Lightweight Steganography Technique for Smartphone

Mohammed J. Mustafa
Computer Science Department
University of Mosul
Mosul, Iraq

Ahmed S. Nori, PhD
Computer Science Department
University of Mosul
Mosul, Iraq

ABSTRACT

Steganography is an embedded technique that hides the message in a digital medium for the purpose of safely transferring the message. The objective of this paper is to develop and improve lightweight hiding techniques used in the systems of smartphones, by improving the well-known Least Significant Bit (LSB) algorithm by changing the attribute that is related to its work and suggesting some additions in a way that does not affect processing speed or memory usage and increases security. It also provides a large area for hiding a large secrete message with very little impact on the cover relative to the size of the cover used.

This paper proposes several additions and modifications to the algorithm, including: using Pseudo Random Number Generator (PRNG) to determine pixels locations in the cover. Coding process was also proposed for the purpose of increasing randomization and provide statistical distribution unrelated to the original message. The LSB hiding process enables the use of less important bits per pixel. In this case, a distribution of hidden bits within one pixel was suggested by different distributions in each hiding process, making data of repeated hiding useless for the attackers. Various measures (MSE, PSNR, BER, and SSIM) were adopted for the purpose of measuring the quality and efficiency of the proposed algorithm through several experiments including cover image sizes in different dimensions with a variety of data for secret messages, whose results were better than other values.

General Terms

Recognition, Security, Algorithms, Lightweight , Hiding.

Keywords

Steganography; PRNG; LSB; Lightweight; Smartphone.

1. INTRODUCTION

The development at communications technology led to the massive use of smartphones like Samsung, IPhone and other types with many applications such as social networking, trade fields as fast manner to share files, such as images, videos files and vocal messages. The smartphones have become an important part of people tradition in this century which have similar duties to computers. The improving in abilities and skills of smartphones will happen extremely with adding technologies of 4G and 5G will lead to continuous popular increasing use of smartphones.

Modern phones rely on the use of various operating systems, including Android, Apple's IOS system, and Windows phone system. Google's Android system is the most popular operating system on smartphones today, and there are hundreds of phone types for many companies that use this system. Android is an open source operating system based on the well-known Linux operating system but for mobile devices, such as smartphones and tablets. The Android system was developed by Open Handset Alliance and led by Google and other companies. Android provides a unified way to develop applications for mobile devices and Android applications are often developed using Java software, but recently a special language was developed for Android application programming by Google and Android stakeholders, and the result was Kotlin language. Kotlin which provides many fixed and repetitive processes in applications easily and easily without having to rewrite them every time.

The Android Studio development environment is used for software writing and app interface design and using a library group known as Android Software Development Kit or Android (SDK). The app is easily exported in the form of a package that can be sold through online markets such as Google Play or Amazon Appstore [1][2].

The characteristics and features of personal computer became gradually being transferred to mobile devices, often called smart devices such as smartphones, tablets, and e-book portals. These days' smart devices have become basically for daily social activities, giving better processing, greater storage capacity, and a wide possibility of messaging information. In a current and constantly changing environment, creating a security connection is a very important concept for research. Unfortunately, today's smartphone users are in a very similar situation to pc users a few decades ago. The researches of smartphones' security are insufficient, and most of them have not been developed completely and still are theoretical. As a result, most smartphones lack the level of security which found with computers while the complexity of smart devices in terms of specifications and performance is constantly increasing as well as the number and type of security threats makes them an easy target for hackers and malicious applications, as is the case with desktop computers [3].

Data can be protected by using Cryptography and Hiding Information, an embedding technology that hides the message in a digital medium for secure transmission. Data hiding methods can be divided into reversible and non-reversible technologies, non-reversible technologies often provide more space and better cover quality than reversible ones, so non-reversible applications such as image documentation, discovering manipulation [1][4][5].

2. THE PURPOSE OF PAPER

The paper aims at developing and improving the lightweight hiding techniques used in smart device systems, in addition to designing and implementing lightweight hiding technology capable of working efficiently in the environment of smartphones, by developing the LSB algorithm known for changing its work in a way that does not affect the speed of processing or using memory increases the degree of security, and provides a lot of space to hide a large secret message with very little effect on the cover relative to the size of the cover used. The technology must be safe, strong against detection and retrieval attacks, and do not require a large amount of processing, in order to be a lightweight, efficient hiding technique. Work is applied using Kotlin language and Android Studio.

3. RELATED WORKS

In 2014, Thanikkal and his colleagues introduced an Android smartphone app for data hiding, in which LSB algorithm was used to hide data sequentially. The hiding process was not directly included, but the logical XOR function was used to locate the core values from their sources sites, because the function is mutual and reversible, the value of the L site is calculated per bit of the M secret message with the least Significant bit in the pixel corresponding to its P, the way :

L = XOR (M, P)

Store the resulting location values in a matrix as a location map. If the value of the L = 1 is that the bit value of the message is equal to the value of the less Significant bit in pixels, otherwise the bit value of the message is the opposite of the bit value that is less Significant than the pixels. The map of the sites is sent separately from the image to the recipient and the image sends the cover to the recipient as it is without any modification. The recipient scans the values of the location map and when he finds value 1, he takes the least Significant bit value as it is, or flips flip the less Significant bit value. The process continues until the secret message is fully retrieved. This method provides an ideal state of non-detection of a hidden message, the cover image and the stego image are fully matched without any change. But it generates a new problem: how to share the map of the sites between the sender and the receiver in a secure manner in each hiding process, making this method impractical and ineffective for actual use [1].

In 2015, Zainab khyioon Abdalrdha developed a proposed algorithm to protect and hide data in smartphones, the proposed algorithm is to hide the secret message by LSB technology and improve hiding by making simple encryption of the secret message bits by performing the logical XOR process between bits The message has a short secret key, and then hides the resulting inside the cover image file. The first bit of each pixel is used and the hiding starts from the 100 pixels and above sequentially, in addition to the algorithm encrypts the message before embedding it in the cover using the NTRU algorithm to provide recoverable encrypted text. It is a public key-type encryption algorithm and one of its most important features is that it needs very little memory and very low processing quantity and provides a high level of security [6].

In 2017, Richard Apau and Clement Adomako used the RSA algorithm to decode data and integrate it with the embedding process using the LSB algorithm, a proposed data protection algorithm in smartphones and within android, where encrypted data is hidden using an encryption algorithm. The RSA asymmetric key inside a digital image acts as a cover chosen from a video file frame. Although the proposed method was straightforward and simple and the only development that was presented was the process of selecting specific images to hide from the video file, the results were fairly good and according to the standards presented by the researchers in their research pages [7].

In their 2018 research, MUSTAFA KASAPBASI and WISAM ELMASRY combined a range of techniques to obtain a new way to hide data in color images in order to improve efficiency, increase the load capacity of confidential data, while maintaining the integrity and security of that data using encryption with encryption. The process of hiding. In the proposed method, the coded section of the secret message is first configured with CRC32 as a checksum, after which the coded section is compressed in Gzip mode before being encrypted using the AES algorithm and added to it as information for subsequent processors, the process of embedding encrypted data. The introduction is done using the LSB algorithm, and the location of the following pixel for the embedding process is made using the Fisher-Yates Shuffle algorithm. The search results indicated that the capacity, security and integration of hidden data have increased. But the treatment has become very large and long and needs high processing power [8].

4. LIGHTWEIGHT APPLICATIONS

In computing and computer science, the lightweight program, also called lightweight software or application, is known as a computer program designed to have a small memory space (RAM usage) and low (CPU usage), and generally using Low for system resources. To achieve this, you should avoid using lengthy and overused software and codes and try to find the most efficient algorithm for purpose. Lightweight applications are also called Lite, an acronym for lightweight [9][10][11][12].

The efficiency of lightweight applications is often measured by calculating the execution time and usage ratio of the processor and memory, the less execution time the more efficient and the application is lightweight and vice versa, and if a very long time is needed for implementation then it cannot be considered lightweight.

The amount of processing (processor usage rate) is inversely proportional to the efficiency of the lightweight program, the higher the processor usage rate, the less efficient the program is as a lightweight application. If the application needs all the processor time to perform, which means delays in other functions, in this case it is not considered a lightweight application. So it can be said that the program has to scare the weight to achieve what comes [4]:

1. Speed of execution (low execution time)

2-A relatively acceptable treatment amount and no pressure on the processor.

3- Little use of memory

5. THE HIDING IN LIGHTWEIGHT SYSTEMS

Data hiding is an application that often requires a lot of complex processes and processes with its simplest units (bits) with many repetitive processes, the more complex the algorithm used to hide, the more difficult it is to detect and attack the cover. Traditional security algorithms may not be suitable for working on their smartphones and operating systems because they are designed to provide high levels of security without focusing heavily on optimal use of resources [13].

Actual smartphones suffer from a set of parameters, they store images locally, have some computing power, and the screens of these small devices usually have a few hundred pixels each side, and can at best display several thousand colors. The smartphone CPU is rarely equipped with special help processors for decimal numbers, and operates at a smaller frequency for computers. Moreover, smartphone RAM is smaller in size than those found in computers [11][14].

However, those limited resources, such as energy or communications, are extremely limited and limit the functions that these systems can provide. Security is particularly affected by this because most of the basic algorithms and protocols for encryption and hiding are too heavy. Therefore, it is difficult to use such devices as a device to hide complete information because of their limited capabilities, so that the designer faces two options: first - the use of strong security methods, but this negatively affects the operation of the system and the way the user interacts with the system, or the second - the use of weaker security methods so as not to affects the expected performance of the system [13][15].

However, smartphones have become widely used and depended, and it can be very useful to use them to securely store a small amount of confidential data, such as passwords, pin numbers, identity data and credit card numbers. In addition, confidential data stored in your smartphone can be safely sent and received via multimedia messages. Therefore, it is necessary to develop algorithms and build lightweight applications to take into account all those parameters for the purpose of hiding the security and robust data [11].

The idea of hiding in expert systems lies in two basic ideas: the first is to sacrifice a large part of the complexity of hiding algorithms and thereby reduce the amount of processing and make the application lightweight, and to sacrifice part of the strength and resistance of the algorithm used against detection and various attacks. The second idea is to try to find concise methods and simplified methods of applying hiding algorithm with a much lower processing amount than the original algorithm and thus achieving the standards of lightweight applications[4][12].

6. HIDING TECHNIQUES IN LIGHTWEIGHT SYSTEMS

The hiding techniques in lightweight systems general1y with smartphones in particular can be limited so that the inclusion technology in LSB is almost the most common and widely used method in the applications of hiding for smartphones and lightweight systems. Many researchers have used some modification and improvement and combined with different coding algorithms[1][3][7][8][13][16].

Perhaps the reason for the great popularity of this method in lightweight systems being an algorithm, it is not complicated so it does not need a lot of sources, but its strength point is its weakness, it is subject to detection because of its wide use that made it known to everyone, as well as for the ease of retrieving the message from it with a set of analysis and experimentation. Therefore, the tendency has always been to change and complicate the LSB hiding process while maintaining the properties of its lightweight applications by finding different methods in selecting the pixels used for hiding and the sequence in which they are hiding [3][8].

Many, however, tended to maintain simplicity and the well-known body of algorithm and reliance on encryption for the purpose of overcoming its weaknesses [6][17].

Some have combined hiding in LSB with other hiding algorithms to increase security, such as XOR [1][6].

7. STEPS OF THE PROPOSED LIGHTWEIGHT HIDING SYSTEM

Proposed hiding algorithm

Inputs: image cover, secret message, number of embedded bits.

Outputs: The stego-image which represent the embedded secret message and its details. Note Fig. 1.

Step 1: Read the secret message file and cover image, then select the number of embedding bits per pixel.

Step 2: Copy the cover image to a new image for stego embedding and then convert the secret message to the binary version and thus into a torrent of bits.

Step 3: Check the cover image capacity is it enough to hide the secret message?

Yes: Continue to Step 7. No: View an alert message and move on to step 16.

Step 4: Generate a random number that represents the distribution of embedding bits per pixel and the number of bits

Step 5: Calculate the seed value of PRNG generator by equation:

Seed=√ (width^2+height^2) + (R XOR G XOR B)

Step 6: Use the seed value to generate random values and the number of image pixels to rescatter and distribute the built-in values.

Step 7: Hide the number of bits in the two less Significant bits from the first pixels and hide the type of secret message file in the two less Significant bits than the second pixel stake in the embedding pixels.

Step 8: Determine the storage space that the length of the secret message needs in the equation LengthSize = Ceil (Log2 (width * height) / 8))

Step 9: Hide the length of the message and the length of LengthSize in the less Significant bits of the image cover starting from the third pixel of the embedding pixels and by the size of the length field of step 8 and the number of embedding bits per pixel.

Step 10: Hide the sequence number that was randomly generated in step 4 in the following pixels of the cover image.

Step 11: Take a clip of the torrent of secret message bits and the number of embedding bits

Step 12: Calculate the next pixel location selected for inclusion based on PRNG generated in step 6 of the two equations:

Xi=PRNGi mod Cover Width Yi = PRNGi / Cover Width

Step 13: Encode the value of every pixel from the selected pitch randomly and by Table 1.

Step 14: Include the selected pitch of the pixels in the site (x, y) of the cover image using a set of logical processes (or, and) on the pixels.

Step 15: Re-steps from 11 to 14 until the secret message is finished and then store a stego image on a file for the purpose of sending or saving.

Step 16: The End

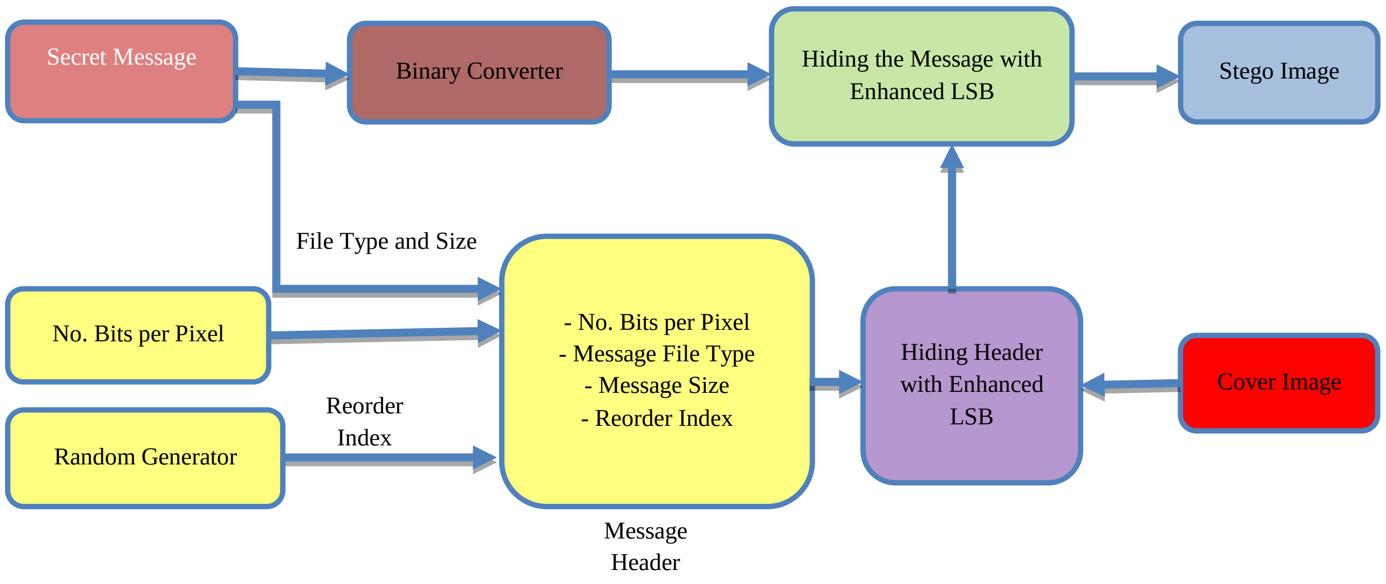


Fig. 1: Proposed hiding algorithm diagram

Proposed retrieve algorithm

Input: The stego- image.

Outputs: The secret message file. Note Fig. 2.

Step 1: Read the stego image.

Step 2: Determine the size of the allocation along the message from the equation:

LengthSize = Ceil (Log2 (width * height) / 8))

Step 3: Calculate the seed value of PRNG generator by equation:

Seed=√ (width^2+height^2) + (R XOR G XOR B)

Step 4: Use the seed value to generate random values and the number of image pixels for the purpose of mixing and distributing built-in values.

Step 5: Retrieve the number of embedding bits from the two less important bits of the first pixel, and then retrieve the type of secret message file from the two less important bits than the second pixels of the embedding pixels. Then retrieve the length of the message from the following pixels for inclusion and along LengthSize.

Step 6: Determine the length of the distribution index number based on the number of bits

Step 7: Retrieve the index number from the following embed pixels from the stego image.

Step 8: Calculate the location of the following pixels from the embedding pixels of the two terms:

Xi = PRNGi mod Cover Width Yi = PRNGi / Cover Width

Step 9: Retrieve the values of the bits that are less important than the three color levels and the number of embedded bits and according to the specified distribution value.

Step 10: Decode the three bits resulting from the previous step to one bit is the original bit of the secret message and the reliance.

Step 11: Return steps from 8 to 10 until the entire message is retrieved according to the length of the message and then stored.

Step 12: The end.

Table 1. Encoding The Message Bits

Corresponding coding		**4**
Code in binary	**Code in decimal**	
0 0 1	**1**	**0**
1 0 0	**4**	
1 0 1	**5**	
1 1 1	**7**	
0 0 0	**0**	**1**
0 1 0	**2**	
0 1 1	**3**	
1 1 0	**6**	

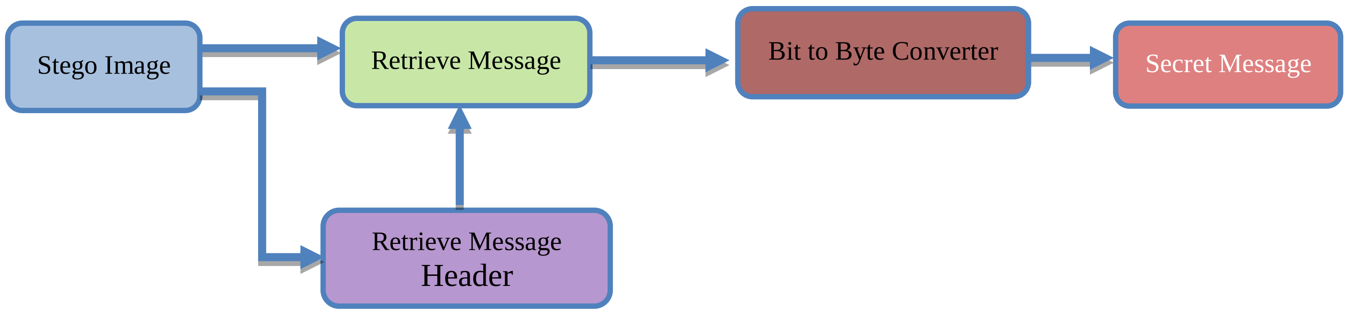


Fig. 2. Proposed retrieval algorithm diagram

8. RESULTS

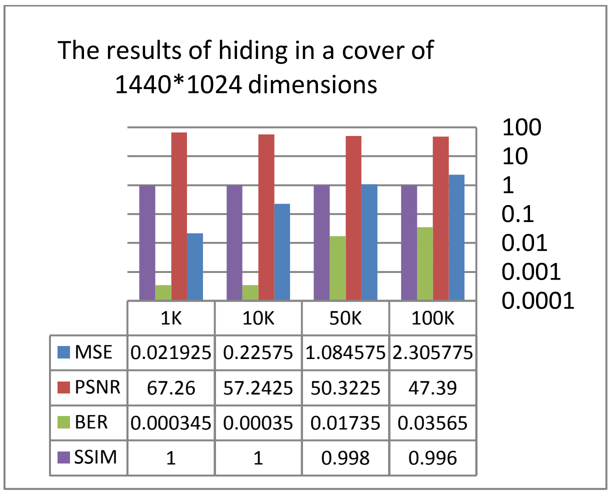
The proposed hiding technique was applied to a different number of less significant bits of cover image pixels, by choosing a color cover image in different sizes by hiding secret message in different sizes. All probabilities of the number of possible bits (4 probabilities) were recorded by calculating the relevant metrics as well as the time of implementation. A comparison of the results obtained with previous work is summarized as shown in the following tables Tables.2 _____, Tables.3 _____, Tables.4 ______. And Note Fig. 3. _____, Fig. 4. _____.

Table 2. The results of hiding secret message in different sizes and cover size with dimensions of1440*1024.

Data size	NO. bits	MSE	PSNR	SSIM	time (ns)
1K	1	0.0028	73.64	1	27
	2	0.0069	69.77	1	27
	3	0.0190	65.26	1	27
	4	0.0590	60.37	1	27
10K	1	0.0278	63.68	1	27
	2	0.0676	59.83	1	27
	3	0.1962	55.20	1	27
	4	0.6114	50.26	1	27
50K	1	0.1389	56.70	0.9980	27
	2	0.3374	52.84	0.9980	27
	3	0.9733	48.23	0.9980	27
	4	2.8887	43.52	0.9980	27
100K	1	0.2777	53.69	0.9960	27
	2	0.6739	49.84	0.9960	27
	3	1.9611	45.90	0.9960	27
	4	6.3104	40.13	0.9960	27

Table 3. The results of hiding secret message in different sizes and cover size with dimensions of1800*2800

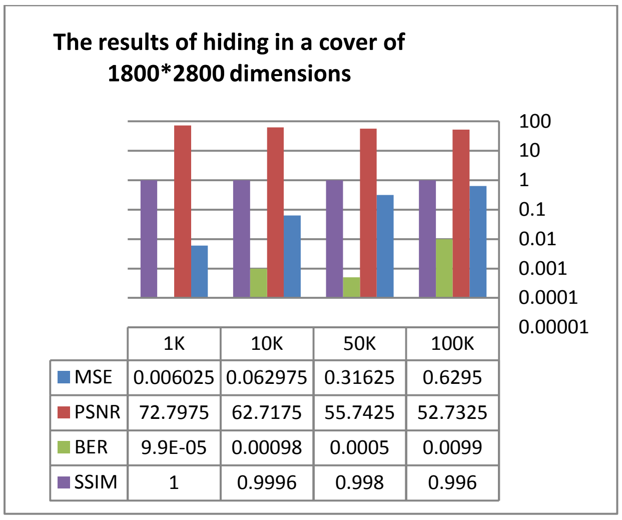
Data size	NO. bits	MSE	PSNR	SSIM	time (ns)
1K	1	0.0007	79.10	1	130
	2	0.0019	75.40	1	130
	3	0.0055	70.69	1	130
	4	0.0160	66.00	1	130
10K	1	0.0079	69.16	0.9996	130
	2	0.0190	65.29	0.9996	130
	3	0.0550	60.69	0.9996	130
	4	0.1700	55.73	0.9996	130
50K	1	0.0400	62.16	0.9980	130
	2	0.0950	58.32	0.9980	130
	3	0.2700	53.72	0.9980	130
	4	0.8600	48.77	0.9980	130
100K	1	0.0780	59.15	0.9960	130
	2	0.1900	55.31	0.9960	130
	3	0.5500	50.67	0.9960	130
	4	1.7000	45.80	0.9960	130



	1K	10K	50K	100K
🟦 MSE	0.021925	0.22575	1.084575	2.305775
🟥 PSNR	67.26	57.2425	50.3225	47.39
🟩 BER	0.000345	0.00035	0.01735	0.03565
🟪 SSIM	1	1	0.998	0.996

Table 4. Comparing results with related works

SSIM		MSE		PSNR		Message size (Byte)	Cover size (Pixels)	Reference No.
Proposed algorithm	Research	Proposed algorithm	Research	Proposed algorithm	Research			
1	NA	0.0019	NA	75	68	1000		
1	NA	0.0096	NA	68	65	5000		
1	NA	0.0189	NA	65	59	10000	512x512	[1]
1	NA	0.0343	NA	62	51	18000		
0.999	NA	0.068	NA	59	49	36000		
1	NA	0.0110	0.56	67	62	1400	256x256	[6]
1	NA	0.013	0.0199	67	65.14	850		
1	NA	0.016	0.0286	66	63.56	1050	512x512	[7]
1	NA	0.023	0.0298	64.5	63.38	1500		
1	NA	0.0156	0.0267	66.19	63.8619	8192		
0.999	NA	0.0311	0.054	63.20	60.8053	16384		
0.999	NA	0.625	0.1034	60.16	57.985	32768	512x512	[8]
0.999	NA	0.1250	0.2064	57.16	54.9833	65536		
0.999	NA	0.2498	0.411	54.15	51.9925	131072		
0.999	NA	1.2289	0.8169	47.23	49.0092	262144		



	1K	10K	50K	100K
🟦 MSE	0.006025	0.062975	0.31625	0.6295
🟥 PSNR	72.7975	62.7175	55.7425	52.7325
🟩 BER	9.9E-05	0.00098	0.0005	0.0099
🟪 SSIM	1	0.9996	0.998	0.996

Fig. 4: Hiding results chart in cover with dimensions 1800*2800

9. CONCLUSIONS

A number of conclusions can be summarized:

• Develop and improve the hiding algorithm currently used in lightweight systems (LSB algorithm) thus

creating an effective new hiding algorithm and achieving good results in lightweight systems.

• Using coding for a secret message instead of cipher, saving a lot of time and effort from processing time and increase the randomness of the coded message.

• The actual bits of the secret message do not correspond to any of the pixel bits in the cover image, thus hiding the relationship between the secret message bits and the less Significant bits in the cover image in the proposed coding method.

• Using the Header at the beginning of the secret message gave the user more ease and provided him with the need to know or understand the algorithm used, and also eliminated the need for a direct security communication between the two parties to share that information.

• The proposed algorithm has achieved very high processing speed in both hiding and retrieval and with files of large sizes. It is a valuable achievement for lightweight systems that work on smartphones, with an execution time of 0.13 per second with large files (5,040,000 pixel cover, 100K secret message).

• The algorithm was applied and the results were collected on different phones in terms of processing capabilities and physical characteristics and the results were very close which indicates that the algorithm can work on most phones and does not need specific features or specific phones.

• The efficiency of the proposed algorithm can be observed given the results obtained from the Values of MSE, PSNR and SSIM, as the amount of change obtained in the cover image is very limited, which means that the suspicion of hiding within the image is removed by the intruders.

10. FUTURE WORKS

A number of future ideas can be done:

• The message in the proposed system consists of a file (image, text, voice). But in fact all three types are treated as raw data by converting them into a torrent of bits. Therefore, in the future, the system can be easily expanded to include more types of files by expanding the file type field at the front of the hidden message to contain more bits.

• The proposed cover in the search is the color image, but the proposed system can be applied to other multimedia covers such as videos, which consist of a set of frames, providing a larger size for hiding.

• Use a method of compressing data (lossless) as a primary processing of message data for the purpose of reducing volume, thereby reducing the amount of changes to the cover image.

• Adding the proposed system to private Source Code of Android operating system and examine the amount of improvement in output which became operational as part of the operating system instead of being a reliable application.

11. REFERENCES

[1] J. Thanikkal, M. Danish, S. Sarwar, 2014, "A New Android Based Steganography Application for Smartphone’s", Journal of Basic and Applied Engineering Research, Vol. 1, No. 8, PP. 32-35.

[2] A. Kadam, and S. Joshi, 2015, "Pervasive and Secure M-Banking using Steganography", International Journal of Emerging Engineering Research and Technology, Vol. 3, No. 16, PP 31-37.

[3] D. Bucerzan, C. Ratiu, and M. Manolescu, 2013, "SmartSteg: A New Android Based Steganography Application", Int J Comput Commun, Vol. 8, No. 5, PP. 681-688.

[4] F. Djebbar, and N. Abu-Ali, 2017, "Lightweight Noise Resilient Steganography Scheme for Internet of Things", IEEE Global Communications Conference, Singapore, PP. 1-6.

[5] M. Mahmood, 2017, "WebP Image Steganography Using M8PAM for Android Applications", Master Degree Thesis for Computer Science in Al-Nahrain University.

[6] Z. Abdalrdha, and et.al, 2015, "LBS Steganography Method on Mobile Image Based on NTRU Algorithm", Journal of College of Education, Vol. 1 No. 1, PP. 149-170.

[7] R. Apau, and C. Adomako, 2017, "Design of Image Steganography based on RSA Algorithm and LSB Insertion for Android Smartphones", International Journal of Computer Applications, Vol. 164, No. 1, PP. 13 – 22

[8] M. Kasapbas, and W. Elmasry, 2018, "New LSB-based Color image steganography method to enhance the efficiency in payload capacity, security and integrity check", Indian Academy of Sciences, Vol. 43, No. 68, PP. 1-14.

[9] S. Mittal and V. Mattela, 2018, "A Survey of Techniques for Improving Efficiency of Mobile Web Browsing", Concurrency and Computation Practice and Experience, Vol.31, No. 5, PP.1-31.

[10] P. Svensson, 2013," Smartphones now outsell 'dumb' phones".

[11] A. Amoroso, and M. Masotti, 2006, "Lightweight Steganography on Smartphones", 3rd IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, PP. 1158-1162.

[12] A. Zaharis, A. Martini, and et.al, 2011, "Lightweight Steganalysis Based on Image Reconstruction and Lead Digit Distribution Analysis", International Journal of Digital Crime and Forensics, Vol. 3, No. 4, PP. 29-41.

[13] S. Sivakumar, and B. Rajesh, 2014," Steganography on Android Based Smart Phones", International Journal of Computer Science and Mobile Computing, Vol. 3, No. 5, PP.1051 – 1054.

[14] H. LEE, 2018, "Data Hiding in Spatial Color Images on Smartphones by Adaptive R-G-B LSB Replacement", IEICE TRANS. INF. & SYST., Vol.E101–D, No.8, PP. 2163-2167.

[15] D. Piens, N. Staffa, 2014, "Android-Based Digital Image Steganography and Steganalysis", Fun Facts About Simon, Time article.

[16] D. Selvaraj, 2014, "Development of A Secure Communication System Based On Steganography for Mobile Devices ", Master Degree Thesis for High Integrity Systems Frankfurt University of Applied Sciences.

[17] S. Ahmed, and et.al, 2017, "Combining Steganography and Cryptography on Android Platform to Achive a High-Level Security", Journal of Engineering and Applied Sciences, Vol.12, No. 17, PP. 4448-445