ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2019.0291

DATA PROTECTION ON MOBILE APPLICATIONS

R. Rejani and D. Murugan
Department of Computer Science and Engineering, Manonmaniam Sundaranar University, India

Abstract

Piracy has been a huge problem to the entire IT industry. In the recent years there has been an increase in the usage of mobile based applications and many a times these applications have been illegally copied and used a lot. Apart from applications, piracy also affects video, audio producers, digital book etc. This results in huge revenue loss to the application developers and also in many cases causes the spread of viruses and backdoor programs causing hijacking of personal data. The most commonly pirated mobile apps are games and utility apps. In this paper we are proposing a new method of protecting mobile apps using steganography which will help in protecting the mobile apps against piracy.

Keywords:

Data Protection, Steganography, Stego Image, Cover Image, LSB, MSB, RGB

1. INTRODUCTION

The communication technologies and broadband access across the world has grown at a great pace in recent times. This has also enabled digital piracy to flourish. Piracy enables unauthorized copying and distribution of any digital information which might include video, songs, software etc. Billions of dollar every year is lost due to illegal copying and distribution of software to the IT industry. Recently with the advent of smart phones and mobile computing, there has been a very large increase in mobile applications or software that can be used on the go. The main motive of using mobile phones has been communication and the exchange of information between two users. However, piracy has started becoming now popular for application which are used in mobile too. Wireless systems like mobile phones are more susceptible to security attacks (e.g. eavesdropping, unauthorized access) than wired networks. The pirated apps often spreads viruses and Trojan programs. This makes Data interception and tampering easy for any hacker with access to proper hardware and/or software tools and knowledge. To an extent, in many countries laws and other enforcement agencies can check piracy. But due to the sheer number of the mobile phones and applications none of these methods are inadequate. Hence it becomes all the more important for the software publisher or the developer to protect the software using different and innovative methods and make sure that it is not pirated easily [21].

In [18], the authors used steganography or cryptography to protect sensitive data as well as extend the usage to protect software from being pirated. In this paper we will discuss about a new method to prevent mobile application piracy by using steganography. Steganography is considered to be better than cryptography because the intended secret messages normally does not attract attention to itself for scrutiny.

A digital image is a collection of data/information about the pixels embedded in it. It is an aggregation of data. This is usually quite large compared to the message that we want to hide inside it. Hence we always prefer a digital image for covert communication channel. We can make use of its innocence for hiding data.

Each pixel in an image is a combination of RGB i.e. is (Red, Green, and Blue). A 24-bit bitmap will have 8 bits representing each of these three color values (red, green, and blue) at each pixel level. Hence it is able to make a wide variety of colors. Since the data is big, any minor and small change in the pixel intensity does not make any notable change. Also human eyes cannot identify the minor changes in the pixel. Maintaining picture quality is an important aspect of protection of the message. For enhancing this we have to use different steganography techniques.

2. RELATED WORK

In [18], the author presented a new software protection scheme for desktop applications using hardware features, cryptography, steganography and self-checks to make sure the executable files are not tampered. It works in four steps. In the first level it uses the hardware features, then strong encryption using triple DES, finally uses steganography and self-checks to hide key from users. A specific installed instance of software must be activated through interaction with the software provider and contains links to the hardware to insure that software cannot simply be copied to another machine. For ensuring more protection make use of Triple DES encryption standards and LSB based steganography. Hence cipher is securely hidden inside the stego image file and can be rechecked randomly to make sure the software is protected. These methods bind together and act as a single technique providing a secure and easy software protection method.

In [1], the authors presented a new steganographic approach within a MMS is presented and discussed. In [2], the authors examined how to hide text or image in mobile phone through Bluetooth. Chaos-based encryption algorithm for images is described in [3]. This algorithm is based on pixel scrambling where in the randomness of the chaos is made utilized to scramble the position of the pixels. Random pixel insertion method is used for hiding the secrete image in cover image. This application created for the Android operating system can be used in smart mobile phones for sending any image in a secrete manner by hiding it in another larger image. In [5], the author introduced related researches and problems of users' privacy protection in current mobile communication, and discusses to apply the idea of active defense to the security mechanism for better protection of users' privacy information.

In [6], the author introduced an improved method for hiding data in images or steganography. This method is used for secure data transfer from a computer to mobile phones. In this method a

message can hide in an image on a PC using a password. The user can download this image from the computer to his mobile phone. The decoder program running on his phone will extract the hidden information by a Java program. The decoder program was installed on a Nokia 6600 mobile phone and tested by posting the students' grades over it.

In [7], the author surveys the state of the art of steganographic techniques for smartphones, with emphasis on methods developed over the period 2005 to the second quarter of 2014. The different approaches are grouped according to the portion of the device used to hide information, leading to three different covert channels, i.e., local, object and network. Also, it reviews the relevant approaches used to detect and mitigate steganographic attacks or threats. Lastly, it showcases the most popular software applications to embed secret data into carriers, as well as possible future directions.

In [10], the study focuses on the implementation of a steganographic algorithm using wireless communication such as zigbee. Furthermore, this study presents experimental results obtained from testing a steganographic algorithm using zigbee devices. The main purpose of implementing such an algorithm using zigbee is to provide security on low and medium cost devices. A mobile application named MoBiSiS (Mobile Steganography Imaging System) is introduced in [12]. MoBiSiS improves the capability of steganography algorithm by implementing the steganography algorithm for Android based application. MoBiSiS is able to send the stego image through the Multimedia Messaging Service (MMS) and the stego image can be retrieved from the device's message inbox to extract the hidden message inside the stego image.

A new technique for secured image transmission is presented in [15], in this a dummy image will be used as a carrier of main image and main image will be hidden inside the dummy image using the most efficient LSB modification algorithm. In addition to this to hide main image inside the dummy image a random key will be used, the key and the main image both simultaneously embedded on the dummy image. The same key will be then used by the receiver to extract the hidden image inside the dummy image. In [16], a text steganography and image steganography is used.

3. HOW MOBILE APPS ARE DISTRIBUTED AND PIRATED

Let us first understand how mobile applications are distributed and pirated. Mobile OS providers like Google, Microsoft provide their own application repositories from where the applications can be downloaded and installed by any user. There are also various ways by which the applications can be downloaded for offline installation as an apk file or xap file. APK is the file format used by Android mobile OS and XAP is the ones used on Windows mobile OS. Once downloaded they can be installed or copied to any number of devices. Some applications do have built in security mechanisms sometimes like serial numbers and registration using email ids etc. However these methods often fail to stop the illegal pirating of the application [26].

As far as a mobile device is concerned it provides a more of a casual computing environment to the user. Devices are often used while doing various casual activities encouraging a relaxed state of mind that and hence most users are not properly alert to security concerns. The necessity of making mobile devices easy to use has made difficult to do basic tasks that may be common on desktop computers, such as identifying viruses. Additionally average consumers are usually not aware of mobile device security options and general best practices in security.

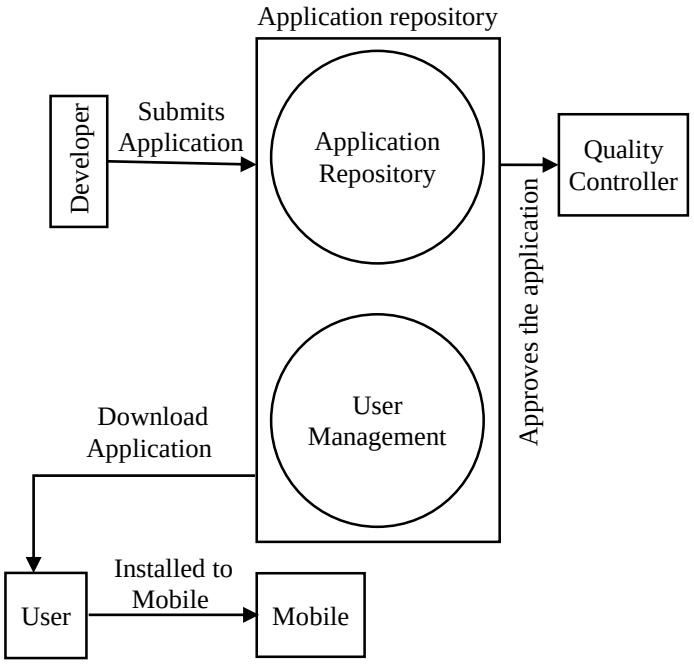

Fig.1. How mobile applications are distributed

In [20] Information Technology and Innovation Foundation and discusses how app developers fight against piracy. The methods discussed in this paper include methods like DRM (Digital Rights Management) to encrypt the application in order to bring in copy protection. Examples of DRM include the FairPlay system used by Apple to enforce licensing agreements on music downloads, the content scramble system (CSS) scheme used to encrypt video on DVDs.

However it has its own drawbacks as it is difficult to implement and another drawback is that it is that it will tie down the application to one device only. DRM also typically imposes additional requirements on the user that can, in some cases, reduce the value of the product. It also discusses about threatening the people who pirate software with legal consequences but these methods are often impractical.

In some places network management has been put in place in places like campus networks and other networks to limit the download capability of people after reaching a certain limit. However this will only have effect to an extent in pirating of applications across P2P networks.

In some countries ISPs have resorted to blocking sites which spread pirated apps. However this also have limited success because when a site is blocked other several sites are started which does the same sharing of pirated apps as before.

With regards to desktop applications in specialized areas there are copy protection techniques like USB dongles, presence of specific CDs carrying key files etc which to an extend can be used

as a protection method against piracy. However these cannot be used for protection of mobile apps.

4. PROPOSED TECHNIQUE

As we have already seen, steganography is the technique of embedding information into the cover image via text, video, and image without causing significant modification to the original image. The purpose of steganography is to hide the existence of the message hence it becomes difficult for attacker to detect it. The main challenge faced in data privacy is the need to share data while protecting identifiable information from hackers and other malicious attacks. We have seen that steganography and cryptography together can protect data effectively. Different steganography techniques have been developed to hide the message in an image. Steganography method hides the textual information in such a way that only sender and receiver can identify that a message is hidden in the image.

Let us see how we can use steganography as an authentication method to help in protecting mobile applications using a combination of steganography and encryption.

The methods that we can use to uniquely identify the user of a mobile application are

- A unique id like a MAC address of the mobile
- An email id used on the mobile phone
- A mobile number of a device to access the GSM network

We can combine the above mentioned properties along with steganography to protect apps. The basic frame work and steps to be done is depicted below.

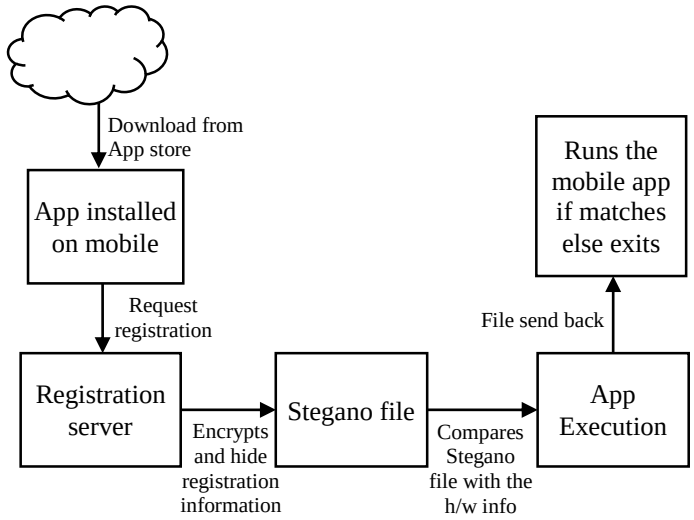

Fig.2. Architecture of mobile application protection

Step 1: User invokes the downloading from app store.

Step 2: Extract the IMEI number, mobile number, email address, mac address etc.

This process automatically detects any of the properties of the mobile which could be used for authentication like the IMEI number, mobile number, email address, mac address etc and transfers to the authentication server. This process automatically detects any of the properties of the mobile which could be used for authentication like the mobile number, email address, mac address etc and transfers to the authentication server.

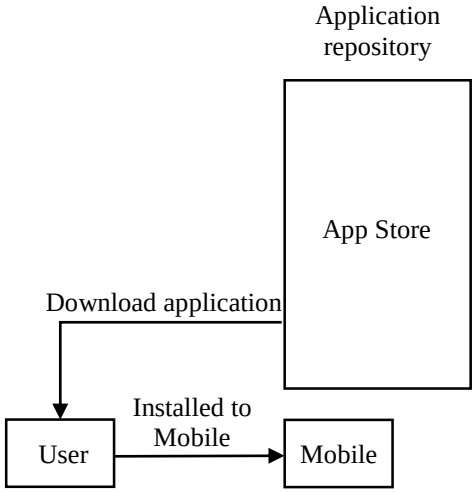

Fig.3. Downloading application from the mobile application store

Step 3: Saves the information in license server.

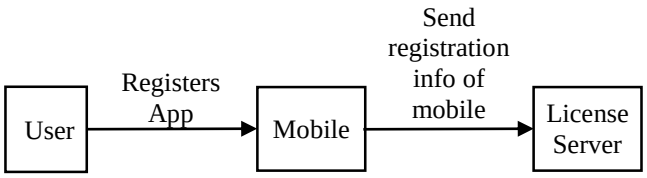

Fig.4. Application registration process

Step 4: Apply SteganoDB algorithm [17] to produce a JSON structure similar to one below and store within the image.

{
“First Name”:“Arun”,
“Last Name”:“Kumar”,
“Email”:“arun.kumar@outlook.com”,
“Address Line1”:“124#Lane 15”,
“Address Line2”:“T-Nagar”,
“Address Line3”:“Chennai-30”,
“Phone”:“999”
“Key”:“3WXpf6UXRV5nxAleQml9BRohlKw0Whs/XKGww4OdPb0yJ8qNZc0nSeSM5Kxldosp”
}

Step 5: The Licensing server creates an authentication token based on an image (using Pixel Pattern based Steganography algorithm [19]).

Windows.Media.Imaging Namespace provides classes to encode/decode or manipulate images. We have used the features of this Namespace to implement steganography. The function written to encode image is as below. This function will accept the image to store text to hide. It will store the text into bytes and will return the image.

Private byte[] EncodeText(byte[] Image, byte[] Text, int Threshold)

```
{
for (int i = 0; i < Text.Length; ++i)
{
int chr = Text[i];
for (int j = 7; j >= 0; --j, ++Threshold)
{
int b = (chr >> j) & 1;
Image[Threshold] = (byte)((Image[Threshold] & 0xFE)
| b);
}
}
return Image;
}
```

Step 6: Stores the token with user mobile.

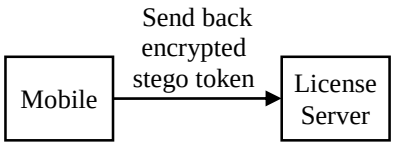

Fig.5. Authenticating the application with the registration information

Step 7: The application saves the stego file with the embedded information in the application folder in a hidden manner.

Step 8: Each usage of application it identify the user using stegofile.

Step 9: Extracts the identity information (using Pixel Pattern based Algorithm [19]).

The function to decode text from the image is as below: This function will accept image and will decode and provide text hidden in it.

```
private byte[] DecodeText(byte[] Image)
{
int length = 0;
int Threshold = 4096;
for (int i = 0; i < 4096; ++i)
{
length = (length << 1) | (Image[i] & 1);
}
byte[] decodedText = new byte[textLength];
for (int j = 0; j < decodedText.Length; ++j)
{
for (int k = 0; k < 8; ++k, ++Threshold)
{
decodedText[j] = (byte)((decodedText[j] << 1) |
(Image[Threshold] & 1));
}
}
return decodedText;
}
```

Step 10: Checks the identity.

For each subsequent application starts, the application will compare the mobile properties with the information in the stego file and in case it matches only will allow the execution of the application. Sometimes hackers will try to hack the application which will result in application files being changed. In this case the CRC checks will fail when compared between the values in the stego file and the application.

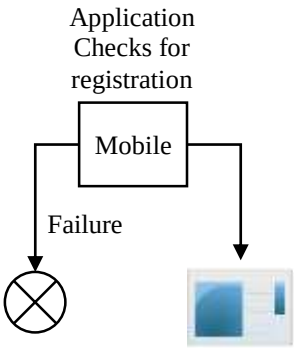

Fig.6. Registration checks while running application

Step 11: Periodic checks with the server file.

Along with this, it will randomly sends information to the authentication server to verify if the stego file are having the correct information. This can happen at random in background when there is internet connection available on the mobile

5. EXPERIMENTAL ANALYSIS

In order to test the methods discussed in this paper, we created a windows mobile based application with additional steganography [19] based protection included. The development environment used was Visual Studio 2013 and tests were run on two mobiles. Mobile-1 was a Lumia 535 running Windows Phone version 8.1 and Mobile-2 was a Lumia 822 running Windows phone 8.1. Both phones have 1GB RAM. The software can however be run on even mobile phones having 512MB RAM. Since most modern mobile phones now a days come with minimum of 1GB of RAM the software will run fine on any mobile phone. For steganography the application needs to have image manipulation capabilities.

Windows.Media.Imaging Namespace provides classes to encode/decode or manipulate images. We have used the features of this Namespace to implement steganography. The function written to encode image is as below. This function will accept the image to store text to hide. It will store the text into bytes and will return the image.

The function to decode text from the image is as below: This function will accept image and will decode and provide text hidden in it.

The sample application we created for the testing purpose is a very simple application to track expenses based on location where the expense occurred. We ran the application when it was protected using Steganography and also when it was not protected.

During the registration process the user information is gathered and it send to the server via message. The collected information is act as the key to the application. The output of the

authentication part gives one of the input of the protection algorithm.

Fig.7. Registration

Output of the Authentication Algorithm:

{
“First Name”:“Arun”,
“Last Name”:“Kumar”,
“Email”:“arun.kumar@outlook.com”,
“Address Line1”:“124#Lane 15”,
“Address Line2”:“T-Nagar”,
“Address Line3”:“Chennai-30”,
“Phone”:“9895312209”
“Key”:“U2FsdGVkX19L7/IiYCBYbur74I5oNTBL/nBaMPfgg+s=“
}

Stego Image Token Creation using the gathered information

Fig.8. Input Image (Cover Image) Output Image (Stego image)

This stegoimage is acted as the protection token of the application. It is unique hence if the application is pirating with the stegoimage also the application will not run. Hence this token protects the application from piracy.

Now consider the various execution conditions of the applications,

Test Case 1: This test case proves that if a mobile app is not protected, it can be easily copied and started on any mobile easily.

Test Case 2: Application which is protected with the stego method. When the application is initially run, it checks for the stego image, matches the values between the image and the mobile properties and if both matches then the application is started.

When the mobile protection module is enabled in the app, it will execute only on the mobile on which the app is registered.

Test Case 3: In case the Application is pirated and installed on a different mobile, the application will check for the registration information and will show an error message that the application is not registered. It will also provide an option to go to the registration screen and register the app.

Basically here the idea is that when an unauthorized user tries to run the app rather than giving error out the app will provide an option for him to register the app by paying and become a legitimate user. This will allow the app developer to get more users as well.

Fig.9. Mobile 1: (Lumia 535) running the application without the protection enabled

Fig.10. Mobile 2: (Lumia 822) running the application without enabling the protection

6. PERFORMANCE ANALYSIS

As a part of the performance analysis we generated the stego image with the user information, email, unique information IMEI number of the phone and address information. For this example, the IMEI number of the phone will be used to uniquely identify it. For testing used three different cover images.

Fig.11. Cover images

The cover images used in Fig.11 is to embed data and produce the stego image. These images were specifically selected because

they look quite innocent similar to normal logo at the same time provide a decent enough color range.

The registration information stored is as below which will be encrypted value of IMEI number along with the other registration info created after the processing of identity algorithm.

Input:

{

“First Name”:“Arun”,

“Last Name”:“Kumar”,

“Email”:“arun.kumar@outlook.com”,

“Address Line1”:“124#Lane 15”,

“Address Line2”:“T-Nagar”,

“Address Line3”:“Chennai-30”,

“Phone”:“9895312209”

“Key”:“U2FsdGVkX19L7/IiYCBYbur74I5oNTBL/nBaMPfgg+s=“

}

Fig.12. Stego images

6.1 CRITERIA I: COMPARISON OF IMAGE QUALITY OF THE RESULTANT IMAGES

In all these cases the image quality is not degrading. Hence any hacker trying to analyze data transfer will not be able to find it since the variation of image is minor the case of suspecting the image is less. From the detail study of image quality analysis of section above it is obviously known that noise is quite less compared to other techniques.

Table 1. Image quality analysis using variable sized images

Size of Image	Size of Data	PSNR	SSIM	MSE
1.22MB	243 bytes	95.1352dB	99.99999	.00014
756KB	243 bytes	94.3465dB	99.99999	.00021
598KB	243 bytes	91.1281dB	99.99999	.00045

6.2 CRITERIA 2: FILE/MESSAGE AS INPUT AND CHECKING THE IMAGE QUALITY

Considering the case of protection of message/file the size of data is much than the scenario application protection. In the case of protecting of messages the size of data is less compared to a file. Due to this reason for comparing image quality, here 3 text files are taken having different sizes and embedded inside the 3rd cover image. Calculated the Stegoimage’s PSNR values for analysis after this.

The Table.2 also shows the same conclusion of the previous table results, once again it can be seen that more amount of data can be protected safely inside the same image and it is safer than any other existing steganography technique.

Table 2. Image quality analysis using same sized images

Size of Image	Size of Data	PSNR	SSIM	MSE
598KB	500 bytes	90.7660dB	99.99999	.00049
598KB	4,895 bytes	90.4317dB	99.99999	.00053
598KB	19,586 bytes	87.9556dB	99.99999	.00094

6.3 CRITERIA 3: COMPARISON OF IMAGE QUALITY ANALYSIS BY CHECKING PSNR, MSE AND SSIM OF DIFFERENT TECHNOLOGIES

For going further evaluation in detail, here calculating the value of PSNR, SSIM and MSE values of different sized images by inserting the same size of data.

Table.3. Image quality analysis using variable sized images

Technology Used	Size of image	Size of Data	SSIM	MSE	PSNR Ratio
LSB	37.5KB	40 bits	.9554	9.6613	38.2805dB
LSB	190KB	40 bits	.9941	2.5076	44.1395dB
MSB	29.1KB	40 bits	.9947	1.8663	45.4205dB
MSB	190KB	40 bits	.9941	2.6766	53.3972dB
RGB	193KB	40 bits	.9531	2.2704	44.112dB
RGB	432KB	40 bits	.9964	5.8890	49.9728dB
Proposed	432KB	40 bits	1.000	0.00	1.#INF
Proposed	881KB	40 bits	1.000	0.00	1.#INF
Proposed	57KB	40 bits	.9999	0.00733	79.0206dB
Proposed	432KB	40 bits	1.000	.000429	91.3480dB

From the test results (Table.3), it is apparent that the results will show variance in values due to difference in the image size. Also when the image size becomes less, the ability of data storing decreases. Hence its PSNR value is decreasing. To avoid this and maintain the better quality of stego image, it is always better to use an image which is larger in size for storing the data. For storing large amount data, it is better to use large sized cover image.

Over LSB, MSB and RGB steganography, the proposed algorithm gives good results than other steganography techniques. Also it can be understood that while taking good sized image gives good perfection. In LSB, MSB and RGB steganography if the size of the image is reduced the quality of the image decreases, which means that for storing large data, large sized image should be used to keep image quality. But in the case of proposed method it is not needed in all cases. Commonly it will keep all the pixels in the same as original image, only rare cases change the pixels. Hence it keeps the image quality.

6.4 CRITERIA 4: IMAGE QUALITY ANALYSIS BY INSERTING DATA WITH DIFFERENT SIZES

To make sure the performance of the proposed algorithm, further tests are conducted with different size of data embedding in fixed sized image to clearly observe what changes will happen in the PSNR value and its image quality. For the second set of tests the image of Tajmahal (24 bit) was used but the size of

inserted text increased first to 17, 34, 50, 100 and 150 bytes. The size of the image used was 128KB.

Fig.13. Tajmahal (Cover image)

Table 4. Image quality analysis using variable size of data

Technology	Size of Data (bytes)	SSIM	MSE	PSNR (dB)
LSB	17	0.99993	0.00336	82.41242
MSB	17	0.99999	0.00199	84.69039
RGB	17	1	0.00187	84.94745
Proposed	17	1	0	1.#INF
LSB	34	0.99998	0.00605	79.85313
MSB	34	0.99999	0.00363	82.07162
RGB	34	1	0.00324	82.56271
Proposed	34	1	0	1.#INF
LSB	50	0.99996	0.01147	77.07855
MSB	50	0.99999	0.00546	80.30161
RGB	50	1	0.00548	80.28348
Proposed	50	1	0	1.#INF
LSB	100	0.99992	0.02131	74.38677
MSB	100	0.9995	0.01002	77.66094
RGB	100	0.99999	0.01026	77.56313
Proposed	100	1	0	1.#INF
LSB	150	0.99999	0.03031	72.85688
MSB	150	0.99999	0.01409	76.18274
RGB	150	0.99999	0.01388	76.24656
Proposed	150	1	0	1.#INF

From the test results (Table.4), it can be found that the proposed algorithm has better PSNR ratio and maintain the image quality compared with other techniques. Since the MSE, SSIM and PSNR values show no change, it can be understood that it doesn't change the pixels in the image when compared with the original image. Hence there is no degradation in the image quality.

6.5 CRITERIA 5: CALCULATING THE DATA INSERTION AND EXTRACTION TIME OF DIFFERENT TECHNOLOGIES

For comparing the efficiency of proposed algorithm the data insertion and data extraction time of various techniques were calculated. For that three size of data is considered then inserted into the same image and corresponding values are checked. The values got are shown in the Table.5.

Table.5. Comparison of insertion extraction time for different steganography techniques

	LSB	MSB	RGB	Pixel Pattern based
Size of data (bytes)	17	17	17	17
Time for Embed (ms)	500	580	550	2000
Time for Extraction (ms)	800	800	850	700
Size of data (bytes)	34	34	34	34
Time for Embed (ms)	600	600	600	2050
Time for Extraction (ms)	900	910	850	750
Size of data (bytes)	50	50	50	50
Time for Embed (ms)	900	900	950	2100
Time for Extraction (ms)	1100	1250	1050	900
Size of data (bytes)	100	100	100	100
Time for Embed (ms)	1000	1020	1000	2100
Time for Extraction (ms)	1300	1300	1250	950
Size of data (bytes)	200	200	200	200
Time for Embed (ms)	1100	1200	1150	2100
Time for Extraction (ms)	1450	1500	1400	1000

From the Table.5, it is seen that LSB, MSB and RGB steganography technique's data insertion time and extraction time is proportional to the data inserted. If the size of the data is increased the data insertion time is also increased, the same is happened in the case of data extraction too. One more thing is noted that, in all other cases the data extraction time is more than the data insertion time. But in the case of proposed algorithm the data insertion time is more compared to another algorithm. That is one of the limitations, but this data insertion time is not varies according to the size of data inserted. In the case of small size of data only it is an issue, if in the case of large amount of data (more than 300 bytes of data) this proposed algorithm took less time compared to the other algorithms. Also one more advantage is, it took less time for extracting the embedded text. That means it is data retrieval time is faster than other algorithms, it is having good response. That makes things easier, the hacker may not think about an algorithm is running to retrieve the data. This makes the data safer.

6.6 CRITERIA 6: QUALITY ANALYSIS OF DATA IN FILE OR MESSAGE OR APPLICATION PROTECTION

The conditions considered in this criteria are easiness in data embedding, accessibility of data in extraction time, completeness of data in retrieval, availability of data in time, data secureness and easy storage and retrieval. These conditions are compared for different data's and different size of data. According to the results data quality specification of data is summarized in Table.6.

Table.6. Comparison of insertion extraction time for different steganography techniques

Data Quality	File Protection	Message Protection	App. Protection
Reliability	High	High	High
Availability	Fast Retrieval	Fast Retrieval	Fast Retrieval
Security	High	High	High
Maintenance	Easy	Easy	Easy

The data extraction in this technique (Pixel Pattern based Steganography) is quick than compared to other techniques. It has proven in the section. In all these protections data quality is good and it has enough security due to the support of steganography and cryptography techniques. It is easy to maintain and use compared to any other techniques.

6.7 CRITERIA 7: SECURITY ANALYSIS - COMPARISON WITH OTHER PROTECTION TECHNIQUES

The security of this protection frame work is compared by analyzing against some of the known protection techniques. The considerations of comparison are how this new technique is excelled in the security, implementation cost, easiness to maintain in the usage and ability to resist the hacking possibilities. After the analysis the results are summarized as follows.

Table.7. Comparison of insertion extraction time for different steganography techniques

Technique	Security Level	Cost Benefit	Protection against Hacking and Piracy
Serial numbers	Low	Low	Very low. Easily replicable
Hardware dongles	Cannot be used with mobiles	NA	NA
Steganography + hardware features + encryption	Very high	Very Low	Extremely high protection.

Some of the mobile apps which use traditional method of protection just uses a serial number for protecting it against pirating. However the new method proposed uses the hardware features like IMEI number or mac address which are unique to a mobile. Hence it offers better protection against the traditional methods.

Secondly when compared to desktop protection methods, things like a hardware dongle cannot be attached against a mobile.

Finally, the proposed method uses a combination of steganography and encryption to protect the hardware generated key against any hacking attempt. This also makes the proposed method quite secure when compared against traditional protection methods.

Some of the various hacking methods used include steganalysis, brute force password cracking method, copying the software to another machine as is etc. Steganalysis will fail as the images do not have any variation as such. Even if a hacker tries to decrypt the text it will be near impossible to do this. Finally copying the software to another machine will not work as the hardware checks will fail.

7. CONCLUSIONS

Thus the heart rate analysis using Hilbert's Huang's transform for identifying heart disease was implemented and their performance were analyzed sufficiently and approximately. The proposed system would automatically estimate heart rate on the basis of power spectral density function and it would also perform efficient face detection using Viola Jones algorithm. This method provides highly accurate cardiovascular signal rate detection for disease identification. The experimental results were shown the guarantee and quiet simple process suitable for real time heart related applications and offline heart related applications.

In future work, the proposed framework will be developed with methodology to estimate heart related diseases based on the heart rate variations using multiple inputs of face video to provide vast improvements in real time applications. Intent to explore more observable features and several age classifications based on heart rate variations related to occurrence of heart diseases. Improvement of future work may also include improvement of intrinsic mode functions and thus increasing power spectral density.

REFERENCES

[1] Mohammad A. Haque, Ramin Irani, Kamal Nasrollahi and Thomas B. Moeslund, "Heart Beat Rate Measurement from Facial Video", *IEEE Intelligent Systems*, Vol. 31, No. 3, pp. 40-48, 2016.

[2] H. Rahman and M.U. Ahmed, S. Begum and P. Funk, "Real Time Heart Rate Monitoring from Facial RGB Color Video using Webcam", *Proceedings of 29th Annual Workshop of the Swedish Artificial Intelligence Society*, pp. 1-8, 2016.

[3] Simmi Dutta, Hiteshwar, Abhimanyu Dev Jamwal and Azhar Ud Din Guroo, "Heart Rate Detection using Independent Component Analysis and Multivariate Adaptive Regression Splines", *Imperial Journal of Interdisciplinary Research*, Vol. 2, No. 10, pp. 174-178, 2016.

[4] M. Kumar, A. Veeraraghavan and A. Sabharwal, "Distance PPG: Robust Non-Contact Vital Signs Monitoring using A Camera", *Biomedical Optics Express*, Vol. 6, No. 5, pp. 1565-1588, 2015.

[5] Hussain A. Jaber, A.L. Ziarjawey and Ilyas Cankaya, "Heart Rate Monitoring and PQRST Detection Based on Graphical User Interface with Matlab", *International Journal of Information and Electronics Engineering*, Vol. 5, No. 4, pp. 311-317, 2015.

[6] J. Moreno, J. Ramos-Castro, J. Movellan, E. Parrado, G. Rodas and L. Capdevila, "Facial Video-based Photoplethysmography to Detect HRV at Rest", *International Journal of Sports Medicine*, Vol. 36, No. 6, pp. 474-480, 2015.

[7] Larissa Carvalho, H.G.Virani and S. Kutty, "Analysis of Heart Rate Monitoring Using a Webcam", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 3, No. 5, pp. 1-7, 2014.

[8] X. Li, J. Chen, G. Zhao and M. Pietikainen, "Remote Heart Rate Measurement from Face Videos under Realistic Situations", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 4264-4271, 2014.

[9] R. Irani, K. Nasrollahi and T.B. Moeslund, "Improved Pulse Dectection from Head Motions using DCT", *Proceedings of* 9th *International Conference on Computer Vision Theory and Applications*, pp. 124-129, 2014.

[10] S. Thulasi Prasad and S. Varadarajan, "Heart Rate Detection using Hilbert Transform", *International Journal of Research in Engineering and Technology*, Vol. 2, No. 8, pp. 12-18, 2013.

[11] Gerard De Haan and Vincent Jeanne, "Robust Pulse Rate from Chrominance-Based rPPG", *IEEE Transactions on Biomedical Engineering*, Vol. 60, No. 10, pp. 94-128, 2013.

[12] G. Balakrishnan, F. Durand and J. Guttag, "Detecting Pulse from Head Motions in Video", *Proceedings of* IEEE *International Conference on Computer Vision and Pattern Recognition*, pp. 3430-3437, 2013.

[13] X. Yu, J. Huang, S. Zhang, W. Yan and D. Metaxas, "Posefree Facial Landmark Fitting Via Optimized Part Mixtures and Cascaded Deformable Shape Model", *Proceedings of* IEEE *International Conference on Computer Vision and Pattern Recognition*, pp. 1944-1951, 2013.

[14] Isayiyas Nigatu Tiba and Li Li, "Image-Based Automatic Pulse Rate Monitoring System Using PC Webcam", *International Journal of Engineering Research and Technology*, Vol. 2, No. 12, pp. 841-847, 2013.

[15] M. Soleymani, J. Lichtenauer, T. Pun and M. Pantic, "A Multimodal Database for Affect Recognition and Implicit Tagging", *IEEE Transactions on Affective Computing*, Vol. 3, No. 1, pp. 42-55, 2012.