SE-510 SE Project
Report

Secure Image Comparison with Hashing and Signature

Submitted by

Roll No	Name of Students
MT2015004	Aditi Raghuvanshi
MT2015007	Akshay Jain
MT2015046	Mahendra Garodi
MT2015073	Paresh Pankhaniya
MT2015083	Priya Sancheti

Under the guidance of
Dr. Shrisha Rao

International Institute of Information Technology, Bangalore

Summer - 2016

Acknowledgement

We would like to thank our supervisor Prof. Shrisha Rao for all his help and guidance. He was always available whenever we needed some advice or ran into any problem. We would also take the opportunity to thank the God and our parents for their blessings and our beloved friends for their encouragement and motivation during this period.

Aditi Raghuvanshi
Akshay Jain
Mahendra Garodi
Paresh Pankhaniya
Priya Sancheti

Abstract

Due to increasing digitalization, the authentication of multimedia content is becoming more and more important. So-called perceptual hash functions have been proposed to establish the perceptual equality of multimedia content. Perceptual image hash functions produce hash values based on the images visual appearance. Such a function calculates similar hash values for similar images, whereas for dissimilar images dissimilar hash values are calculated. Finally, using an adequate distance or similarity function to compare two perceptual hash values, it can be decided whether two images are perceptually different or not. Perceptual image hash functions can be used e.g. for the identification or integrity verification of images. Secure Image Hashing is done by using Blockhash algorithm. We use GnuPG library for signing of generated hash and hence authenticating it's identity.

Github Link

`https://github.com/Secure-Hash/blockhash`

Contents

1 Introduction **1**

2 System Architecture **2**

3 Hash generation with sign **4**
3.1 Image Preprocessing and color extraction 4
3.2 Quantization of image 4
3.3 Hash generation 6
3.4 Sign hash file 6

4 Comparing hash **7**

5 Test cases and Results **8**

6 Future Work and Conclusion **9**

References **10**

1 Introduction

In day to day life we need to compare different objects. When we need to compare two different files or objects in computer, we need to use correct method. When authenticating an executable file, it is important that every single bit exactly matches the original executable. Cryptographic hash functions are adequate for such tasks. A multimedia object,e.g. an image, can have different digital representations that all look the same to the human perception. Different digital representations can emerge from an image through image processing steps like scaling,rotating etc. Each of these image processing steps changes the binary representation of the image. Using a cryptographic hash function to authenticate the modified images therefore does not work.

So-called perceptual hash functions have been proposed to establish the perceptual equality of multimedia content.Perceptual hash functions extract certain features from multimedia content and calculate a hash value based on these features. When authenticating a multimedia object the hash values of the original object and the object to be authenticated are compared using specific functions. Such functions calculate a distance or similarity score between two perceptual hash values. The final verdict is based on a chosen threshold. The purpose of this report is to discuss the Secure Image Hash using blockHash algorithm.

A system is created where two images can be compared to see if they are identical or not, given that the images themselves may be private and not subject to open exchange and viewing. Standard algorithms for image hashing and digital signatures are used.

The system is robust enough to take into account attacks on a given input and yet be flexible to give correct output. Attacks included in the system are rotation, skew, scaling and random removal of pixel of images. After hash is generated we use Digital Signature to authenticate its identity.

2 System Architecture

Secure Image Comparison is blockhash based system implemented to compare similarity between two images without exposing image to third party. This system consists of two main modules.

Below is hash generator module explained.

- System takes an input from user in the form of image.
- This image gets preprocessed and then sent it to hash generator module.
- Hash generator module will compute hash of image based on color values of image.
- This generated hash value will be signed by user of system.

Hash comparator module takes input as two digitally signed hash files.

- Validate both the files for authenticity, using GnuPG library.
- After authenticating two files, system will detach hash value from signed copy.
- System will verify whether both file contains hash values with same number of bits or different.
- This value will be then compared with second file
- Based on number of matching bits in hash size, and threshold set, application will give appropriate messages.

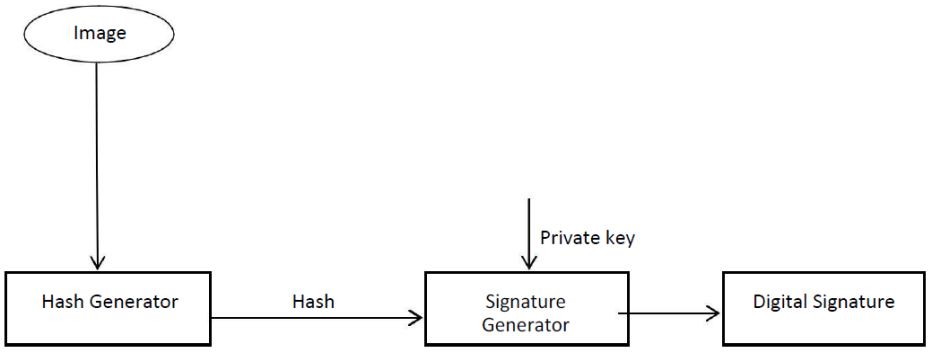

Figure 1: Hash and Signature Generator

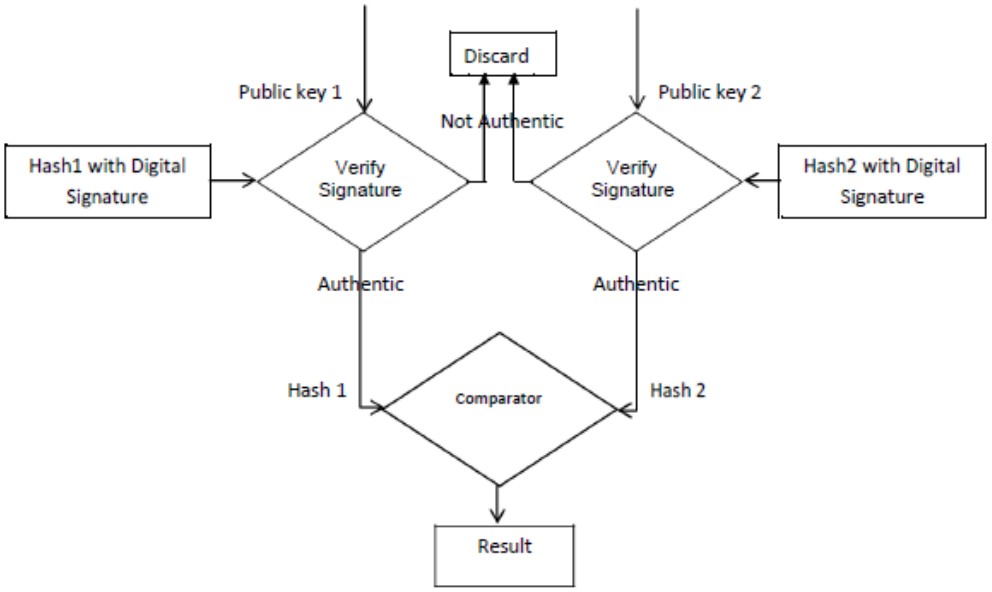

Figure 2: Comparator Module for comparison

3 Hash generation with sign

Below are the steps required to generate hash from given image

3.1 Image Preprocessing and color extraction

Secure image comparison application takes image as input from the user. This system accepts images in the format of .jpg, .jpeg and .png. Application reads the image file provided by user. As different image formats are encoded differently, we need to make sure that it should be converted to some standard encoding format and then generate its hash value. In secure image comparison system, we are using RGB as standard encoding format. At the start of preprocessing, application will read file from disk which is specified by user. After that it will remove different color profiles if any present in the image. It then converts image into standard RGBA format. As there might be some images which does not have alpha channel present, so we need to convert this RGBA to RGB format so that all the images will be of same format.

After image preprocessing is done, application will extract all the pixels color information in buffer. This buffer information is of Quantum format specified by imagemagick library. For every pixel from image, we will have 3 values ranging from 0-65,535 corresponding to each color Red, Green and Blue. Application internally stores this information in the buffer of Quantum. To extract Quantum values from image we have used imagemagick library.

3.2 Quantization of image

: In the next step of hash generation process, application will logically group all these pixels into blocks. Every block consist number of pixels. More number of blocks give better results. If we consider every pixel as block then hash size will increase, also it will increase hash computation time. As we try to increase number of blocks perceptual hashing tends to behave like cryptographic hash. Even if the images are perceptually same we will be end up getting matching below 50%. For that reason we need to find out optimal value of number of blocks. Here we are considering optimal value as 1024. Application also provides option for user to choose different number of blocks. Number of blocks in image decides number of bits present in hash value. Suppose user wants to generate hash value of N^2 bits where N

is an integer, then the number of blocks formed will be $N * N$. Application provides facility to choose hash size. All the logical blocks will of euqal size. Block width and block height will be calculated by using below formula.

$$Block\_Height = \frac{Image\_Height}{\sqrt{Hash\_Size}}$$

$$Block\_Width = \frac{Image\_Width}{\sqrt{Hash\_Size}}$$

Figure 1: Quantization of image

3.3 Hash generation

By using blockhash algorithm, system will assign values to individual block. The value assigned in first phase of process will be sum of all the color values. For further processing all the blocks will be logically grouped into 4 bands. All the bands will get sorted by using qsort to find the median of bands. For each individual band, the median value gets computed. In the second phase of algorithm, blockhash will assign value to original blocks and the value assigned will be 0 or 1. System compares each block against median value of the band to which this block belongs. If block value is less than the median value then 0 gets assigned to block, otherwise 1. After processing all bands, we will get stream of 0's and 1's. Application will then convert this blockhash values to hexadecimal representation. To accommodate rotating upto 90 degrees, application calculates hash value of image with different rotations. In this repeated hash value computation, we will create duplicate of preprocessed image after first phase. This duplicate image will then used for next hash calculation. Here system will generate total 19 streams each corresponds to one rotation value form {0, 5, 10, 15,...., 90} degree.

Algorithm 1 Blockhash

1: **procedure** ComputeHash
2: Let N denote the bit length (e.g. 256 bit) of the final hash value.
3: Divide the pixels of the image I into non-overlapped blocks $I_1, I_2, ..., I_N$.
4: Calculate the mean of the pixel values of each block. Let $M_1, M_2, ..., M_N$ are mean values corresponding to $I_1, I_2, ..., I_N$.
5: Obtain the median value M_d of the mean value sequence.
6: Normalize the mean value sequence into a binary form and obtain the hash value h as,

$$h(i) = 0 \, if \, M_i < M_d, \, otherwise \, 1$$

3.4 Sign hash file

Hash generated in above process will get stored in temporary location in a file. This file will be then signed by private key of user. Signing module internally uses GnuPG library to sign file. System signs hash file in attached mode. In attached mode signature and file will remains in same file. The generated new file will be then used get stored on path specified by user. This file can

be used for image comparison purpose by using same application. To sign file its assumed that system is already having GnuPG installed and properly configured.

4 Comparing hash

Hash comparison is another module in secure image comparison system. This module gives interface to compare two images by using hash files generated using hash generator module of this system. Hash comparison has below steps.

1. **Validate signature** : Hash comparison module takes to hash files which are generated by same application, might be by same user or different users. System reads to files and by using GnuPG, it verifies whether the files are signed or not. It also checks whether the sign is valid or not. If sign is invalid then files are discarded. If both the files are having valid sign then it will extract hash from both the files. This extracted hash will be stored in two temporary files for comparison purpose.

2. **Comparing hash** : The extracted hash will be then compared using hamming distance. Before staring comparison between two hash values, we need to make sure that hash values should have same size. If hash values are of different size then system discards both the files by giving appropriate error message. Each hash file will have 19 streams of hash values. Each stream from one hash file will get compared with all streams of other file. Streams will be compared bit by bit. Matching criteria will be maximum matching.

5 Test cases and Results

We have tested secure image comparison tool with different images and below are the results of same.

Table 1: Test Case Description

Sr. No.	Test Case	Result
1	Hash generation and comparison - JPEG/JPG image	PASS
2	Hash generation - PNG image	PASS
3	Hash size - Generation and verification with 64, 256 and 1024 bit hash	PASS
4	Rotation - 5 degree	PASS - Match Score 100%
5	Rotation - 45 degree	PASS - Match Score 100%
6	Rotation - 45 and 75 degree	PASS - Match Score 100%
7	Rotation - 53 degree	PASS - Match Score 84%
8	Scaling - by 0.5	PASS - Match Score 100%
9	Scaling - by 1.25	PASS - Match Score 100%
10	Scaling - by 2.0	PASS - Match Score 100%
11	Gausion Noise - 1.0	PASS - Match Score 100%
12	Totally two different images	PASS - below 50%
13	Scaling -,First image 1.25, second image to 0.75	PASS - Match Score 87%
14	Combinational - Scaling 0.5, rotation 53 degree, Noise -1 .0	PASS - Match score 84%

6 Future Work and Conclusion

1. **Document Comparison:** This system can be enhance for comparing two documents. If any modification is done in document ,system should be able to detect that.

2. **Improvement in Manually Distorted Image Result:** By taking overlapping blocks efficiency of system can be improved.

Conclusion
A block mean value based perceptual image hash function,was implemented in C/C++. The implementation was integrated into pHash.Now, we could digitally sign the hash generated and send it to the other party so that image comparison can be done securely.

References

[1] Yang, B., Gu, F., and Niu, X.: Block mean value based image perceptual hashing. In Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Multimedia Signal Processing (IIH-MSP),pp. 167172. IEEE, 2006, ISBN 0-7695-2745-0.

[2] Zauner, Christoph: Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis,Upper Austria University of Applied Science Hagenberg Campus, 2010.

[3] ImageMagick Libraries[Online]
Available: `http://www.imagemagick.org/script/index.php/`

[4] Digital Signature Generation and Verification[Online],
Available: `https://www.gnupg.org//`

[5] QTFrameWork Configuration[Online]
Available: `https://www.qt.io//`