Diyala Journal of Engineering Sciences

ISSN 1999-8716
Printed in Iraq

Second Engineering Scientific Conference
College of Engineering –University of Diyala
16-17 December. 2015, pp. 479-492

MULTIFACTOR AUTHENTICATION FOR SOFTWARE PROTECTION

Ali J. Abboud
Computer Engineering Department, University of Diyala
College of Engineering, Iraq-Diyala
E-mail: ali.j.abboud@gmail.com

ABSTRACT: - Software protection is a mechanism to make software systems more difficult (or impossible) to be accessed by attackers. Such mechanisms are gaining more importance nowadays for increasing number of attacks on the software systems in the large internet networked environment. Hence, in this paper, two schemes are proposed to achieve effective software protection tools. They are based on the use of multifactor authentication techniques and steganography algorithms. Multifactor authentication techniques are used to strength access policy to the software intellectual property while steganography algorithms are utilized to make protected software imperceptible to attackers. Finally, conducted experiments have shown that developed schemes are able to provide immunity against illegal accesses by invaders.
Keywords: Software Protection, Multifactor Authentication, Steganography, privacy, Security.

1. INTRODUCTION

Conveying data through the internet is becoming prevalent matter. Such data may be text, image, video, or software applications. Downloading software applications from apple store or Google play causes several problems to these applications by modifying this software by attackers and then uploading to the store web site [1]. Such behaviour is not acceptable at all because writing software application to solve a problem is non-trivial task. It needs time, efforts and patience to obtain effective software that overcomes problem within limited resources. In addition, the software code includes ideas, data and approaches to tackle certain real problem for specific applications or services. Hence, the contents of software represent valuable thing (intellectual property) that necessarily to be protected [2]. Software protection techniques can be categorized into three main classes [3]: (1) software obfuscation (2) software watermarking and (3) software tamper resistance.

Software obfuscation is a transformation process that change software into another form not understandable by attacker but it can be executed on the machine. While software watermarking is the process of embedding watermark inside the software to know the origin of the software (i.e. who produces this software). Finally, software tamper resistance is a process of preventing undesirable people from modifying the software in the illegal way. Also, we have to mention that some reports indicated that loses of software industry because threats hit approximately 51 billion dollars [4]. For stated earlier explanation, today large companies such as Microsoft, Google and others invest a lot of money in developing protection mechanisms to protect their products. To sum up, the software protection is an ultimate objective of this paper to secure software systems. To achieve this goal, multifactor authentication techniques and steganography algorithms are used to design robust and

capable software security mechanisms. The rest of paper is organized as follows: section II is dedicated to explain currently used software protection mechanisms, section III is used to explain multifactor authentication techniques, section IV is to present steganography techniques, section V is used to explain proposed schemes for software protection and finally conclusions and future work are presented in section VI.

2. SOFTWARE PROTECTION

Software protection techniques are divided into three main classes as mentioned in last section. In this section, we explain them in some details in aim to make picture clearer:

a) Software Obfuscation: It is a conversion way widely adopted to protect software systems from reverse engineering attacks. Several techniques are proposed in the literature in this category that can grouped into the following [2,5,8]:
   - *Layout Obfuscators*: In this transformation the source code formatting is removed and it cannot recover. Such conversion has low strength in protecting software systems since does change the semantics of the software. Techniques within this class including scramble identifiers, change formatting and remove comments.
   - *Control Obfuscators*: In this transformation the control flow of the source application is confused. This category of obfuscators is reliable and potent against different kinds of deobfuscation attacks. Techniques within this class including aggregation, ordering, opaque or computation.
   - *Data Obfuscators*: In this transformation the data structures of the source application is confused. This category of obfuscators including storage, encoding, aggregation or reordering techniques.
   - *Functional Obfuscators*: In this transformation the software application is obfuscated in a way that computationally impossible to recover original software.

b) Software Watermarking: It is a way to copyright protection of the software by embedding a piece of information inside the software script. Software watermarks can be categorized into the following classes [2,6]:
   - *Static Watermark*: It is stored inside the software code and does not change during the execution of the software. There are two types of this kind including data and code watermarks.
   - *Dynamic Watermark*: It is revealed during execution time of the software and stored into specific data structures.

   There are other watermarking techniques but not mentioned here for lack of space.

c) Software tamper resistance [2, 7]: it is used to prevent illegal users from modifying software (i.e. monitoring the integrity of the software). Techniques in this category including cyclic redundancy checksum, anti-debugging measures, white-box cryptography or virtual machine.

There are several toolkits in the research community to do software protection using known algorithms. Table I [9] shows a comparison among a number of obfuscation toolkits by several parameters including number of supported algorithms, user interface, flexibility, complexity, resource and cost. In this paper, some algorithms of sand mark software toolkit have been used to implement our proposed schemes. In the next section, we explain multifactor authentication mechanisms thoroughly.

3. MULTIFACTOR AUTHENTICATION

Authentication is the process of validating the identity of the user or machine is to be what claimed to be [10].The problem of one factor authentication systems such as password-based verification system is the necessity to store password database [10]. In addition the cracking of these systems by brute force attacks becomes easier in presence high speed computing devices that can generate millions of passwords per second. To mitigate the influence of these problems Multifactor authentication (MFA) security systems come in to

solve these problems. MFA is an effective mechanism to control access by authorized persons to the valuable resources such as software application, network server or computing device. In this mechanism the user must present more than one credential in order to be authenticated. These credentials or authentication factors varies according to the security needs of the application. The user should submit several authentication factors to the security system consecutively. These factors must be independent to avoid breaking the whole security system in case of compromising one of these credentials. Examples of these credentials are:

- Something you know (password).
- Something you have (Token).
- Something you are (Biometric).
- Somebody knows you (person).
- Time factor.
- Location factor.
- Mobile Phone.
- Graphical password images.

MFA reaps several benefits to enterprises, companies and individual users. These benefits include strong layer of security, dynamic selection of authentication factors and online monitoring and alerting of security threats.

4. STEGANOGRAPHY

Steganography is the art of hiding any kind of digital data such as text, image or video inside another cover digital media such as text, image, or video [11, 12]. The main idea of steganography is to conceal bits of file inside unused bits of another digital file such as (image, text, HTML, voice). It is different from cryptography that is not attracting attention for existence of information security techniques inside digital media. Therefore, we can say that steganography is trying to hide information from third force while cryptography is keeping information non-understandable by third force. Hence, steganography is the complementary of cryptography. There are many steganography algorithms suggested in the literature such as least significant bit (LSB) and hiding behind corner (HBE) algorithms. The steganography algorithms can be classified according to the encoding method into [11, 12]:

(1) Substitution: This method substitutes the redundant parts in the cover media with embedded data.
(2) Transform domain: This method conceal secret message inside transform domain of 1D or 2D digital signal.
(3) Spread spectrum: This method embed hidden data based on the concepts of spread spectrum communication.
(4) Statistical: This method hides secret data inside cover carrier by varying its statistical properties.
(5) Distortion: This method hides secret data by signal distorting.
(6) Cover generating: This method conceal secret message by creating secret communication way.

Steganography has several applications and one of most practical applications is the digital watermarking. It is used to embed logo, text or image inside the product to be authenticated Steganos and S-Tools. Fig. 1 below shows an example of modern. We have to mention that there are several software packages available to do steganography and watermarking such as Hide4PGP, MP3Stego, Stash, steganographic communication and how the encoding step of a steganographic system identifies redundant bits and then replaces a subset of them with data from a secret message [11]. In this paper, two steganography methods are used:-

a) LSB [12]: It is one of the oldest techniques to hide the secret message in the least significant bits of pixel values of carrier image in the spatial domain. The main idea of LSB is that human visual system does not need so much number of colours but just

sufficient number to discern main features in the image hence we can embed secret data in the least significant bits of image pixels without noticeable difference in the perception of the image.

b) Bit-Plane Complexity Segmentation Based Embedding (BPCS) [13]: It is a powerful and efficient steganography method proposed by Eiji Kawaguchi, Richard O. Eason to overcome the data hiding capacity weaknesses of earlier methods including LSB. The basic concept of this method is to divide n-bits pixels carrier image into (n) binary subimages. The cover image is then divided into informative regions and noisy regions. The secret data is hidden in the noisy region that replaces the noisy region of carrier image in such a way keep high quality cover image. In the LSB secret data is hidden in the least significant four bits of the carrier image while in the BPCS is hidden in the complex region of the most significant planes (MSP) and least significant planes (LSP). To sum up, the BPCS method is use human vision system characteristics to increase the embedding capacity and robustness of the cover image. The properties of steganography algorithm that specify its usefulness are:

1) Hiding Capacity: This property determine the size of information that can be embedded inside cover image.

2) Perceptual Transparency: This property refers the ability of steganography to keep hidden information unnoticeable to the attacker inside cover image without any distortion.

3) Robustness: This property means that concealed data still undestroyed if any of the following happened to the image: rotation, scaling, transformation, scaling, blurring or any other variation.

4) Tamper Resistance: This property means prevent attackers from damaging or modifying original secret message (i.e. preserves the integrity of secret message).

5. PROPOSED SOFTWARE PROTECTION SCHEMES

The basic component of our schemes is the multifactor authentication. It is composed of the following factors:

1. User information/user name and PIN: they are used together to protect user identity from theft and also in case of losing computing device.
2. **Global Positing system (location, time)**: in case of using mobile device, we can use the time and location of this device as authentication factor together with other factors and the time factor consists of year, month, day, hour, minute and seconds.
3. **Device Information**: The information of mobile computing device are used as authentication factors which consist of international mobile equipment identity number (**IMEI**) and international mobile subscriber identity number (**IMSI**).
4. **Social**: This new authentication factor represents the friend of your mine that can prove your identity.

We can illustrate the whole process of multifactor authentication process using the following example:

1. User name: john
2. PIN: 6253
3. Location: 032451
4. Time: 1/9/2015 at 2:30 AM
5. IMEI:354687901273491
6. IMSI:95443327889246
7. Friend name: smith
8. Start Multifactor authentication procedure after knowing all credentials:
   a) Link together user name and password to obtain (john6253). Then compute its hash value using MD5 algorithm to get the value (llJYhz/OGo92xDk0PJuy0g).
   b) Link together location and time to obtain (032451192015230). Then compute its hash value using MD5 algorithm to get the value (zIpBzR8opw17LE2+QYGRTA).

c) Link together IMEI and IMSI to obtain (354687901273491954433278892 46). Then compute its hash value using MD5 algorithm to get the value (91OFjsdAgRQz9FDFgIxqGw).

d) Compute the hash value of friend name (smith) using MD5 algorithm to obtain the value (yn51RBC0MIk5YfNovY+A+A).

e) Link together all hash values by xoring all hashed values and then hash this value using SHA1 hash algorithm to obtain final multifactor authentication password (10gtC/woqnO/EJ+6Wf39MUFcAlg). Based on the multifactor authentication techniques and steganography algorithms, two software protection schemes are proposed. These two schemes can be described as follows:

Scheme1: After-Obfuscation-Division-and- Steganography (**AODS**), this scheme works as shown in the Fig. 3. The input java software is protected if the machine or user authenticated to do protection for the software.

After the user/machine has been authenticated using different verification factors, the java software is obfuscated using AES encryption algorithm. For example, figure 4 shows a piece of java software and its obfuscation (i.e. encryption) illustrated in the figure 5. This is the first stage of software protection in our proposed schemes to secure java software codes. The next step is to divide software into several obfuscated blocks as shown in the figure 6 and hide each block inside single image. The embedding process is done using either **LSB** or **BPCS** algorithm. The final output of this process is a set of images to be sent to different locations in the networks or to be saved in different places of the computing device. This scheme is called AODS because software obfuscation is done and then divides obfuscated software into several blocks to be hided inside images.

Figure 7 shows the extraction process of embedded obfuscated blocks of java software from images. It starts by ensuring if the user is authenticated to do obfuscation or not. Multifactor authentication scheme is used for this purpose to verify the identity of the user as explained earlier in figure 2. The extraction of blocks is done using **LSB** or **BPCS** algorithm. The images are collected from computer network locations or from certain places inside computing device. The collection of images in this stage represents the obfuscated blocks of the java software that to be integrated together to obtain final obfuscated java software. The whole obfuscated software is then de-obfuscated using AES encryption/decryption algorithm to obtain original software java as shown in figure4.

Scheme2: Figure 8 shows the Before-Obfuscation-Division-and-Steganography (BODS), this scheme works as illustrated below.

The basic difference between **BODS** and **AODS** scheme is that **BODS** divides the input java software into several blocks before obfuscation while **AODS** divides input java software into several blocks after obfuscation**.** This scheme also uses multifactor authentication scheme to prove the individuality of the user by using several authentication factors as described earlier in figure 2. If the user is confirmed authorised to obfuscate software, then the input java software is divided into several equivalent blocks. The obfuscation (i.e. encryption) is applied for each one of these block separately without concern to the other blocks. After encryption of each block individually using AES algorithm and different ciphering key, each obfuscated encrypted block is embedded inside single different image.

Once embedded process is completed using one of steganography (i.e. embedding) algorithms described earlier, then these blocks are kept in different locations in the computer network with specific addresses or in different places of the computing device. We can observe from figures 3 and 8 that these schemes share the same obfuscation algorithm(s) and embedding techniques. Table II below shows differences between BODS and AODS.

Figure 9 shows the extraction process of embedded blocks in the **BODS** scheme. It starts by checking the authenticity of the user to do obfuscation. Multifactor authentication scheme is used for this purpose to verify the genuineness of the user as explained earlier in

figure 2. **LSB** or **BPCS** algorithm is used to extract embedded blocks inside images. These blocks are collected from computer network locations or from certain places inside computing device. The collection of blocks in this stage represents the obfuscated blocks of the java software that each to be de-obfuscated using different obfuscation algorithm (i.e. AES decryption algorithm and key). Finally, these de-obfuscated are integrated together to obtain final original java software. Figure 10 shows the images that are used to hide obfuscated blocks of java software. The quality of these images still the same after embedding the blocks (shown in the figure 6) inside them. Also after extraction the blocks from images using either **LSB** or **BPCS** algorithm, the quality of these images and blocks stay invariable with high quality during embedding and extraction processes. Furthermore, **BPCS** is capable to hide more information than LSB and it is more robust to different types of attacks. Finally, all experiments were conducted using Matlab software and sandmark software package available publicly.

6. SECURITY ANALYSIS

The information security systems are susceptible to different kinds of attacks in aim to break safety of these systems. In this paper, obfuscation and steganography techniques are used to protect software. However, there are some difficult attacks on the earlier mentioned safety techniques including compression, destroy everything, and reformat while other possible defeated attacks including visual, structural and statistical. In the following, we present some statistical metrics that prove immunity of our proposed schemes against some of these possible attacks as follows:

The first metric is the image histogram. It represents the distribution of pixels in the image and the frequency of these pixels. In the figure 11 above the first row represents the histograms of red, green and blue channels of original colour image while the second row in the figure represents the histograms of the red, green and blue channels of stego image (i.e. contains obfuscated software).

The clear fact in this figure that histograms of original image and stego image are similar. We can conclude from this fact that our developed schemes can resist statistical attacks since the attacker cannot recognize there is a hidden message inside the stego image.
The second metric is the correlation coefficient between original image and stego image. From table III below, we can notice that there is a strong correlation between colour channels of the original image and its counterpart of stego image. Also, the results in this table prove that both steganography algorithms have almost the same correlation with original image. Finally, this correlation metric inserts another proof that stego image is almost similar to the original image and there is no indication of hidden message.

The third metric is the entropy of image. It measures the amount of uncertainty in the image (i.e. amount of the information). Table IV presents the measurement of the entropy for the original images and the stego images. We can conclude from the results of this table that the entropy of all images is high. It is also all images have almost the same entropy value. These results add another indication that original images and the stego images have the same amount of information and can resist some earlier mentioned attacks. To sum up, all these metrics helps to ensure that proposed schemes have certain acceptable level of software security against some attacks.

7. CONCLUSIONS AND FUTURE WORK

This paper presented two novel schemes to protect software. These schemes are based on the multifactor authentication techniques to verify the identity of the user. The first scheme is called **AODS** since it obfuscate and then divide software into blocks. In contrast, the second scheme is called **BODS** since it is divide software and then obfuscate it**.** However, **BODS** uses different obfuscation for each software block, therefore gives more strength to the security of the java software.

Hence, **BODS** is better security scheme than AODS proposed scheme. In addition, we tested two steganography algorithms (i.e. **LSB** and **BPCS**), and we found that **BPCS** is better than **LSB** for several reasons including hiding capacity, robustness and efficiency. In the future work, we will try using other authentication factors in the multifactor authentication schemes such as biometrics and DNA cryptography. Also, our investigations will be extended further by protecting several software (s) simultaneously.

REFERENCES

1) Alizadeh, Mojtaba, Wan Haslina Hassan, and Touraj Khodadadi. "Feasibility of Implementing Multi-factor Authentication Schemes in Mobile Cloud Computing", 2014 Fifth International Conference on Intelligent Systems, Modelling and Simulation, 2014.
2) Collberg, Christian, Clark Thomborson, and Douglas Low," A taxonomy of obfuscating transformations", Department of Computer Science, The University of Auckland, New Zealand, 1997.
3) Collberg, Christian S., and Clark Thomborson. "Watermarking, tamper-proofing, and obfuscation-tools for software protection." Software Engineering, IEEE Transactions on 28.8, 735-746, 2002.
4) B. S. Alliance," Seventh annual bsa/idc global software 09 piracy study. [Online]. Available:http://portal.bsa.org/globalpiracy2009/studies/globalpiracystudy2009.pdf, May 2010.
5) Hou, Ting-Wei, Hsiang-Yang Chen, and Ming-Hsiu Tsai. "Three control flow obfuscation methods for Java software." IEE Proceedings-Software 153.2, 80-86, 2006.
6) Zeng, Ying, et al. "Robust software watermarking scheme based on obfuscated interpretation." Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE, 2010.
7) Wang, Ping, Seok-kyu Kang, and Kwangjo Kim. "Tamper resistant software through dynamic integrity checking." 2005.
8) Thomborson, Clark. "Benchmarking Obfuscators of Functionality.", arXiv preprint arXiv: 1501.02885, 2015.
9) Ertaul, Levent, and Suma Venkatesh. "Novel obfuscation algorithms for software security." Proceedings of the 2005 International Conference on Software Engineering Research and Practice, SERP. Vol. 5, 2005.
10) Stan Z. Li, Anil K. Jain," Encyclopedia of Biometrics", 2014.
11) Provos, Niels, and Peter Honeyman. "Hide and seek: An introduction to steganography." Security & Privacy, IEEE 1.3: 32-4, 2003.
12) Morkel, Tayana, Jan HP Eloff, and Martin S. Olivier. "An overview of image steganography." ISSA. 2005.
13) Kawaguchi, Eiji, and Richard O. Eason. "Principles and applications of BPCS steganography." Photonics East (ISAM, VVDC, IEMB). International Society for Optics and Photonics,

TABLE (1): COMPARISON OF OBFUSCATION TOOLKITS [9].

	JHide	Sand Mark	JObfuscator	JMangle
Supported Algorithms	30	25	1	1
User Interface	GUI	GUI	C L	GUI and CL
Flexibility	High	High	Low	Low
Complexity	Medium	High	High	High
Resource	Low	High	Medium	Medium
Cost	Free	Free	Trial	Free

TABLE (2): Comparison of BODS and AODS Schemes

	BODS	**AODS**
Software Division	The software is divided into several blocks **before** obfuscation.	The software is divided into several blocks **after** obfuscation.
Obfuscation	We can obfuscate each block individually using one or several algorithms.	We can obfuscate the whole software only using one or more obfuscation algorithms.
Security	stronger	strong

TABLE (3): CORRELATION BETWEEN SIMILAR COLOUR IMAGES

Steganography Method		LSB	BPCS
Image No.	Colour Channel		
1	R	0.980	0.997
	G	0.989	0.999
	B	0.972	0.996
2	R	0.994	0.999
	G	0.993	0.999
	B	0.989	0.997
3	R	0.995	0.999
	G	0.995	0.999
	B	0.999	0.999
4	R	0.997	0.977
	G	0.997	0.990
	B	0.998	0.985

TABLE (5): ENTROPY OF ORIGINAL, LSB AND BPCS COLOUR IMAGES

STEGANOGRAPHY METHOD/IMAGE NO.	ORIGINAL	LSB	BPCS
1	7.022	7.059	7.059
2	7.199	7.210	7.177
3	7.547	7.549	7.532
4	7.801	7.799	7.777

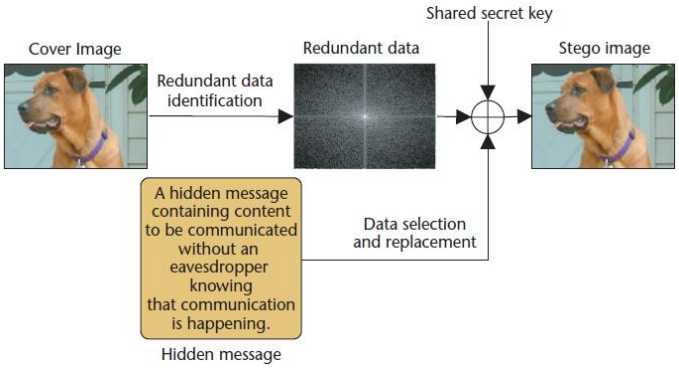


Figure (1): Modern steganographic communication. [11].

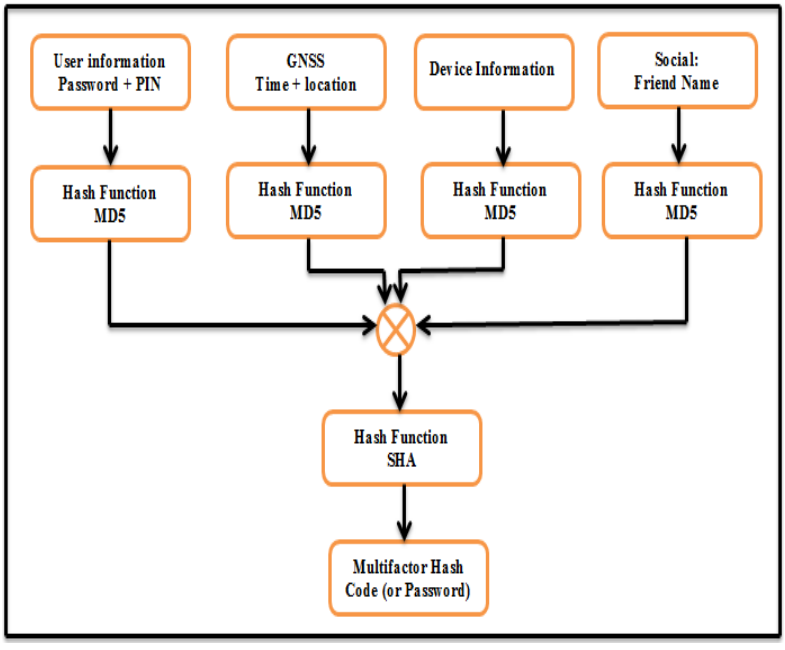
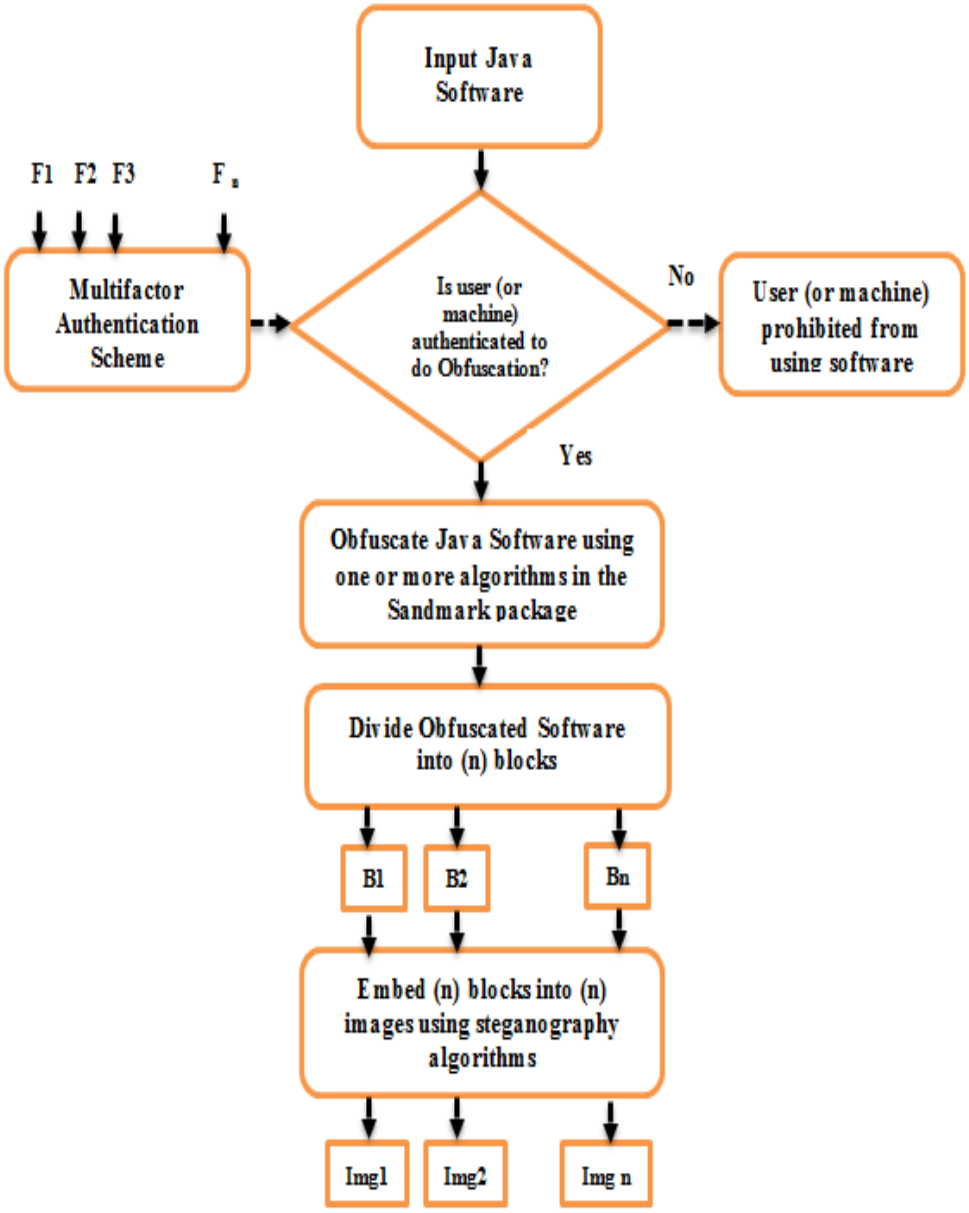


Figure (2): extraction process of (n) obfuscated software blocks from (n) pre-selected (n) images, scheme1.



Figure(3): Embedding process of (n) obfuscated software blocks inside (n) pre-selected (n) images, scheme1, AODS.

```
/**
 A class to measure time elapsed.
*/

public class Stopwatch
{
    private long startTime;
    private long stopTime;

    public static final double NANOS_PER_SEC = 1000000000.0;

        /**
         start the stop watch.
        */
        public void start(){
                startTime = System.nanoTime();
        }

        /**
         stop the stop watch.
        */
        public void stop()
        {       stopTime = System.nanoTime();   }

        /**
        elapsed time in seconds.
        @return the time recorded on the stopwatch in seconds
        */
        public double time()
        {       return (stopTime - startTime) / NANOS_PER_SEC;  }

        public String toString(){
            return "elapsed time: " + time() + " seconds.";
        }

        /**
        elapsed time in nanoseconds.
        @return the time recorded on the stopwatch in nanoseconds
        */
        public long timeInNanoseconds()
        {       return (stopTime - startTime);  }
}
```

Figure(4): Java software source code example.

```
U2FsdGVkX18YdSYUv9UITsliv/ZYf0plON+Gc9ab3iutReVszWpSl2xZDxOCUxT
my1xwJCBpsfheBJqmh0q+Xk0rxdOjHIdOBW+qW5E7Mo0x7HMVN/SVCGC49E
ZfQJd+xJiEMZHI9VvsWwFIOzj/0lv5aCVi0+GPgJktorpsGRJEXgBm8hwsi08Yy9O
n5cPb454aR7AFDKev8SgFDmp35hxuA1uDEOs3qQhwswafXN5+y/mN6/LTEnp/n
YPomMlWodBKZEyhFeIoOjJCuB6JSCdAUqeby700CZzazlxZoH0jbuUasBZlBMD
ke5qVdtRBBOpLuQLKPL5cEIUdyzbE6bjx/Gl1OVrbGRZMF1vhenijV88nT7SfqP6
GdJSaJW1KqvClTPoDVXIsdQkIDZjb9zTpM0iXmu+s9m2tWulsj0rWalhJHL0J67S
WTos1WQqPJNFO/V3yVVRoM5QxX2ioVa7Vp7PIPtwDhQ35igvz+AwzKm87rZc
kDuaQf4otEOUHqfaTXaQeZshI7P+C9l5O35by5KmwBjTJiUGMe7kVmm4ItpabHe
PsJoRkJE7YDk5YTeUav93vVic6TVQmAF6WjTV7DTS/lHOMZh2g64qmFpxGQz
GnmIVEfSbiynK4xWsowwCQ+xhRgUdN2RTsRytIPyZNnCfOKa9dogDly83QouG6
nO6e1AABHuAqarNSvqTG2G++3yZyNb8gTb2xSKZDMDQQW/xqEcfsGoexzNaY
Um9hUoQrdkNFwwGbd/ooBgIiw1XP7MgvDKN0a3TOsluc4s9C5s1UJaqYF7BTER
n6GitSxeuRziW2AiDHrROJOGqolXpiehMJWjFUmQOKsFY+VI7qBPBFfohVypnb
j9PFsHRH0UnBDIGmi+rvM2WyOBl2z+0ieHtrGWcQyO6Dv2h8EHAP5TnMnQC
W625NK6NLlc1JHbLDY6x0ByI8QD/NvTrAv28Tx/ikC9p4ipr2RtckV13nt9Z3thgrI
NsQsq+zpfwOyIJR8UkHR+eZYWz/5gd88K5gm4/BGLGvBJKfHgdNpFJGsSI02ihA
wtrjMXn4U1NbaqX/gN5xucmIX0/2VaLz
```

Figure(5): Obfuscated java software source code using AES encryption algorithm.

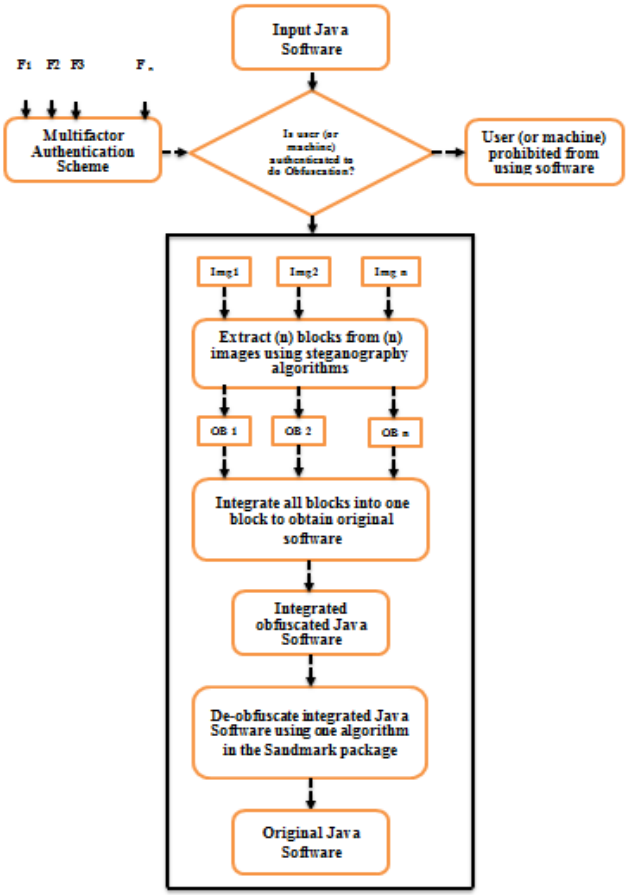
(a)
```
U2FsdGVkX18YdSYUv9UITsliv/ZYf0plON+Gc9ab3iutReVszWpSl2xZDxOCUxT
my1xwJCBpsfheBJqmh0q+Xk0rxdOjHIdOBW+qW5E7Mo0x7HMVN/SVCGC49E
ZfQJd+xJiEMZHI9VvsWwFIOzj/0lv5aCVi0+GPgJktorpsGRJEXgBm8hwsi08Yy9O
n5cPb454aR7AFDKev8SgFDmp35hxuA1uDEOs3qQhwswafXN5+y/mN6/LTEnp/n
YPomMlWodBKZEyhFeIoOjJCuB6JSCdAUqeby700CZzazlxZoH0jbuUasBZlBMD
```

(b)
```
ke5qVdtRBBOpLuQLKPL5cEIUdyzbE6bjx/Gl1OVrbGRZMF1vhenijV88nT7SfqP6
GdJSaJW1KqvClTPoDVXIsdQkIDZjb9zTpM0iXmu+s9m2tWulsj0rWalhJHL0J67S
WTos1WQqPJNFO/V3yVVRoM5QxX2ioVa7Vp7PIPtwDhQ35igvz+AwzKm87rZc
kDuaQf4otEOUHqfaTXaQeZshI7P+C9l5O35by5KmwBjTJiUGMe7kVmm4ItpabHe
PsJoRkJE7YDk5YTeUav93vVic6TVQmAF6WjTV7DTS/lHOMZh2g64qmFpxGQz
```

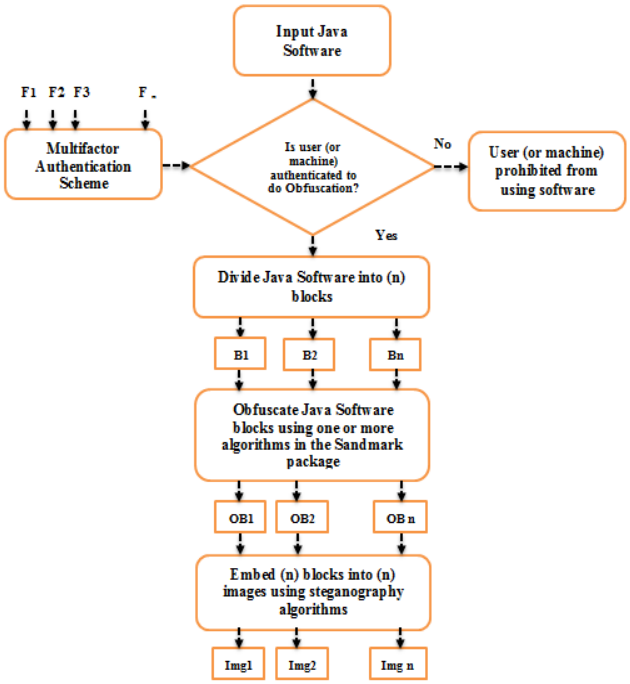
(c)
```
GnmIVEfSbiynK4xWsowwCQ+xhRgUdN2RTsRytIPyZNnCfOKa9dogDly83QouG6
nO6e1AABHuAqarNSvqTG2G++3yZyNb8gTb2xSKZDMDQQW/xqEcfsGoexzNaY
Um9hUoQrdkNFwwGbd/ooBgIiw1XP7MgvDKN0a3TOsluc4s9C5s1UJaqYF7BTER
n6GitSxeuRziW2AiDHrROJOGqolXpiehMJWjFUmQOKsFY+VI7qBPBFfohVypnb
```

(d)
```
j9PFsHRH0UnBDIGmi+rvM2WyOBl2z+0ieHtrGWcQyO6Dv2h8EHAP5TnMnQC
W625NK6NLlc1JHbLDY6x0ByI8QD/NvTrAv28Tx/ikC9p4ipr2RtckV13nt9Z3thgrI
NsQsq+zpfwOyIJR8UkHR+eZYWz/5gd88K5gm4/BGLGvBJKfHgdNpFJGsSI02ihA
wtrjMXn4U1NbaqX/gN5xucmIX0/2VaLz
```

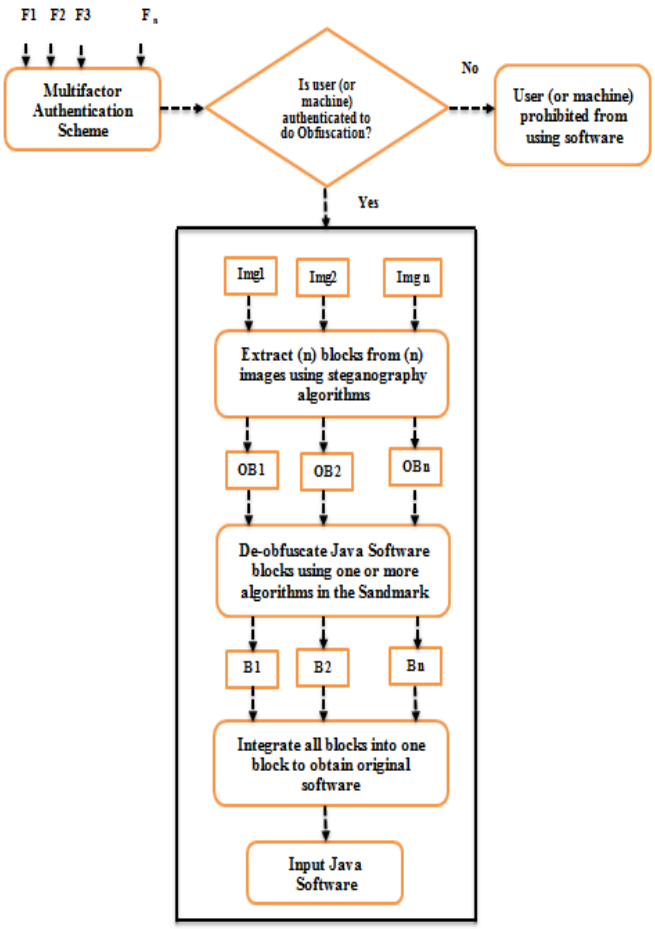
Figure(6): Dividing obfuscated java software into several blocks.



Figure(7): Extraction process of (n) obfuscated software blocks from (n) pre-selected (n) images, **scheme1, AODS.**

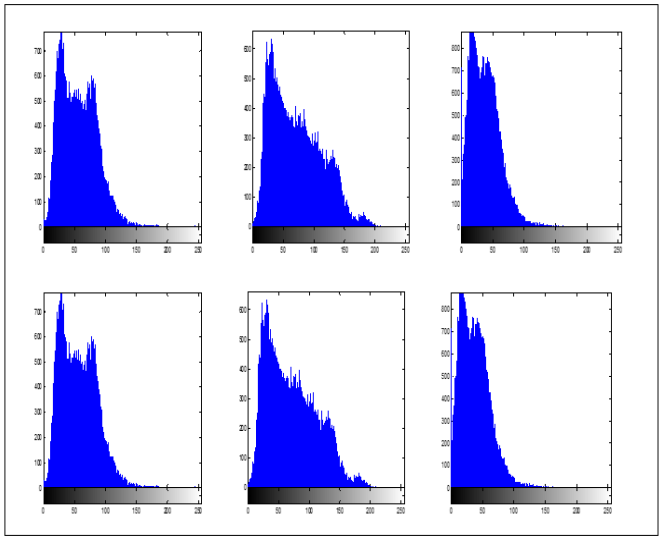


Figure(8): Embedding process of (n) obfuscated software blocks inside (n) pre-selected (n) images, **scheme2**, **BODS.**



Figure(9): Extraction process of (n) obfuscated software blocks from (n) pre-selected (n) images, **scheme2**, **BODS**.

Figure(10): The images that used to embed obfuscated software blocks.



Figure(11): Histogram based distribution of pixels in the R, G and B channels of colour image where first row represents the (R, G, B) of original image while second row represents the (R, G, B) of protected encoded image.