Authenticity Management Algorithm for Digital Images

Todor Balabanov, Peter Dojnow, Vasil Kolev, Nikolai Manev, Walter Mudzimbabwe, Petar Tomov, Ilian Zankinski, Stela Zhelezova

Abstract

Nowadays network services are gaining great attention. Therefore authentication of media content is very important. For the considered problem, the group propose an algorithm with client-server architecture. The chosen digital signature algorithm is based on image content and is according to NIST specifications. The most important parameters for an effective tampering detection are discussed.

Key words: authentication, watermarking, tampering detection

1. Introduction

Image authentication is of great importance due to the large number of multimedia applications in various fields. Currently, the amount of digital images transmitted over non-secure channels is growing rapidly. Therefore, the protection of image integrity is of great interest.

A digital watermark is called robust if it resists a designated class of transformations and it is imperceptible if the watermarked content is perceptually equivalent to the original one.

There are active and passive image authentication methods. The active methods extract some information of the image to be authenticated while the passive ones perform the authentication without needing previous information about the authenticated image. Active methods include watermarking and hashing based methods. We extract the needed authentication and tampering detection information from the original picture so we use active methods for the proposed algorithm.

With respect to the embedding method there are different watermarking techniques. Some of them embed message in spatial domain. Other methods use message embedding in a frequency-based representation of a digital image. The proposed authentication schema uses spatial domain.

Because of the particular case in which the watermarking is considered, we employ steganographic technique to embed data. Least Significant Bit (LSB) hiding is one of today's easiest techniques for image steganography. It imply adding some secret information in the least significant bits of the image pixel. The image quality is distorted for the number of bits embedded in a pixel greater than 3. However, such a technique is very insecure because the watermark can be easily destroyed. But for our problem, the most desirable property is imperceptibility to human senses than robustness therefore it is appropriate. The creation of robust and in the same time imperceptible watermarks has proven to be quite challenging [3].

In the case of mobile devices the signature scheme must be efficient enough without delays. We choose to develop a simple image authentication scheme.

1.1. Definition of the problem

There is an image taken in an Android application. The application have to sign digitally the image without the user's knowledge in order to verify its origin. Apart from this, a digital watermark has to be added such that in case of tampering the modified part of the image is indicated. The application sends the image to the server where a corresponding algorithm part verifies the origin and the integrity of the image.

1.2. Historical Review

A Vector Quantization (VQ) based digital image watermarking scheme is proposed in [4]. The codewords in the VQ codebook are classified into different groups according to different characteristics and then each binary watermark bit is embedded into the selected VQ encoded block. This technique cannot resist geometrical distortion.

There are algorithms based on the discrete Radon transform [5]. The Global Hash estimation of the image is used to establish if the image under analysis was tampered and if this is the case, the local Hash estimation is used for obtaining the exact tampered regions. This algorithm is not suitable for our problem because of the requirement of sending the original Hash values apart from the watermarked image.

Some steganographics techniques also are suitable for hiding an information in image. A steganographic scheme is applied if the user has a secret message that is to be hidden in the image. The authors [7] use a hash function to find out the locations to store the secret message which is encrypted using AES encryption. In the problem that we consider, the secret message has to be the private key of the application. This scheme cannot work if the picture is tampered.

1.3. The Algorithm Requirements

The performance of the proposed algorithm have to meet the following main properties:

- Robustness: It may not be possible without knowledge of the procedure and the secret key to manipulate the watermark. Robustness also means the resistance ability of the watermark information to changes and modifications made to the original file.
- Nonperceptibility: It is important to recognize whether the brought bit sample of the watermark produces perceptible changes optically. A perfect nonperceptible bit sample is present if data material marked with watermark and the original cannot be distinguished from each other.
- Blind: The detection of digital watermark has to be done without the original data, so in order to detect watermark information, blind techniques have to be used.
- Effectiveness: Tampering detection and localization of the changes made to a watermarked image.

2. Our Approach

1.1. Initial statements

The entire algorithm consists of two main steps:

- client part in which the embedding is done;
- server part where the verification and extracting is done.

The initial restrictions are:

- Size of the original image $n \times m$ pixels in Bitmap. Minimal size: 1024×768. This is the worst case, because the hidden bits place is proportional to the image size.
- Content: interior and car pictures. The content is important in the connection with the hiding bits methods. If the image content is almost the same, i.e. the colors are evenly distributed hiding of information is much more difficult.
- The value of perceptual transparency is measured with Signal-to-noise ratio (SNR). SNR is used in science and engineering to compare the level of a desired signal to the level of background noise. It is defined as the ratio of signal power to the noise power:

$$SNR = 10.log(MAX^2/MSE)$$

MAX - the expected pixel values,
MSE - the standard deviation of the pixel values between the original image (OI) and the watermarked image (WI) bits.

$$MSE = \frac{1}{m.n}\sum_i\sum_j (O_{i,j} - W_{i,j})^2,$$

where $i \in \{1, ..., n\}; j \in \{1, ..., m\}$ are the dimensions of the original picture.

For the considered problem it is required $SNR > 30$.

The most common way to model color images in Computer Graphics is the RGB color model - each pixel is represented by tree values, the amount of red, green and blue. We can use a packed ordering - the tree color components are placed together in a single array element. Each color pixel is presented by a 32-bit value according to Fig.3. Eight bits are used to represent each of the RGB components and 8 bits are reserved for the transparency (α) component. The main input element in our algorithm is the original image (OI). It is an $n \times m$ matrix C, $c_i = (R_i, G_i, B_i)$. Each element of the matrix is $0 \leq R_i, G_i, B_i \leq C_{max}$, for the most digital images $C_{max} = 255$. Because there are three color channels, it's possible to store three hidden bits of information in each pixel. But our proposition is to use only one of the color channels because the application does not need more capacity.

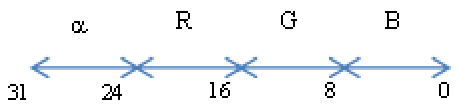

Figure 1: The bits of a color pixel

In order to be able to create a digital signature, we need a private key. For the considered problem, the user must not be involved in the signing procedure. The algorithm aim is to verify that the image is taken with certain device. Its corresponding public key is needed in order to verify the authenticity of the signature. The key pair (private and corresponding public key) generation is done in two parts. The choice of algorithm parameters and the key pair generation for particular image and user. The algorithm parameters are: q - N-bit prime, p -

L-bit prime, $p-1$ is a multiple of q, g - its multiplicative order modulo p is q. The NIST recommend in Federal Information Processing Standard (FIPS) 186-3 key length pairs (2048, 256), and (3072, 256) for (L, N) - the associated pair of length parameters for a DSA. The algorithm parameters can be used as they are implemented in library according to some provider (as SUN) for instance. Also the `DSAParameterSpec` class can be used to chose specific (p,q,g) parameters.

A particular key pair is generated by `KeyPairGenerator` class. At this step the `String algorithm` (we choose DSA) have to be specified. To initialize two arguments are needed: the key length L_k and the source of randomness. For the second parameter the `threadedSeedGenerator` have to be used several times with different parameters, each of them connected with the particular user, device and time parameters.

So the next input element the algorithm needed is the private key k with length $L_k = 2048$ bits, k - random, $0 < k < q$ and the public key $y = g^k \ mod \ p$.

1.2. The client part

Run-up the image

From considered `OI` matrix C,k it is needed to obtain a matrix with zeroes `ZBI` in each bit which we will use later for hiding information. We chose LSB and 3-LSB hiding strategy to achieve the highest possible robustness without degrading image quality.

Next from `ZBI` the sequence is obtained. The order in which pixels are taken is important due to the avalanche effect in hash functions. This can be done in connection with private key. On the server there can be stored different `PATH` schemes for each key pair. A scheme is actually a permutation of $n.m$ variables. All permutation are considered in lexicographical order and each has its consecutive number therefore in variable `PATH` the corresponding permutation number is saved. So the function `MakeSequence` takes as input `ZBI` and `PATH` and returns the "zerobits" image sequence $\texttt{ZBI}_s$.

Signing

Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value. The standard Message Digest algorithms are for example MD5, SHA-1, SHA-256 etc. Bear in mind that the first two are compromised [3], [8] for the implementation of the proposed algorithm we suggest SHA3-256, SHA3 is the newest one but it has BITE-only implementation in Java so the length of input sequence has to be divisible by 8. It takes as input $\texttt{ZBI}_s$ and gives as output $|MD| = 256$ bits. A common way to sign things is the

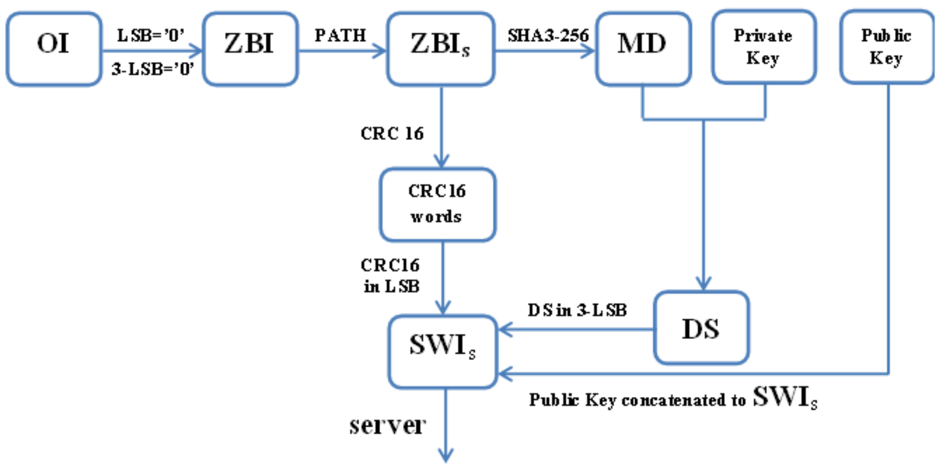

Figure 2: The algorithm client part

Digital Signature Algorithm. The signing is done in `Signature` object from the `Signature` class with the following steps:

- Get a `Signature` object - the signature algorithm name is specified and the the name of the message digest algorithm used by the signature algorithm is specified. `SHA3-256withDSA` is a way of specifying the DSA signature algorithm, using the SHA3-256 message digest algorithm (Spongy Castle Crypto package).

- Initialize the `Signature` object - the private key is used.

- Supply the `Signature` object the $\texttt{ZBI}_s$ to be signed - the *update* method of the `Signature` object is called.

- Generate the digital signature sequence $DS = (r \| s)$ applying *sign* method, where $r = (g^{k_i} \ mod \ p) \ mod \ q$ and $s = k_i^{-1}(\text{SHA3-256}(\texttt{ZBI}_s) + k.r) \ mod \ q$, k_i - a random per-image value and L_{DS} is about 2×160 bits but it depends on Hash function and converting of integers r and s into strings.

Now the DS and public key are ready to be sent to the server. Usually they are stored in additional files and sent separately. The proposed algorithm embed DC in the $\texttt{ZBI}_s$. We store DS not in each 3-LSB but in each $(n.m) \ mod \ L_{DS}$ 3-LSB according to `PATH` permutation. The public key is added at the end of $\texttt{SWI}_s$.

Watermarking for tampering detection

For the algorithm purpose, the non visible watermark have to be hidden eventually in the $\texttt{ZBI}_s$. We use wide spread blocking technique. The algorithm apply CRC16 on each different block. The `Crc16` class of Java has to be used. It gives as output 16 bits codeword for each input string. There are different standard CRC polynomial on these parameters included in the library but for each string length we can choose the best one with respect to value of undetected error probability [1]. We have to hide 16 bits in LSB of one color channel so the input string length IL_{CRC16} has to be divisible by 32 and at least $32.16 = 512$ bits, i.e. 16 pixels. In this case, each LSB of the chosen color is used. The greater IL_{CRC16} the rougher the tamper detection.

As a result, $\texttt{ZBI}_s$ become $\texttt{SWI}_s$ - the image bits sequence with encoded public key (X.509 standard) concatenated at its end, DS bits as 3-LSB and $CRC16$ output bits as LSB. Only the signed and watermarked image sequence $\texttt{SWI}_s$ is sent to the server part of the proposed algorithm.

1.3. The server part

Run-up

The encoded public key is extracted from the end of the obtained $\texttt{SWI}_s$. Next the DSV from the used 3-LSB and the $CRC16V$ words from LSB are extracted, $\texttt{SWI}_s$ the $\texttt{ZBI}_s$ remain. It is needed in this manner because exactly $\texttt{ZBI}_s$ is used to produce signature in the client part of the algorithm.

Verifying a Digital Signature

A `KeyFactory` object in `KeyFactory` class have to be used in order to get DSA public key from its encoding. A signature is verified by an instance of the `Signature` class. For the `Signature` object the same algorithm (DSA) and the same message digest algorithm (SHA3-256) are specified. But now the initialization is with the public key. Then the *update* method of `Signature` object supplies $\texttt{ZBI}_s$. At the end, $verify$ method is applied on the extracted digital signature DSV and the calculated one in the $(r\|s)$ form. This method returns "true" only if DSV is the actual signature of the supplied data $\texttt{ZBI}_s$ generated by the private key k_i corresponding to the read public key. It calculates $\omega = s^{-1}\ mod\ q$, $u_1 = SHA3-256(\texttt{ZBI}_s).\omega\ mod\ q$, $u_2 = r.\omega\ mod\ q$ and $\upsilon = (g^{u_1}y^{u_2\ mod\ p})\ mod\ q$. So the signature is valid for $\upsilon = r$. If the returned value is "true" we can go on with CRC checking else it is clear the image is not taken with the given application.

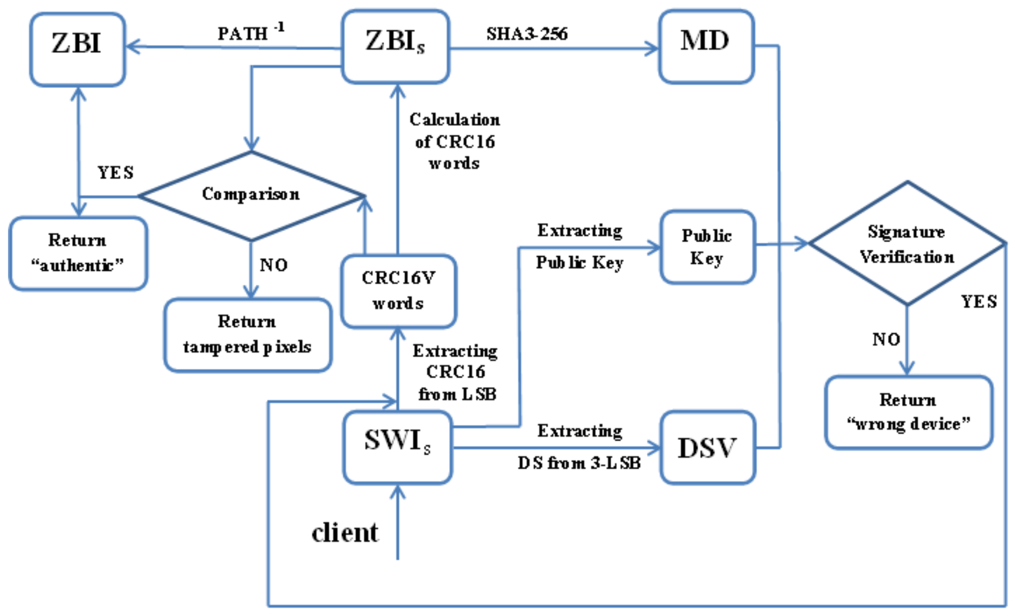

Figure 3: The algorithm server part

Tampering detection At this step the algorithm compare the extracted $CRC16V$ values and $CRC16$ words obtained from the received $\texttt{ZBI}_s$. If the image is not tampered it returns "true". Then applying on $\texttt{ZBI}_s$ the inverse permutation $\texttt{PATH}^{-1}$, `ZBI` is obtained. It is not the original image, but perceptually indistinguishable.

In the other case, the pixels with wrong $CRC16$ values have to be considered as tampered and a new two color image (the same size as OI) can be done - with black color only at tampered pixels.

Conclusions

We propose a signing and watermarking schema suitable for the features of a particular business problem. The parameters and tools are chosen according to the newest scientific achievements. The experiments with real data are needed in order to evaluate the lowest possible SNR for different parameters and to ensure the best performance of the algorithm. The real media environment have to be considered to supply the algorithm with error protection during the transmission.

Acknowledgements.

References

[1] Baicheva, T., Dodunekov, S., Kazakov, P., Undetected error probability performance of cyclic redundancy-check codes of 16-bit redundancy, IEE Proc Commun., 147 (5) 2000, 253 - 256.

[2] CERT Vulnerability Note VU# 836068, Kb.cert.org., December 31, 2008.

[3] Cox, I.J., Miller, M.L., Bloom, J.A., Fridrich J., Kalker, T.: Digital watermarking and steganography. Morgan Kaufmann, Burlington, MA, USA, 2008

[4] Hsien-Chu Wu, Chin-Chen Chang, A novel digital image watermarking scheme based on the vector quantization technique, Computers and Security, 24, (2005) 460-471.

[5] Liu, Z., Li, Q., Zhang, H., Peng, X., An Image Structure Information based Robust Hash for Tamper Detection and Localization, Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Darmstadt, Germany, 430-433.

[6] http://csrc.nist.gov/publications/PubsFIPS.html

[7] Rinu Tresa M. J., Athira M. B., Sobha T., A novel steganographic scheme based on hash function coupled with AES Encryption, Advanced Computing: An International Journal, 5 (1) (2014)

[8] Schneier,B., Schneier on Security: Cryptanalysis of SHA-1, February 18, 2005.