A Novel Patch-Based Digital Signature

Mahdi Saadatmand-Tarzjan[1]

[1] Assistant Prof, Medical Imaging Lab, Department of Electrical Engineering, Engineering Faculty,
Ferdowsi University of Mashhad, Mashhad, 91775-1111 Iran
saadatmand@um.ac.ir

Abstract:

In this paper a new patch-based digital signature (DS) is proposed. The proposed approach similar to steganography methods hides the secure message in a host image. However, it uses a patch-based key to encode/decode the data like cryptography approaches. Both the host image and key patches are randomly initialized. The proposed approach consists of encoding and decoding algorithms. The encoding algorithm converts the characters stream of the secret message to the patches stream of the DS image. The final image is further distorted by noise to hide the source patches. Nevertheless, the decoding algorithm uses a similarity measure to decode the DS image. Experimental results demonstrated that it is significantly robust against noise. The proposed approach has been successfully used for digital signature generation/verification and software copyright protection.

Keywords: Digital Signature, Patch-Based Image Processing, Steganography, Watermarking.

Submission date: 17 , 09 , 2016

Acceptance date: 10 , 06 , 2017

Corresponding author: M. Saadatmand-Tarzjan

Corresponding author's address: Medical Imaging Lab, Engineering Faculty, Ferdowsi University of Mashhad, Vakil-Abad Blv., Mashhad, P.O.Box 91775-111 Iran.

1. Introduction

The demand of secure transmission of information has been an issue since the first invented communication methods. Security has a top priority in any organization dealing with confidential data. Whatever is the method we choose for the security purpose, the major issue is the degree of security [1].

Generally, in publishing industries, unauthorized copying is a great concern. To tackle this problem, some invisible information (which cannot be easily extracted without a specialized technique) is embedded in the digital media [2]. Information hiding which includes a number of applications such as watermarking, steganography, fingerprinting, graphical passwords, copyright protection, digital signature (DS), and copy deterrence of software is an active research area [3].

For more details, in watermarking, the message, a low-level signal including owner identification and a digital time stamp, is directly placed into the host data set (e.g. an image) [4]. The buyer-seller watermarking protocol is a good example of using invisible watermarks for copy deterrence applications [5, 6]. Various techniques have been proposed for public-key watermarking systems, which allow two parties who have never met or exchanged a secret, to send hidden messages over public channels [7, 8].

Steganography, which is also considered as invisible watermarking, imperceptibly hides the secret message within the host data [9-11]. For example, substituting the message data with the least significant bits of the host data is a well-known frequently-used approach [12-16]. As an example, Aly [13] used the least significant bit method to embed the secret message in the motion vectors of the P- and B-frames in a compressed video.

By using the fingerprint approach, the owner can embed a serial number in the data set. It uniquely indicates the copyright information and thus, any unauthorized use of the data set can be traced [3, 17].

Generally, one of the most important topics in information security is the user authentication. There is a good security by using the text-based strong password schemes, but memorizing the password is often so difficult and users usually write them down on a piece of paper or save inside the computer. The graphical user authentication (or graphical password), known as an alternative solution, is actually based on this fact that the human tends to remember images. However, the graphical passwords are vulnerable to five attacks including brute force, dictionary, spyware, shoulder surfing, and social engineering attacks [18, 19].

Furthermore, by using DS in electronic government, confidentiality, integrity, authenticity, and anti-denial of circulating documents in network environment can be solved [20-22]. Jarusombat and Kittitornkun [23] proposed location based DS on mobile devices which increases the security and also compromises with low-computation capability and limited battery life of these devices.

However, with the advent of illegal copying of software and pervasive access to the Internet, software protection has gained increasing importance. Therefore, a complete solution is required to provide the software piracy protection capability. It should allow one to protect his/her software applications against piracy, illegal use, or illegitimate copy [24- 26].

Generally speaking, copy-protection techniques can be separated into token-based and token-less categories. Although token-based techniques are most common, users simply refuse to run the software until a specified token is present [12]. However, it can be simply cracked by removing tokens checks from the application code while the software remains fully functional. In contrast, typical tokenless techniques (such as obfuscation, encryption, watermarking, fingerprinting, and so forth) try to authenticate the user.

In this paper, we propose a new DS generation and verification method which can provide a breakthrough in all the mentioned applications. The proposed approach, called PBIS (patch-based image signature), includes encoding and decoding algorithms. First, the encoding algorithm (referred to as C-PBIS) decomposes the secret message to its fundamental characters while the DS image is randomly initialized. Then, for each message character, the corresponding patch in the codex (*i.e.* a set of randomly generated primary patches) is copied to a randomly-chosen position of the image. After encryption of some essential encoding parameters, whole the image is distorted by noise. Finally, the DS image is reshaped to hide its original size from the user. In the decoding algorithm (referred to as D-PBIS), after decryption of the encoding parameters, for each patch in the image, the corresponding character is specified by using a simple similarity measure.

Experimental results demonstrated significant robustness of PBIS against noise. Furthermore, the computational burden of both the encoding and decoding algorithms is light. They are easy and straightforward to implement. Therefore, PBIS can be used in a wide variety of security applications such as user authentication, software license agreement, and so forth.

The main contributions of this paper can be counted as follows:

- Development of the first digital signature method based on a patch-based image processing algorithm
- Encoding the secret message as a stream of successive patches in a randomly generated host image
- Decoding the patches stream by using a simple similarity measure
- Low computational complexity of both the encoding and decoding algorithms
- Significant robustness against noise

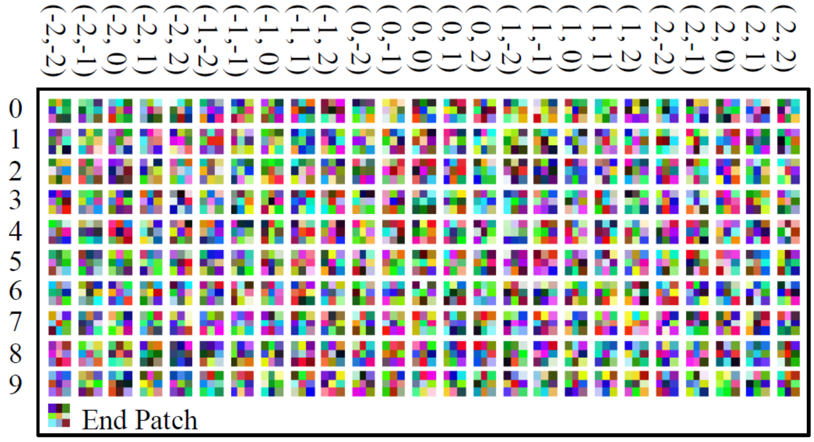


Fig. 1: A sample codex: each row corresponds to a digit and every column matches to a doublet (Δ*x*,Δ*y*). The single patch in the last row is the end-patch.

- Providing a new software copyright protection method based on PBIS

Furthermore, compared to typical DS methods, the proposed approach provides outstanding security protection due to,

- Using a very large codex as the encryption key
- Randomly altering a number of parameters of the encoding algorithm in every run
- Randomly generating the host image
- Distorting the encoded DS image by using strong additive noise

Therefore, PBIS is very secure against statistical analysis methods.

This paper is organized as follows: the proposed algorithm is explained in Section 2. Experimental results are given in Section 3. Finally, Section 4 is devoted to concluding remarks.

Notations used in this paper are fairly standard. Both sets and matrices are represented by upper-case letters while the later can be distinguished by italicface symbols. Boldface symbols are also used for vectors. For more details about the index-based representation of matrix/set components, see Appendix A.

2. Patch-Based Image Signature

Both C-PBIS and D-PBIS, require some primary parameters including the symbols set (S), codex set (P), and permitted movements matrices along *x*-axis (D^x) and *y*-axis (D^y) to properly work.

In more detail, the symbols set S, given by the DS images provider (so-called the security administrator), includes all the characters necessary to write the text-based secret message such as alphabets, digits, space, dash, under-line, comma, and so forth, as follows:

$$\mathrm{S}=\left\{\mathrm{S}_{(k)}\middle|k\in\mathrm{Z},0<k\le L_{\mathrm{S}}\right\} \tag{1}$$

where $S_{(k)}$ represents the *k*th component of S and L_{S}=||S|| where the operator ||.|| returns the cardinality of a set or the number of components within a matrix.

The matrices D^x (L_x=||D^x||) and D^y (L_y=||D^y||) are used to locate each encoding patch in the DS image with respect to the previous patch position.

The codex is a set of $L_{\mathrm{S}}L_xL_y$+1 randomly-initialized matrices of size *w*×*w* (called patches) with three-dimensional vectorial components. Therefore, we can write:

$$\mathrm{P}=\left\{P_{(i)}\middle|i\in\mathrm{Z},0<i\le L_{\mathrm{S}}L_xL_y+1\right\} \tag{2}$$

where $P_{(i)}$ is the *i*th patch of the codex. Specifically, the last patch of P (called the end-patch) is used to indicate the end of a patches stream (see Section 2.1.2). For example, a sample codex, including 10×5×5+1 patches, with S={0,1,…,9} and D^x=D^y=[-2,-1,0,1,2], is illustrated in Fig. 1.

Excluding the end-patch, P can be simplified to a matrix of size L_{S}×$L_x$$L_y$, as follows:

$$P=\begin{bmatrix} P_{(1,1)} & P_{(1,2)} & \cdots & P_{(1,L_xL_y)} \\ P_{(2,1)} & P_{(2,2)} & \cdots & P_{(2,L_xL_y)} \\ \vdots & \vdots & & \vdots \\ P_{(L_{\mathrm{S}},1)} & P_{(L_{\mathrm{S}},2)} & \cdots & P_{(L_{\mathrm{S}},L_xL_y)} \end{bmatrix} \tag{3}$$

where

$$P_{(p,q)}=\mathrm{P}_{((q-1)L_{\mathrm{S}}+p)} \tag{4}$$

Furthermore, the patch $P_{(p,q)}$ corresponds to the triplet ($\mathrm{S}_{(k)}$,Δ*x*,Δ*y*) only when *p*=*k*, $\Delta x=\mathrm{D}^x_{(r)}$, and $\Delta y=\mathrm{D}^y_{(s)}$; such that,

$$q=L_y(r-1)+s \tag{5}$$

or equivalently:

$$\left\{\begin{array}{l} r=\left[\dfrac{q-1}{L_y}\right]+1 \\ s=\mathrm{mod}(q-1,L_y)+1 \end{array}\right. \tag{6}$$

For example, in Fig. 1, $P_{(2,9)}$ corresponds to ($\mathrm{S}_{(2)}$=1,Δ*x*=-1,Δ*y*=1).

2.1. Encoding Algorithm

Fig. 2 shows the block diagram of the proposed encoding algorithm. C-PBIS treats the secret message as a stream of successive characters. Indeed, it converts this characters stream to a patches stream within the DS image as stated below.

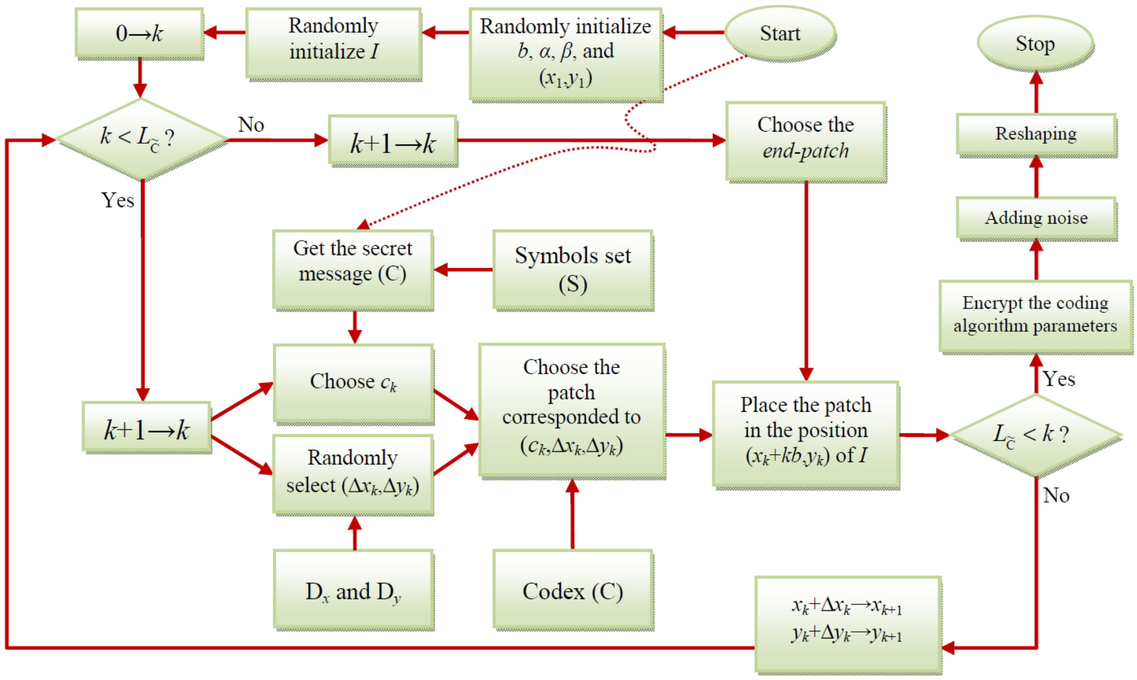

0→k
Randomly initialize I
Randomly initialize b, α, β, and (x1,y1)
Start
Stop
k < LC ?
No
k+1→k
Choose the end-patch
Reshaping
Yes
Adding noise
Get the secret message (C)
Symbols set (S)
Encrypt the coding algorithm parameters
Choose ck
Choose the patch corresponded to (ck,Δxk,Δyk)
Place the patch in the position (xk+kb,yk) of I
Yes
k+1→k
Randomly select (Δxk,Δyk)
LC < k ?
No
Dx and Dy
Codex (C)
xk+Δxk→xk+1
yk+Δyk→yk+1

Fig. 2. The block diagram of C-PBIS.

2.1.1. Initialization

First, all pixels of the DS image (*I*) are randomly initialized in the range [0,255]. Indeed, *I* is an image of size *b*×*αb*×3 which can be considered as a matrix of size *b*×*αb* with three-dimensional vectorial components. The parameters *b* and *α* are integer constants, separately specified for each secret message.

The secret message can be decomposed to its fundamental characters, as follows:

$$\mathrm{C} = \{\mathrm{C}_{(k)} \,|\, 1 \le k \le L_{\mathrm{C}}\} \subset \mathrm{S} \tag{7}$$

where C is the message characters set and $\mathrm{C}_{(k)}$ indicates the *k*th character of the message, and $L_{\mathrm{C}}=\|\mathrm{C}\|$.

As shown in Fig. 3, *I* is divided into *α* square blocks (including B_0, B_1, …, $B_{\alpha\text{-}1}$), all of size *b*×*b*. Except the first block (B_0) which is reserved for C-PBIS parameters (see Section 2.1.3), every remaining block can include only one patch. Thus, to preserve the possibility of using all components of D^x and D^y for localization of patches (see Section 2.1.2), we should have:

$$w + \max\left(\max_{\Delta x \in \mathrm{D}^x}(|\Delta x|), \max_{\Delta y \in \mathrm{D}^y}(|\Delta y|)\right) \le b \tag{8}$$

where |.| computes the first-order norm [27].

Besides, the patches stream should be finished by the end-patch. Thus, *α* should observe the following constraint:

$$L_{\mathrm{C}} + 2 \le \alpha \tag{9}$$

2.1.2. Patches Stream

First, the integer parameters x_1 and y_1 are randomly initialized in the range [1,*b*-*w*] with the following constraint (Let's *k*=1):

$$1 \le x_1, y_1 \le b - w \tag{10}$$

They are used to locate the first stream patch in B_1. Generally, in the *k*th step ($1 \le k \le L_{\mathrm{C}}+1$), the position of top-left corner of the current patch within B_k is specified by (x_k,y_k).

To increase the security of the encoding algorithm, every (x_k,y_k) should be randomly initialized. For achieving this purpose, the parameters Δx_k and Δy_k are randomly chosen from D^x and D^y, respectively, with the following constraints:

$$\begin{cases} -x_k < \Delta x_k \le b - w - x_k \\ -y_k < \Delta y_k \le b - w - y_k \end{cases} \tag{11}$$

In each step of the encoding algorithm, $(\Delta x_k,\Delta y_k)$ is provided to specify the position of the next patch in the image.

Next, if all message characters have been already encoded in the image (*i.e.* $L_{\mathrm{C}} \le k$), the end-patch is selected to specify the end of the stream. Otherwise, the patch $P_{(p_k,q_k)}$, which corresponds to the triplet $(c_k,\Delta x_k,\Delta y_k)$ according to (4)-(6), is chosen to specify both the *k*th message character and next patch position.

Subsequently, the selected patch is copied to the corresponding components of B_k as follows:

$$B_k(i,j) = P_{(p_k,q_k)}(i,j), \quad 1 \le i,j \le w \tag{12}$$

such that,

$$I(x_k + kb + i - 1, y_k + j - 1) = B_k(i,j) \tag{13}$$

For example, according to the above equations, the component located on the top-left corner of the *k*th patch (with *i*=*j*=1) is copied to the position (x_k+kb,y_k) of *I*, equalivalent to the position (x_k,y_k) of B_k.

Afterwards, the position of the next patch is explicitly computed as follows:

$$\begin{cases} x_{k+1} = x_k + \Delta x_k \\ y_{k+1} = y_k + \Delta y_k \end{cases} \tag{14}$$

Both x_{k+1} and y_{k+1} are also in the range [1,*b*-*w*] due to the constraints of (11).

After increasing *k* by one, the above process is repeated until the patch stream finishes with the end-patch.

2.1.3. Encoding Algorithm Parameters

Three essential parameters of C-PBIS including *b*, x_1, and y_1 should be encrypted within B_0, because they separately alter for each secrete message. For this aim, we take advantage of the index-based representation (see Appendix A) for components of *P* and *I*.

In more detail, for encrypting *b*, x_1, and y_1, whole components of the patches P_b, P_{x_1}, and P_{y_1}, respectively, are successively copied to the corresponding components of B_0 as follows:

$$B_0(i)=\begin{cases} P_b(\mathrm{mod}(i-1,w^2)+1) & 0<i\le w^2 \\ P_{x_1}(\mathrm{mod}(i-1,w^2)+1) & w^2<i\le 2w^2 \\ P_{y_1}(\mathrm{mod}(i-1,w^2)+1) & 2w^2<i\le 3w^2 \end{cases} \tag{15}$$

For the sake of importance of encoding parameters, the above copy procedure is repeated for γ ($1\le\gamma$) times. Thus, we should have:

$$3\gamma w^2 \le b^2 \Rightarrow \sqrt{3\gamma}\,w \le b \tag{16}$$

Combining the above constraint with (8), we can write:

$$w+\max\left(\max_{\Delta x\in D^x}(|\Delta x|),\ \max_{\Delta y\in D^y}(|\Delta y|),(\sqrt{3\gamma}-1)w\right)\le b \tag{17}$$

Obviously, the above equation provides a more sever limitation for the height of *I*.

2.1.4. Distortion by Noise

After encoding all stream patches, *I* is further altered by additive noise as follows:

$$\hat{\boldsymbol{I}}(i)=\boldsymbol{I}(i)+\eta\,\mathrm{rand}(), \qquad 1\le i\le \alpha b^2 \tag{18}$$

where the function rand() generates a three-dimensional random vector (with scalar components in the range [0,255]), according to the multiplicative congruential approach [28]. The coefficient η also indicates the strength of noise.

As demonstrated in experimental results (see Section 3.2), D-PBIS is fairly robust against noise. Therefore, in this step, *I* is distorted by noise to improve the security of the DS image by hiding the source codex patches from the user.

2.1.5. Image Reshaping

By copying each component of $\hat{I}$ to the corresponding one with the same index, the final DS image $\bar{I}$ of size $b\beta\times\frac{\alpha b}{\beta}$ is produced as follows:

$$\bar{\boldsymbol{I}}(i)=\hat{\boldsymbol{I}}(i), \qquad 1\le i\le \alpha b^2 \tag{19}$$

Here, β is a randomly-chosen divisor of α as follows:

$$\beta\in\{n\in\mathrm{N}|\ 1<n,\ \mathrm{mod}(\alpha,n)=0\} \tag{20}$$

Actually, in this step, the final DS image is reshaped to further hide its true width (*i.e.* *b*).

2.1.6. PBIS Key

The primary parameters of the proposed algorithm including P, S, D^x, D^y, *w*, and γ should be specified by the security administrator. Both C-PBIS and D-PBIS should use these parameters with exactly the same values to correctly work. At first glance, it may seem like a constraint, but that can provide a secure method to separate DS images produced by different administrators. In other words, the above primary parameters totally form the encryption key of PBIS. To correctly decode the DS image, this key should be entirely known. In other words, PBIS, similar to watermarking and steganography methods, hides the secure message in a host image. However, it provides a encoding key similar to the cryptography algorithms [29].

Anyway, for its very long length, the PBIS key is significantly complex. In more detail, the main part of the key is the codex. It consists of $L_\mathrm{S}L_xL_y$+1 patches. Each patch includes w^2 three-dimensional vectorial components whose elements range between 0 and 255. Thus, the number of all possible codexes is given by:

$$N(\mathrm{P})=256^{3w^2(L_\mathrm{S}L_xL_y+1)} \tag{21}$$

For example, when L_S=10 (*e.g.* once S includes only digits), L_x=L_y=3 (*e.g.* for D^x=D^y=[-1,0,1]), and *w*=3, we have $N(\mathrm{P})=2^{19656}\approx 1.11\times 10^{5917}$. If a computer evaluates each codex in one nanosecond, it takes more than 2.77×10^{5901} years to check all possible situations.

To indicate the security strength of PBIS, it is sufficient to compare its key length with those of other typical cryptography methods. In more detail, the key-length of DES, 3DES, AES, Blowfish, and RSA are 56, 168, 256, 448, and 4096 bits, respectively [30]. In other words, PBIS key length is at least 4.5 times larger than best conventional cryptography algorithms.

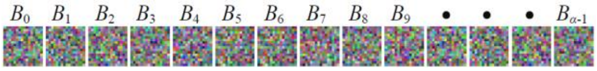


Fig. 3. The DS image *I* is divided into *α* blocks, each of size *b*×*b*.

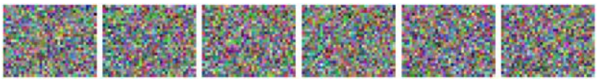


Fig. 4. Six iconic DS images provided for the secret message 0123456789 with the codex of Fig. 1 and *η*=100%, all of size 36×27.

Furthermore, C-PBIS uses a number of parameters, including *b*, *α*, *β*, and (x_1,y_1), which should be separately initialized for each secret message, with observing (17), (9), (20), and (11), respectively (see Section 2.1.3). Therefore, as illustrated in Fig. 4, even for the same secret message, each time, C-PBIS provides a different DS image because first, the DS image and above parameters are randomly initialized and second, the final DS image is further altered by noise.

2.2. Decoding Algorithm

Once the PBIS key is given, decoding of the DS image is straightforward. Generally, for the candidate patch $\overline{P}$, the most similar patch in the codex can be given by using the first-order norm:

$$m = \text{index}\left(\min_{j}\left(\operatorname{sum}_{i=1}^{w^2}(\left|\overline{\boldsymbol{P}}(i) - \boldsymbol{P}_j(i)\right|)\right)\right) \tag{22}$$

Although we take advantage of the above similarity function for its simplicity, higher order norms or cross-correlation measure [27] can be also used in the same manner.

2.2.1. Initialization

Initially, to decrypt the exclusive encoding parameters, including b, α, x_1, and y_1, the first $3\gamma w^2$ components of $\bar{I}$ (*i.e.* $\bar{\boldsymbol{I}}(1)$ to $\bar{\boldsymbol{I}}(3\gamma w^2)$) are divided into 3γ subsequent patches (including $\overline{P}_1$, $\overline{P}_2$, …, and $\overline{P}_{3\gamma}$) such that each consists of w^2 components. According to Section 2.1.3, all the patches $\overline{P}_1$, $\overline{P}_4$, …, $\overline{P}_{3\gamma-2}$ correspond to P_b. Thus, b can be obtained as the median of indexes of all matched patches as follows:

$$b = \operatorname*{median}_{k=1}^{\gamma}\left(\text{index}\left(\min_{j}\left(\operatorname{sum}_{i=1}^{w^2}(\left|\overline{\boldsymbol{P}}_{3k-2}(i) - \boldsymbol{P}_j(i)\right|)\right)\right)\right) \tag{23}$$

In the same manner, we can write:

$$x_1 = \operatorname*{median}_{k=1}^{\gamma}\left(\text{index}\left(\min_{j}\left(\operatorname{sum}_{i=1}^{w^2}(\left|\overline{\boldsymbol{P}}_{3k-1}(i) - \boldsymbol{P}_j(i)\right|)\right)\right)\right) \tag{24}$$

$$y_1 = \operatorname*{median}_{k=1}^{\gamma}\left(\text{index}\left(\min_{j}\left(\operatorname{sum}_{i=1}^{w^2}(\left|\overline{\boldsymbol{P}}_{3k}(i) - \boldsymbol{P}_j(i)\right|)\right)\right)\right) \tag{25}$$

Obviously, α is given by:

$$\alpha = \frac{\left\|\bar{I}\right\|}{b} \tag{26}$$

Therefore, $\hat{I}$ can be obtained by reshaping $\bar{I}$ to a matrix of size b×αb, according to (19).

It is worth to note that there is no way to obtain I from $\hat{I}$, because of distortion by unknown additive noise. Instead, we can decrypt the secret message through the similarity measure of (22).

2.2.2. Characters Stream

Similar to the approach used in C-PBIS, we should, again, convert the patches stream of the DS image to the characters stream of the secret message.

Generally, in the kth step (initially, set k=1) of the decoding algorithm, the position of the top-left corner of the current patch, of size w×w, in $\hat{I}$ (shown by $\hat{P}_k$) is given by (kb+x_k,y_k). In this case, the kth character of the secret message can be specified by obtaining the most similar patch in the codex as follows:

$$(p_k, q_k) = \text{index}\left(\min_{(p,q)}\left(\operatorname{sum}_{i=1}^{w^2}(\left|\hat{\boldsymbol{P}}_k(i) - \boldsymbol{P}_{(p,q)}(i)\right|)\right)\right) \tag{27}$$

Obviously, S_{p_k} specifies the kth character of the secret message. Furthermore, according to (6), to obtain the position of the next patch in $\hat{I}$, we should have:

$$\begin{cases} r_k = \left[\dfrac{q_k - 1}{L_y}\right] + 1 \\ s_k = \operatorname{mod}(q_k - 1, L_y) + 1 \end{cases} \tag{28}$$

such that $x_{k+1} = x_k + \mathrm{D}^x_{r_k}$ and $y_{k+1} = y_k + \mathrm{D}^y_{s_k}$. After augmenting k by one, the above process is repeated until the patch stream finishes with the end-patch.

2.3. Computational Complexity

The proposed encoding algorithm is significantly simple and straightforward to implement. Its most computationally complex steps are randomly initialization of the DS image and distortion by noise. Therefore, the overall computational complexity of C-PBIS is of $O(\alpha b^2)$.

However, compared to the encoding algorithm, the decoding algorithm may have slightly more computational burden. In more detail, the computational complexity of (22) is of $O(w^2 L_x L_y L_\mathrm{S})$. This similarity measure is evaluated for each candidate patch of the DS image. Thus, the overall computational complexity of D-PBDIS is of $O(w^2 L_x L_y L_\mathrm{S} L_\mathrm{C})$. The parameters w, L_x, and L_y are usually set to small constants. Thus, the computation burden of the decoding algorithm is linearly related to the length of the secret message and cardinality of the symbols set, *i.e.* $O(L_\mathrm{S} L_\mathrm{C})$.

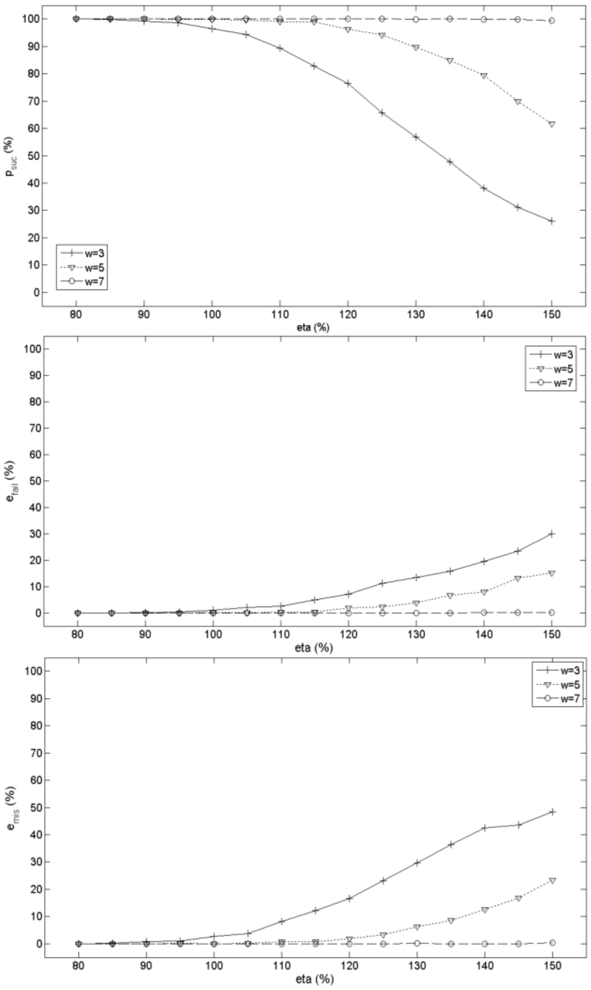


Fig. 5. Variations of (top) p_{suc}, (centre) e_{fail}, and (bottom) e_{mis} versus η for different values of w.

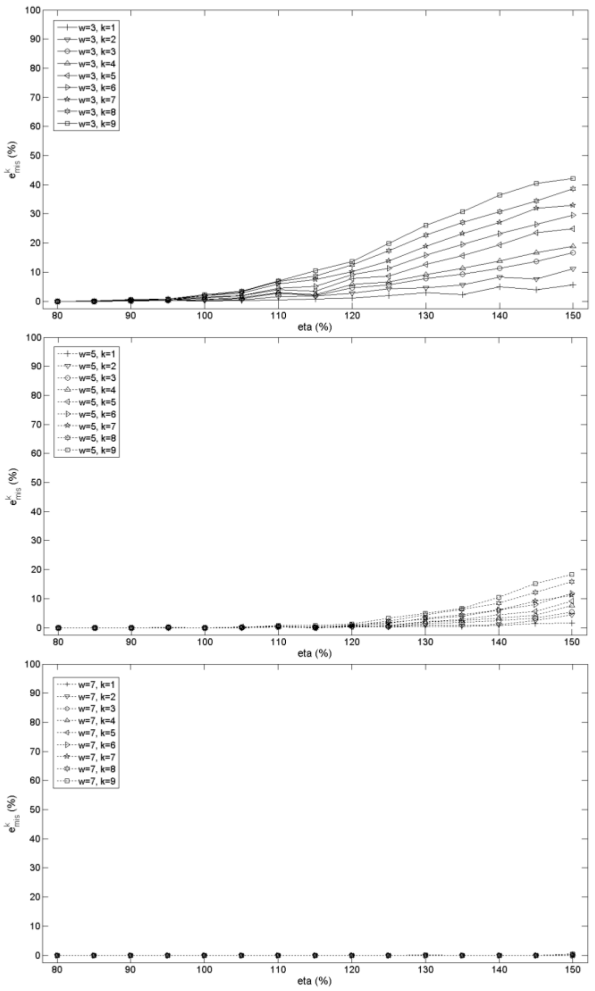


Fig. 6. Variations of of e_{mis}^{k} (1≤k≤9) versus η with (top) w=3, (centre) w=5, and (bottom) w=7.

3. Experimental Results

Although patch-based methods are well-known and frequently-used in image processing [31-36], PBIS can be counted as the first patch-based algorithm to generate digital signatures.

Furthermore, the proposed digital signature is, indeed, an iconic small image file (with the size of a few kilo Bytes) which can easily be saved on the hard or flash disks. It can be attached to emails or upload to WEB sites as the user authentication. Besides, PBIS can convert whole the email (SMS) text to a DS image for providing a very secure communication on Internet.

The performance of PBIS was evaluated by using an Intel Core 2 Duo 2.0-GHz Laptop with 3-GB main memory in the MATLAB environment.

3.1. Producing Codex Sets

In all experimental evaluations, the symbols set included only digits and D^x=D^y=[-2,-1,0,1,2]. Consequently, each codex includes 251 patches. Also, we experimentally choose γ=3.

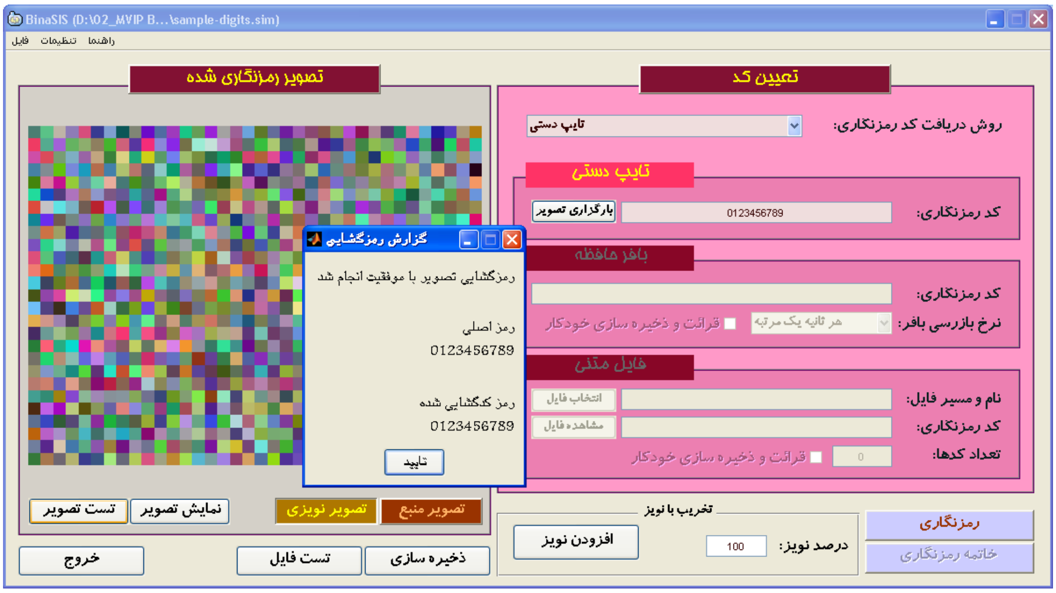
Furthermore, in each codex, the patches should be as diverse as possible to improve the performance of D-PBIS against noise. Generally, more dissimilar patches can be better distinguished from each other.

Therefore, we provided a simple and effective algorithm to produce diverse patches. In more detail, after initialization of the first patch, a new one is randomly produced. The patch will be added to the codex, if it observes the following constraint:

$$\theta < \min_{j=1}^{k-1}\left(\frac{1}{w^2}\operatorname{sum}_{i=1}^{w^2}\left(\left|P_k(i)-P_j(i)\right|\right)\right) \tag{29}$$

where θ is the threshold of minimum acceptable difference between patches. Otherwise, a new patch is randomly generated. The above procedure repeated until all required patches are initialized.

Obviously, with a too large θ, the codex initialization algorithm may fail to converge. In contrast, with a too small θ, D-PBIS cannot provide acceptable performance against noise. Here, we experimentally obtained θ=69.



BinaSIS (D:\02_MVIP B...\sample-digits.sim)

گزارش رمزگشایی

رمزگشایی تصویر با موفقیت انجام شد

رمز اصلی

0123456789

رمز کدگشایی شده

0123456789

روش دریافت کد رمزنگاری: تایپ دستی

کد رمزنگاری: 0123456789

نرخ بازرسی بافر: هر ثانیه یک مرتبه

□ قرائت و ذخیره سازی خودکار

تعداد کدها: 0

□ قرائت و ذخیره سازی خودکار

درصد نویز: 100

Fig. 7. The developed software for PBIS (GUI texts are in Farsi).

3.2. Robustness against Noise

The proposed digital signature approach is significantly secure, because of:

- Using very long key
- Randomly initialization of some parameters of C-PBIS for each secure message
- Distortion of the DS image by additive noise

Indeed, C-PBIS produces different DS images for even the same secure message. This may be the most important and unparalleled property of the proposed algorithm. For example, Fig. 4 illustrates six different DS images provided for the same message 0123456789 with the codex of Fig. 1 and η=100%.

Furthermore, the statistical analysis methods are not useful to discover the PBIS key because; both the host image and encoding patches are generated by using the same random-number generation method. Moreover, it is also employed to distort the final DS image.

D-PBIS takes advantage of compactness and connectivity of patches to decode the DS image. Thus, it should properly work even for an altered DS image.

To test performance of D-PBIS against noise, three different codex sets with w=3, 5, and 7 were employed. D^x, D^y and S were also chosen as the same as those used in Fig. 1. η was increased by the step of 5% ranging from 80% to 150%. In each step, the same secure code (*i.e.* 123456789) was encoded and then decoded for 1000 times. In decoding of the secure message, three situations may take place:

- The secure message is successfully decoded.
- The decoding algorithm fails because of wrong encryption of the encoding parameters.
- Some characters of the decrypted message mismatch with those of the source message.

Thus, in order to evaluate the performance of D-PBIS, the following evaluation measures were employed:

$$p_{suc}(w,\eta)=\frac{\text{Number of successes}}{1000}\times 100\% \tag{30}$$

$$e_{fail}(w,\eta)=\frac{\text{Number of failures}}{1000}\times 100\% \tag{31}$$

$$e_{mis}(w,\eta)=\frac{\text{Number of mismatches}}{1000}\times 100\% \tag{32}$$

$$e^{k}_{mis}(w,\eta)=\frac{\text{Number of mismatches of the } k\text{th digit}}{1000}\times 100\% \tag{33}$$

where p_{suc}, e_{fail}, e_{mis}, and e^{k}_{mis} computes the percents of successful decoding attempts, decoding failures, whole message mismatches, and mismatches of the kth character, respectively. Obviously, we have:

$$p_{suc}(w,\eta)=100\%-e_{fail}(w,\eta)-e_{mis}(w,\eta) \tag{34}$$

and

$$e_{mis}(w,\eta)=\sum_{k=1}^{L_{\tilde{C}}} e^{k}_{mis}(w,\eta) \tag{35}$$

Fig. 5 shows variations of p_{suc}, e_{fail}, and e_{mis} versus η for different values of w. For all values of w with $\eta\leq$80%, D-PBIS was quite successful to decode all DS images (p_{suc}=100%). Although p_{suc} decreases by increasing η, the sensitivity to noise significantly improved by augmenting w; such that for w=7, we have p_{suc}=99.4% with η=150%.

For every values of η and w, e_{mis} was approximately 2 times greater than e_{fail}. In other words, from among every three unsuccessful decoding attempts, D-PBIS failed only once.

Fig. 6 illustrates variations of e^{k}_{mis} versus η for different values of w. Obviously, the mismatch error increases by augmenting the order of the character in the message. As an explanation for this outcome, the patches/characters stream is a sequential process. A specific character of the message can be correctly decoded only when all previous characters have been properly decrypted. Thus, by increasing the order of

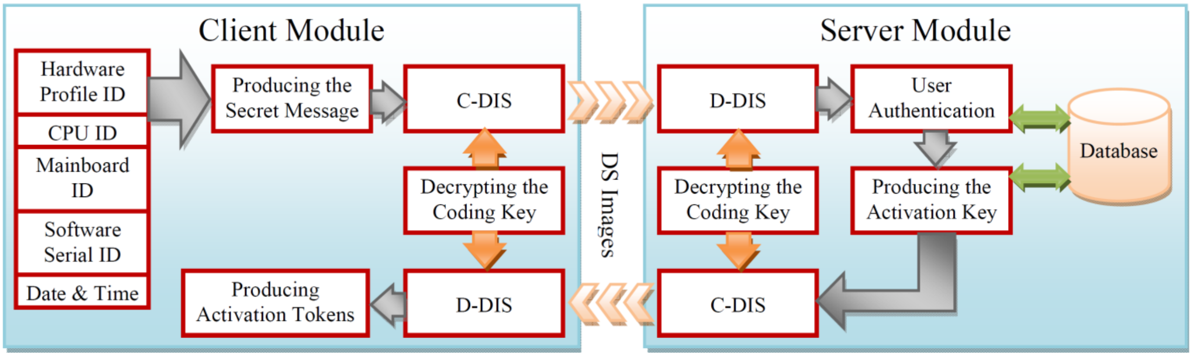


Fig. 8. The block diagram of the developed software for license agreement.

the character in the message, the probability of incorrect decoding is raised. However, as illustrated in Fig. 5, by using large-enough patches (*e.g.* w=7), the error percent significantly decreases.

As demonstrated, D-PBIS may sometimes fail to decode the distorted DS image. Therefore, after finishing C-PBIS, it is necessary to check correct decoding of the image. In this case, if D-PBIS fails, another DS image should be produced. Fig. 7 illustrates the developed software for PBIS.

3.3. Software Copyright Protection

Although PBIS is originally developed for digital signature, it can be used for a wide variety of security applications such as user authentication, software license agreement, secure communication, and so forth. For example, the block diagram of the developed software for license agreement is shown in Fig. 8. It consists of two parts: server and client modules. The license checking process includes three steps:

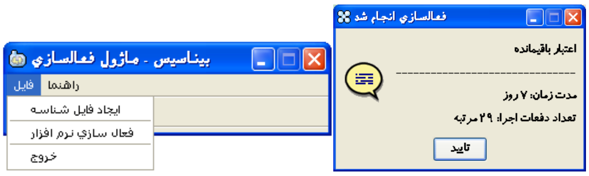
- First, the client module (see Fig. 9) collects some fundamental information about the user's PC including date, time, serial numbers of the mainboard and CPU, hardware profile (developed by the operating system), and the serial number of the current copy of the software. Then it encrypts all these data in an image by using C-PBIS. Next, the image is sent via Internet or multimedia messaging service (MMS) to the server module.
- Second, by using D-PBIS, the server module decrypts the data of the received image. It saves the user's PC specifications and the software serial number in the database. After license agreement, D-PBIS produces an activation message including the activation time period, maximum allowed number of runs, and activation key serial number. Next, the activation message is encrypted in a new image for sending back to the client module (see Fig. 10).
- Finally, after decoding the activation message, the client module produces proper secure soft-tokens on the user's computer to activate the software (see Fig. 9).
- We successfully used the above scheme for license copyright protection of all software products of Binatooth Co. [1] such as Bina camera calibration (BCC), Bina plate recorder (BPR-4P and BPR-iV), and Binacamp.

4. Conclusion

In this paper, a new patch-based algorithm for digital signature generation and verification was proposed. It uses a set of randomly generated patches to encode a set of character symbols. Indeed, C-PBIS converts the characters stream of the secret message to the patches stream of the DS image. C-PBIS uses a very long key which considerably improves the security of generated DS images. Besides, both the host image and codex patches are initialized by using the same random-number generator method. It is also employed to further distort the final DS image. For all these reasons, the proposed algorithm is very secure against statistical analysis methods.

D-PBIS decodes the DS image by using a simple similarity measure. Actually, it takes advantage of the compactness and connectivity of patches. For this reason, D-PBIS was significantly robust against noise. For example, experimental results demonstrated that D-PBIS successfully decoded 99.4% of DS images generated by C-PBIS with the patch size of w=7 and noise strength of η=150%.

Although PBIS is originally proposed for digital signature applications, it can be used for a wide variety of security applications such as user authentication, software license agreement, secure communication, and so forth.



اعتبار باقیمانده

مدت زمان: ۷ روز

تعداد دفعات اجرا: ۲۹ مرتبه

Fig. 9. The client module of the license agreement software (left) and its message after activation (right). GUI texts are in Farsi.

[1] http://www.binatooth.ir

Appendix A

Let’s *M* be a matrix of size *e*×*f* whose components are also matrices of size *g*×*h*. The component located in the *r*th row and *c*th column of *M* is represented by $M_{(r,c)}$. Moreover, the component of *p*th row and *q*th column of $M_{(r,c)}$ is also indicated by $M_{(r,c)}(p,q)$.

Furthermore, we can equivalently use the index-based method to uniquely indicate each component of *M*. In more detail, the *i*th component of *M*, shown by $M_{(i)}$ (where *i* is the index), corresponds to $M_{(r,c)}$, if and only if:

$$i = (c-1)\times e + r \tag{36}$$

or equivalently:

$$\begin{cases} c = \left[\dfrac{i-1}{e}\right] + 1 \\ r = \operatorname{mod}(i-1, e) + 1 \end{cases} \tag{37}$$

where [.] returns the integer fraction of a number and mod(*a*,*b*) computes *a* modulo *b*. Similarly, the *j*th component of M_i is given by:

$$M_{(i)}(j) = M_{(r,c)}(\operatorname{mod}(j-1, g), \left[\dfrac{j-1}{g}\right]) \tag{38}$$

Obviously, when *e*=1 or *f*=1, we have:

$$\begin{cases} e = 1 \Rightarrow M_{(c)} = M_{(1,c)} \\ f = 1 \Rightarrow M_{(r)} = M_{(r,1)} \end{cases} \tag{39}$$

In the same manner, we can write:

$$\begin{cases} g = 1 \Rightarrow M_{(i)}(q) = M_{(r,c)}(1,q) \\ h = 1 \Rightarrow M_{(i)}(p) = M_{(r,c)}(p,1) \end{cases} \tag{40}$$

References

[1] S. K. Bandyopadhyay, D. Bhattacharyya, D. Ganguly, S. Mukherjee, and P. Das, “A tutorial review on steganography,” In Proc. Int’l Conf. Contemporary Computing, 2008.

[2] N. Sharma, J. Bhatia, and N. Gupta, “An encrypto-stego technique based secure data transmission system director,” CDAC Mohali, India, 2000. [Online] Available: http://www.scribd.com/doc/63422680/encrypto

[3] M. M. Amin, M. Salleh, S. Ibrahim, M. R. Katmin, and M. Z. I. Shamsuddin, “Information hiding using steganography,” In Proc. 4th National Conf. Telecommunication Technology, Malaysia, pp. 21–25, 2003.

[4] M. Chandra, S. Pandey, and R. Chaudhary, “Digital watermarking technique for protecting digital images,” In Proc. 3rd IEEE Int. Conf. Computer Science and Information Technology, vol. 7, pp. 226–233, 2010.

[5] N. Memon, and P. W. Wong, “A buyer–seller watermarking protocol,” IEEE Trans. Image Processing, vol. 10, no. 4, pp. 643-649, 2001.

[6] A. Khan, F. Jabeen, F. Naz, S. Suhail, M. Ahmed, and S. Nawaz, “Buyer seller watermarking protocols issues and challenges – A survey,” J. Network and Computer Applications, vol. 75, pp. 317–334, 2016.

[7] Q. Wang, F. Sun, and F. Liu, “Research on public-key digital watermarking system,” In Proc. IEEE 3rd Int’l Conf. Communication Software and Networks, pp. 158–162, 2011.

[8] R. Xie, K. Wu, J. Du, and C. Li, “Survey of public key digital watermarking systems,” In Proc. 8th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 2, pp. 439–443, 2007.

[9] M. C. Chen, S. S. Agaian, and C. L. P. Chen, “Generalized collage steganography on images,” In Proc. IEEE Int’l Conf. Systems, Man and Cybernetics, pp. 1043–1047, 2008.

[10] S. Singh and G. Agarwal, “Use of image to secure text message with the help of LSB replacement,” Int. Journal of Applied Engineering Research Dindigul, vol. 1, no. 1, pp. 200–205, 2010.

[11] M. Nooshyar, A. Aghagolzadeh, H. R. Rabiee, and E. Mikaili, “Robust distributed source coding with arbitrary number of encoders and practical code design technique,” J. Iranian Association of Electrical and Electronics Engineers, vol. 10, no. 2, pp. 57–69, 2013.

[12] D. Artz, “Digital steganography: hiding data within data,” IEEE Internet Computing, vol. 5, no. 3, pp. 75–80, 2001.

[13] A. H. Aly, “Data hiding in motion vectors of compressed video based on their associated prediction error,” IEEE Trans. Information Forensics and Security, vol. 6, no. 1, pp. 14–18, 2011.

[14] G. Swain, “A steganographic method combining LSB substitution and PVD in a block,” Procedia Computer Science, vol. 85, pp. 39–44, 2016.

[15] M. B. Begum and Y. Venkataramani, “LSB based audio steganography based on text compression,” Procedia Engineering, vol. 30, pp. 703–710, 2012.

[16] R. Sharma, E. Walia, and D. Sharma, “Analysis of non-adaptive and adaptive edge based LSB steganography for colored images,” Int. Journal of Computing and Business Research, vol. 2, no. 1, 2011.

[17] R. Sun, X. Yan, and J. Gao, “Robust video fingerprinting scheme based on contourlet hidden Markov tree model,” Optik - International Journal for Light and Electron Optics, vol. 128, pp. 139–147, 2017.

[18] A. H. Lashkari, M. Masrom, and A. A. Manaf, “A secure recognition based graphical password by watermarking,” In Proc. 11th IEEE Int. Conf. Computer and Information Technology, pp. 164–170, 2011.

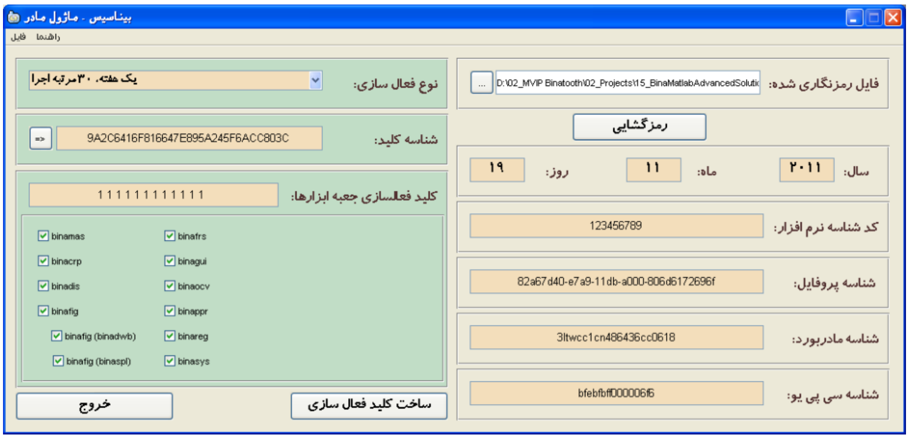
[19] W. Meng, W. Li, L.-F. Kwok, K.-K. R. Choo, “Towards enhancing click-draw based graphical passwords using multi-touch behaviours on smartphones,” Computers & Security, vol. 65, pp. 213–229, 2017.

[20] N. Nikolaidis and I. Pitas, “Copyright protection of images using robust digital signatures,” In Proc. IEEE Int’l Conf. Acoustics, Speech, and Signal Processing, vol. 4, pp.2168–2171, 1996.

[21] N. Zhu and G. X. Xiao, “The application of a scheme of digital signature in electronic government,” In Proc. Int. Conf. Computer Science and Software Engineering, vol. 3, pp. 618–621, 2008.

[22] S. Rashidi, A. Fallah, and F. Tohidkhah, “Dynamic signature verification based on extraction of stability dominant points and segmentation of pattern,” Journal of Iranian Association of Electrical and Electronics Engineers, vol. 9, no. 1, pp. 23-32, 2012.

[23] S. Jarusombat and S. Kittitornkun, “Digital signature on mobile devices based on location,” In Proc. Int’l



فایل رمزنگاری شده: D:\02_MVIP Binatooth\02_Projects\15_BinaMatlabAdvancedSolutic

سال: ۲۰۱۱ ماه: ۱۱ روز: ۱۹

کد شناسه نرم افزار: 123456789

شناسه پروفایل: 82a67d40-e7a9-11db-a000-806d6172696f

شناسه مادربورد: 3ltwcc1cn486436cc0618

شناسه سی پی یو: bfebfbff000006f6

نوع فعال سازی: یک هفته، ۳۰ مرتبه اجرا

شناسه کلید: 9A2C6416F816647E895A245F6ACC803C

کلید فعالسازی جعبه ابزارها: 111111111111

☑ binamas ☑ binafrs
☑ binacrp ☑ binagui
☑ binadis ☑ binaocv
☑ binafig ☑ binappr
☑ binafig (binadwb) ☑ binareg
☑ binafig (binaspl) ☑ binasys

Fig. 10. The server module of the license agreement software (GUI texts are in Farsi).

Symp. Communications and Information Technologies, pp. 866–870, 2006.

[24] S. Mumtaz, S. Iqbal, and E. I. Hameed, "Development of a methodology for piracy protection of software installations," In Proc. 9th Int'l Multitopic Conference, pp. 1–7, 2005.

[25] P. Djekicand and C. Loebbecke, "Software piracy prevention through digital rights management systems," In Proc. 7th IEEE Int. Conf. E-Commerce Technology, pp. 504–507, 2005.

[26] A. Rasch and T. Wenzel, "The impact of piracy on prominent and non-prominent software developers," Telecommunications Policy, vol. 39, no. 8, pp. 735–744, 2015.

[27] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Elsecier Academic Press, 2nd ed., 2003.

[28] D. Y. Downham and F. D. K. Roberts, "Multiplicative congruential pseudo-random number generators," The Computer Journal, vol. 10, no. 1, pp. 74–77, 1967.

[29] M. J. Simovits, *The DES, An Extensive Documentation and Evaluation*. Aegean Park Press, 1994.

[30] P. Patil, P. Narayankar, D. G. Narayan, S. M. Meena, "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish," Procedia Computer Science, vol. 78, pp. 617–624, 2016.

[31] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H264 video encoding standard," IEEE Trans. Circuits and Systems for Video Tech., vol. 13, no. 7, pp. 560–576, 2003.

[32] M. Ashikhmin, "Fast texture transfer," IEEE Computer Graphics and Applications, vol. 23, no. 4, pp. 38–43, 2003.

[33] J. Hays and A. A. Efros, "Scene completion using millions of photographs," ACM Trans. Graph., vol. 26, no. 3, pp. 87–94, 2007.

[34] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch-based image sequence restoration," IEEE Trans. Patt. Anal. Mach. Intell., vol. 29, no. 6, pp. 1096–1102, 2007.

[35] C. Li, C. Y. Kao, J. C. Gore, and Z. Ding, "Minimization of region-scalable fitting energy for image segmentation," IEEE Trans. Image Processing, vol. 17, no. 10, pp. 1940–1949, 2008.

[36] M. Saadatmand-Tarzjan, H. Ghassemian, "On analytical study of self-affine maps," J. Iranian Association of Electrical and Electronics Engineers, vol. 12, no. 3, 2015.