Computer Science and Engineering Department
Indian Institute of Technology, Delhi

Major Project-1 Report

IMAGE FORGERY DETECTION SYSTEM

Submitted to.

Prof. Vireshwar Kumar

Department Of Computer Science & Engineering, IITD

Submitted by

Shiva (2021MCS2149)

OUTLINES

1. Abstract
2. Introduction
3. Methodology
4. Experiment results
5. Conclusion

ABSTRACT

The image forgery detection technique involves analyzing the image for any tempering and inconsistencies, which tells whether the image is tempered or not. If a small portion (e.g., a small dot in the image) of the image has been edited or tempered, it is more difficult to detect than a large portion of a tempered image. The implemented technique is used to detect forgeries in the images. It is capable to detect any small point of tempering in the image. It generates a hash from all pixels bits except the bits which will be used in embedding. The Secure hash algorithm (sha256, sha384, and sha512) is used for hash generation. It uses two algorithms ECDSA and RSA of different key lengths (256, 320, 384, 1024, and 2048) for signing image information to make a digital signature. Least-significant-bit substitution mechanism is used to embed the signature and corresponding public key in the designated pixels. The embedding of the digital signature and public key in the designated pixels remains completely undetectable to the human visual system. We developed a platform at which we check whether there is any forgery or not in the image. We will be evaluating the performance of the algorithms by using the metrics which contain various times including embedding time, extraction time, verification time, etc. Through the results, we analyzed that ECDSA is taking less time than RSA.

INTRODUCTION

Images are used for preserving and sharing important events of life. Images are also used as solid proof of criminal activities and they can be used in court as legal evidence. Images can be modified or manipulated and the genuineness of the image can be lost due to these tempering. There are many such tools (i.e., Photoshop) available through which we can change the content of the image. For this, we have to first process the image to know whether it is forged or not.

Image forgery detection has grown exponentially over the past years and it is a new active area of research. The technique involves analyzing the image for any tempering and inconsistencies, which tells whether the image is tempered or not. If a small portion (e.g., a small dot in the image) of the image has been edited or tempered, it is more difficult to detect than a large portion of a tempered image. The forgery detection technique is one of the authentication methods which works on the presumption that the original image contains some inherent patterns introduced by the various imaging devices or processing. These patterns remain consistent in the original image and are altered after some forgery operations. The detection of forgeries in digital images has become complex due to advanced and sophisticated processing tools. So, various techniques and algorithms have been developed to detect the forgeries in the images. There are broadly two categories for Digital Image forgery:

Active/intrusive/non-blind method: There are mainly two examples of active forgery methods, they are digital signature and watermarking. These methods have been introduced to authenticate the contents of digital images. These methods require certain digital information to be embedded along with the original image, such as signature generation and watermark embedding at the time of the creation of the images.

Passive/non-intrusive/blind method: The original image contains some inherent patterns, which were introduced by various imaging devices or processing. These patterns are always remained consistent in the original image and are altered after some tampering operations have been carried out on the images. The original image is said to be forged when the patterns of this image become disrupted. Comparing it with the prior active methods, this method does not need any extra information such as a watermark or signature. Examples are copy-move, image splicing, Re-sampling (resize, rotate, etc), and compression.

METHODOLOGY

Our implementation includes Two major steps: -

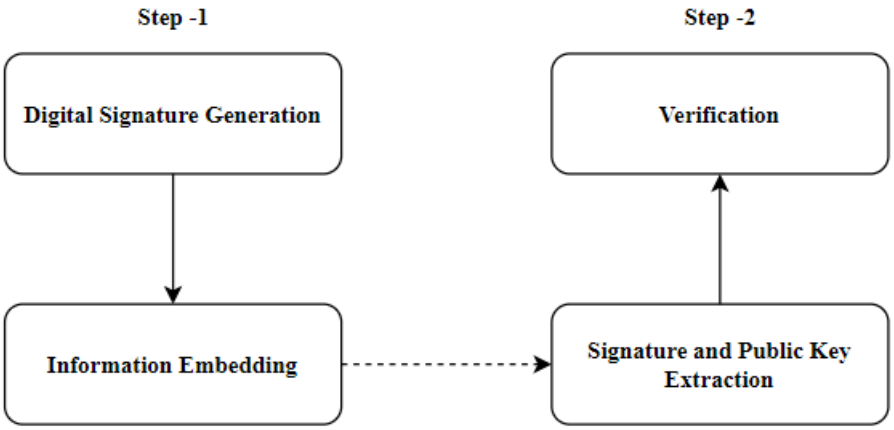

Fig -1 Flow diagram

Step -1 –

I. **Digital Signature Generation –** To calculate the digital signature of the image We have generated the hash by the use of secure hash algorithms (SHA256, SHA384, and SHA512). Then We encrypted this hash using generated private key to generate a digital signature.

How I computed Hash – We used here the least significant bit from a colour, which means 3 bits from a pixel. That's why we have taken 7 bits from each colour of those pixels which will be used in embedding. Other than these, we have taken the whole pixel and finally passed Image Info. string to SHA to generate a hash of the image.

II. **Information Embedding –** We have used the LSB substitution mechanism in our implementation. First, we embedded the generated public key in LSB bits, and after that the generated signature.

suppose we have a digital signature and public key of lengths 384 and 384. This means we have to embed 768 bits into our image and we are using here LSB substitution So we will embed 1 bit of generated public key into the least significant bit of colour and similarly we embedded the digital signature.

768 / 3 = 256 pixels will be used in embedding.

Given below are the images of the before and after embedding. The red box in the embedded image is showing the embedding information in the image.

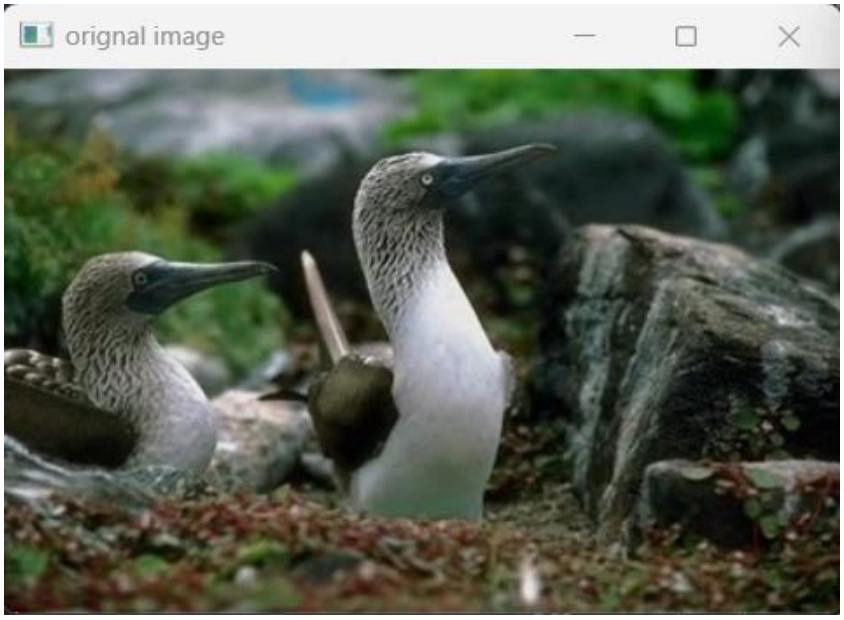

Fig – 2 Original Image

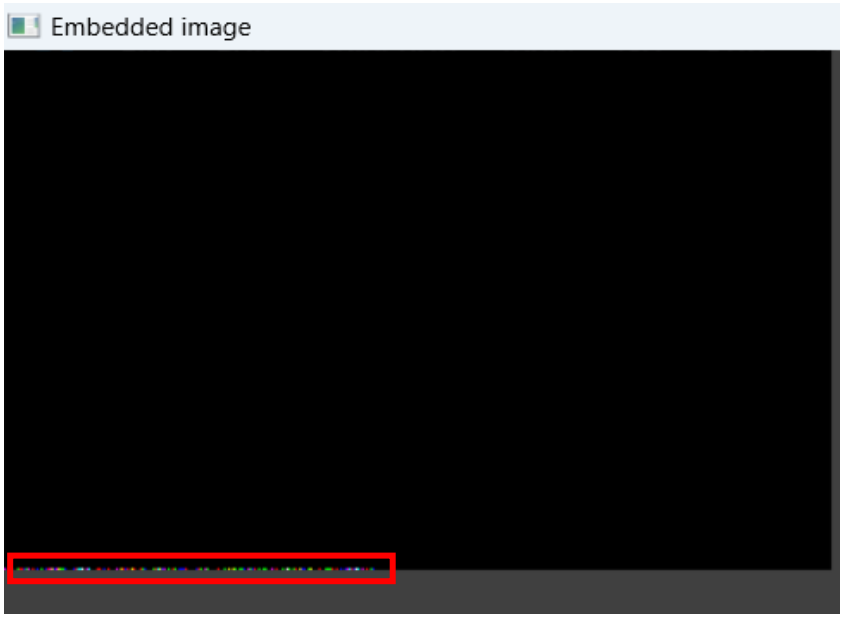

Fig – 3 Embedded Image

Given below is the diagram of the whole process of step-1 (Embedding phase)-

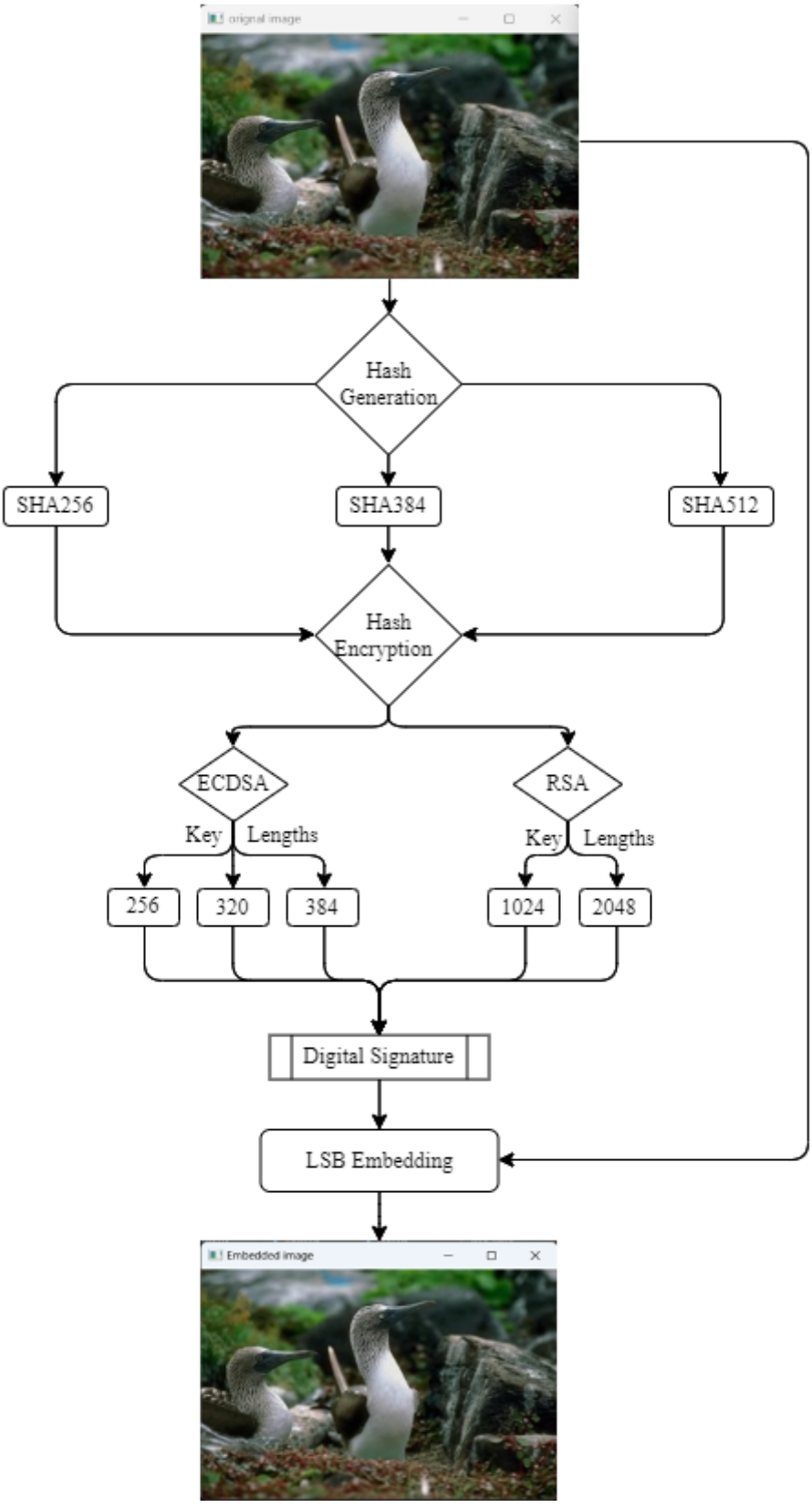

Fig – 4 Information Embedding flow diagram

Step -2 –

I. **Public key and Signature Extraction –** First we extracted the public key which is embedded in the image. Then we extracted the embedded digital signature from the image.

II. **Verification –** In this step, we checked and detected any possible forgery in the image. We generated a hash from the decryption of the digital signature (which we extracted above) using the extracted public key. Then we computed the hash from the whole image (except the embedded bits) and matched both the hash to authenticate the image.

We can see the whole working of step-2 (Verification phase)-

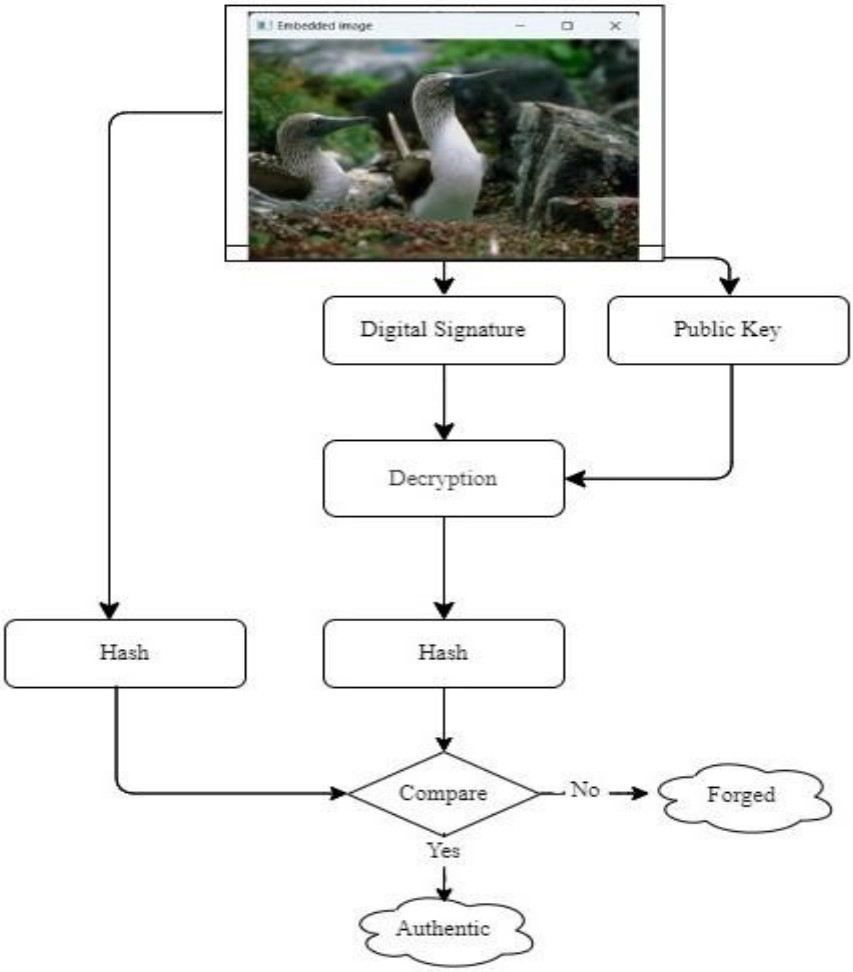

Fig – 5 Verification Phase flow diagram

EXPERIMENTAL RESULTS

There are two phases of our implementation first we embedded the image then we verified whether the embedded is forged or not. We have used RSA and ECDSA for key generation and the different lengths of keys (i.e., 256, 320, 384, 1024, 2048) and 3 hash generation algorithms (i.e., sha256, sha384, sha512).

system Specification: Processor- i5 10 gen, Operating system-Windows, RAM- 8GB

Table 1. (ECDSA)

Table 1 represents the time taken by the program for the ECDSA algorithm. There is the time taken by the signature generation, signature embedding, and total time taken for the embedding. Signature & public key extraction time, verification time, and total time taken by the program to authenticate the image.

Key Length (In bits)	SHA	Signature generation Time (ms)	Signature embedding Time (ms)	Total Embedding time (ms)	Sig and PU key Extraction Time (ms)	Verification Time (ms)	Authentication Time (ms)
256	256	126.073	1.575	127.648	1.586	137.317	138.903
256	384	128.147	1.475	129.622	2.085	207.421	209.506
256	512	125.568	1.487	127.055	1.275	130.686	131.961
320	256	138.073	2.110	140.183	1.645	124.002	125.647
320	384	126.946	1.844	128.790	1.694	134.57	136.264
320	512	146.358	2.036	148.394	1.813	151.433	153.246
384	256	150.800	2.567	153.367	2.081	139.031	141.112
384	384	142.973	2.431	145.404	1.972	132.631	134.603
384	512	183.445	2.659	186.004	2.182	139.290	141.472

Table 2. (RSA)

Table 2 represents the time taken by the program for the RSA algorithm. The columns for times are the same as above table.

Key Length (In bits)	SHA	Signature generation Time (ms)	Signature embedding Time (ms)	Total Embedding time (ms)	Sig and PU key Extraction Time (ms)	Verification Time (ms)	Authentication Time (ms)
1024	256	145.276	5.900	151.176	6.416	152.114	158.53
1024	384	145.048	4.838	149.886	5.552	140.749	146.301
1024	512	159.476	6.503	165.979	5.466	150.499	155.965
2048	256	162.338	10.657	172.995	10.506	150.643	161.149
2048	384	155.958	11.149	167.107	11.652	153.925	165.577
2048	512	174.846	11.978	186.824	11.383	154.513	165.896

Image Forgery Detection Platform-

We have implemented a platform for verification on which we can upload an embedded image and it tells whether the image is authentic or not. If the uploaded image is authentic then it will show verification successful. If the image is tempered or forged then it will show verification unsuccessful. We can see the platform (Image Forgery Detection System) in the given below Figure, we simply choose the image from the system and we upload it and it authenticates the image.

CONCLUSION

The implemented technique is an efficient technique to detect forgeries in the images in very less time. It can detect any small point of tempering in the image. By the use of the Least-significant-bit substitution for embedding the embedded information is completely undetectable to the human visual system. By the ECDSA key for digital signature generation, this technique is taking a fraction of the time for embedding. While RSA is taking more time as compared to the ECDSA. In conclusion, this technique is an efficient technique for image forgery detection and authentication.

REFERENCES

1. Sahib Khan and Arslan Ali, "CLIFD: A novel image forgery detection technique using digital signatures", Journal of Engg. Research Vol. 9 No. (1) March 2021 pp. 168-175.
2. S. Bayram, I. Avcibas, B. Sankur, and N. Memon, "Image manipulation detection," J. Electron. Imaging, vol. 15, no. 4, p. 41102, 2006.
3. Birajdar, G. K., & Mankar, V. H. (2013). Digital image forgery detection using passive techniques: A survey. Digital investigation, 10(3), 226-245.
4. Thakur, Tulsi, Kavita Singh, and Arun Yadav." Blind Approach for Digital Image Forgery Detection." International Journal of Computer Applications 975 (2018): 8887
5. J.Fridrich, D.Soukal, and J.Luka, "Detection of copy-move forgery in digital images", *in Digital Forensic Research Workshop,* pp.6-8,2003
6. Y.-F. Hsu and S.-F. Chang, "Image splicing detection using camera response function consistency and automatic segmentation," in Proc. Int. Conf. Multimedia and Expo, Beijing, China, 2007.
7. J. Lukás, J. Fridrich, and M. Goljan, "Digital camera identification from sensor noise," IEEE Trans. Inform. Forensics Security, vol. 1, no. 2, pp. 205–214, 2006.