            {
                byte[] bytesToBeEncrypted =
Encoding.UTF8.GetBytes(original);
                byte[] passwordBytes =
Encoding.UTF8.GetBytes("MEU");

                passwordBytes =
SHA256.Create().ComputeHash(passwordBytes);
                byte[] bytesEncrypted =
AES_Encrypt(bytesToBeEncrypted, passwordBytes);
                string result =
Convert.ToBase64String(bytesEncrypted);
                if (saveIt == true)
                {
                    var user = en.Users.Where(x => x.UserName ==
User.Identity.Name).FirstOrDefault();
                    user.EASHash = result;
                    en.SaveChanges();
                }
                return result;
            }
        }

        public byte[] AES_Encrypt(byte[] bytesToBeEncrypted,
byte[] passwordBytes)
        {
            byte[] encryptedBytes = null;
            byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8
};
            using (MemoryStream ms = new MemoryStream())
            {
                using (RijndaelManaged AES = new
RijndaelManaged())
                {
                    AES.KeySize = 256;
                    AES.BlockSize = 128;
                    var key = new
Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);
                    AES.Key = key.GetBytes(AES.KeySize / 8);
                    AES.IV = key.GetBytes(AES.BlockSize / 8);
                    AES.Mode = CipherMode.CBC;
                using (var cs = new CryptoStream(ms,
AES.CreateEncryptor(), CryptoStreamMode.Write))
                    {
                        cs.Write(bytesToBeEncrypted, 0,
bytesToBeEncrypted.Length);
                        cs.Close();
                    }
                    encryptedBytes = ms.ToArray();
                }
            }
            return encryptedBytes;
        }
        protected void Encrypt_Click(object sender, EventArgs e)
        {
            var timeStamp = DateTime.Now;
            if (User.Identity.IsAuthenticated == false)
            {
                Response.Redirect("Account/Login.aspx");
            }
            var date = DateTime.Now;
            Bitmap image1 = new Bitmap(FileUpload1.FileContent);
            string currentDate = date.ToString("yyyymmddHHMMss");
            Bitmap newbitmap = new Bitmap(image1.Width,
image1.Height);
            byte blue = 0, temp = 0;
            int x = 0, y = 0;


            string text = "";
            Button source = (Button)sender;
            if (source.Text == "AES")
            {
                text = getAESHashData(User.Identity.Name, true);
            }
            else
            {
                text = GetSHA1HashData(User.Identity.Name);
            }
            var signature =
System.Text.Encoding.Unicode.GetBytes(text);

            byte[] bytearray =
System.Text.Encoding.Unicode.GetBytes(currentDate + spliter +
image1.Width + spliter + image1.Height + spliter +
signature.Count() + spliter);

            var positions = getPositions(currentDate,
image1.Width, image1.Height, signature.Count());

            foreach (byte b in bytearray)
            {
                for (int i = 0; i < 8; i++)
                {
                    Color pixelColor = image1.GetPixel(x, y);
                    blue = pixelColor.B;

                    temp = Convert.ToByte(b >> i);
                    temp = Convert.ToByte(temp & 0x01);
                    blue = Convert.ToByte(blue & 0xfe);
                    blue = Convert.ToByte(blue | temp);
                    Color newColor = Color.FromArgb(blue,
pixelColor.G, pixelColor.B);
                    newbitmap.SetPixel(x, y, newColor);
                    x++;
                }
            }
            x = 0; y = 1;
            int count = 0;

            string bitsVal = "";
            foreach (byte b in signature)
            {

                for (int i = 0; i < 8; i++)
                {
                    var position = positions[count++];
                    x = position.x;
                    y = position.y;

                    Color pixelColor = image1.GetPixel(x, y);
                    blue = pixelColor.B;
                    BitArray Rbits = new BitArray(new byte[] {
blue });
                    BitArray Tbits = new BitArray(new byte[] { b
});
                    if ((x + y) % 2 == 0)
                    {

                        if (Tbits[i] != Rbits[1])
                        {
                            Rbits[0] = Rbits[1];

                        }
                        Rbits[1] = Tbits[i];
                    }
                    else
                    {

                        if (Tbits[i] != Rbits[2])
                        {
                            Rbits[0] = Rbits[2];
                            Rbits[1] = Rbits[2];
                        }
                        Rbits[2] = Tbits[i];
                    }
                    byte[] bytes = new byte[1];
                    Rbits.CopyTo(bytes, 0);
                    blue = bytes[0];
                    Color newColor = Color.FromArgb(pixelColor.R,
pixelColor.G, blue);
                    newbitmap.SetPixel(x, y, newColor);
                    x++;

                }
            }
            x = 0;
            y = 1;

            for (; y < image1.Height; y++)
            {
                for (; x < image1.Width; x++)
                {
                    var pos = positions.Where(item => item.x == x
&& item.y == y).FirstOrDefault();
                    if (pos == null)
                    {
                        Color pixelColor = image1.GetPixel(x, y);
                        newbitmap.SetPixel(x, y, pixelColor);
                    }
                }
                x = 0;
            }



            encSig.Text = User.Identity.Name;
            string imageName = "~/Images/" + FileUpload1.FileName;
            string savePath = Server.MapPath(@"Images\" +
FileUpload1.FileName);
            newbitmap.Save(savePath,
System.Drawing.Imaging.ImageFormat.Jpeg);
```

```
            Image1.ImageUrl = imageName;

            encTime.Text = (DateTime.Now -
timeStamp).TotalSeconds.ToString();
        }


        protected void Decrypt_Click(object sender, EventArgs e)
        {
            var timeStamp = DateTime.Now;
            string filename = FileUpload2.FileName;
            Bitmap image1 = new Bitmap(FileUpload2.FileContent);
            byte[] bytarray = new byte[image1.Width];
            int x = 0, y = 0;
            byte blue = 0, temp = 0;

            for (int i = 0; i < image1.Width; i++)
            {
                blue = 0;
                for (int z = 0; z < 8; z++)
                {
                    if (x == image1.Width)
                    {
                        break;
                    }
                    Color pixelColor = image1.GetPixel(x, y);
                    temp = pixelColor.B;

                    temp = Convert.ToByte(temp & 0x01);
                    blue = Convert.ToByte(blue |
Convert.ToByte(temp << z));
                    x++;
                }
                bytarray[i] = blue;
                if (x == image1.Width)
                {
                    break;
                }
            }

            string text =
System.Text.Encoding.Unicode.GetString(bytarray.ToArray());
            string[] separator = new string[] { spliter };
            string[] info = text.Split(separator,
StringSplitOptions.RemoveEmptyEntries);
```

```
            var positions = getPositions(info[0],
int.Parse(info[1]), int.Parse(info[2]), int.Parse(info[3]));
            byte[] signetureArray = new byte[int.Parse(info[3])];
            int count = 0;
            x = 0;
            y = 1;

            for (int i = 0; i < signetureArray.Length; i++)
            {
                blue = 0;
                BitArray Rbits = new BitArray(8);

                for (int z = 0; z < 8; z++)
                {
                    var pos = positions[count++];
                    x = pos.x;
                    y = pos.y;
                    Color pixelColor = image1.GetPixel(x, y);
                    BitArray Tbits = new BitArray(new byte[] {
pixelColor.B });
                    if ((x + y) % 2 == 0)
                    {

                        Rbits[z] = Tbits[1];
                    }
                    else
                    {
                        Rbits[z] = Tbits[2];

                }
                byte[] bytes = new byte[1];
                Rbits.CopyTo(bytes, 0);
                blue = bytes[0];
                signetureArray[i] = blue;
            }
            string res =
Encoding.Unicode.GetString(signetureArray);
            var user = en.Users.FirstOrDefault(val => val.EASHash
== res || val.HashSignature == res);
            if (user != null)
            {
                signature.Text = user.UserName;
            }
            string imageName = "~/Images/" + FileUpload2.FileName;
            string savePath = Server.MapPath(@"Images\" +
FileUpload2.FileName);
            image1.Save(savePath,
System.Drawing.Imaging.ImageFormat.Jpeg);
            Image2.ImageUrl = imageName;
            decTime.Text = (DateTime.Now -
timeStamp).TotalSeconds.ToString();
        }

        public GeneratedXandY getXandY(DateTime date)
        {
            return en.GeneratedXandies.FirstOrDefault(x =>
x.StartTime <= date && x.EndTime >= date);
```

```
namespace ImageSignature
{
    public class Global : HttpApplication
    {
        void Application_Start(object sender, EventArgs e)
        {
            // Code that runs on application startup
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            AuthConfig.RegisterOpenAuth();
            startTimer();
            generateNewXandY();
        }
        void startTimer()
        {
            Timer tm = new Timer();
            tm.Interval = 3 * 60 * 60 *1000;
            tm.Enabled = true;
            tm.Elapsed += tm_Elapsed;
            tm.Start();
        }

        void tm_Elapsed(object sender, ElapsedEventArgs e)
        {
            generateNewXandY();
        }

        public void generateNewXandY()
        {
            Entities en = new Entities();
            var gen = new GeneratedXandY();
            Random r = new Random();
            gen.Id = en.GeneratedXandies.Count() + 1;
            gen.XValue = r.Next(1, 570);
            gen.YValue = r.Next(1, 7);
            gen.StartTime = DateTime.Now;
            gen.EndTime = DateTime.Now.AddHours(3);

            RNGCryptoServiceProvider rng = new
RNGCryptoServiceProvider();
            byte[] buff = new byte[128];
            rng.GetBytes(buff);
            gen.EASKey = Convert.ToBase64String(buff);
            en.GeneratedXandies.Add(gen);
            en.SaveChanges();
        }

        void Application_End(object sender, EventArgs e)
        {
            //  Code that runs on application shutdown



        void Application_Error(object sender, EventArgs e)
        {
            // Code that runs when an unhandled error occurs
```