Secret image sharing based on cellular automata and steganography

Z. Eslami*, S.H. Razzaghi, J. Zarepour Ahmadabadi
Department of Computer Science, Shahid Beheshti University, G.C., Tehran, Iran

1. Introduction

Techniques to share a secret image have attracted considerable attention in the recent years [1–8]. However, there are important issues that such techniques should deal with. The first one is accidental or intentional loss/corruption of images that might occur if only a single party has access to the data. On the other hand, if several participants share parts of the secret image, care must be taken to ensure that no malicious shareholder is able to manipulate his/her data. The second issue is the need to keep invaders unaware not only of the content of the secret image itself but also of the very fact that an image is being transferred.

Secret sharing schemes which protect and distribute a secret content among a group of participants provide solutions to the first issue. In this regard, the basic example, proposed first by Shamir [9] and Blakley [10], is the concept of a \((t, n)\)-threshold secret sharing scheme which encodes a secret data set into \(n\) shares and distributes them among \(n\) participants in such a way that any \(t\) or more of the shares can be collected to recover the secret data, but any \(t – 1\) or fewer of them provides no information about the secret. Moreover, to ensure recovery of the original secret information some authentication process must be employed so that any manipulation of shares is detected with high probability.

To tackle the second concern, steganographic techniques are usually employed [1–3,11–14]. In these methods, first some innocent-looking images, called cover images, are selected. Then the secret data are embedded into cover images and the resulting stego images are distributed among participants using some secret sharing scheme. Clearly, in order not to invoke suspicion, the embedding should create high-quality stego images such that the changes are not visually perceptible.

So far, two most popular steganographic methods are the least significant bits (LSBs) replacement [15–17] and the modulus operation [18–20]. For example, Wu et al. [8] proposed a sharing and...
interpolation algorithms, the proposed scheme is of order 2. One-dimensional memory cellular automata scheme of Chang et al. [3].

The rest of this paper is organized as follows: related work is covered in Section 2. In Section 3, the proposed scheme as well as our motivation to employ CA instead of Shamir’s sharing scheme is outlined. Section 4 provides comparison between our proposed scheme and other methods in the literature. Finally, the conclusions of this paper are presented in Section 5.

2. Related works

In this section, we briefly review cellular automata and the scheme of Chang et al. [3].

2.1. One-dimensional memory cellular automata

One-dimensional finite memory cellular automata are discrete dynamical systems formed by a finite array of N identical objects called cells, where each cell can assume a state \( s \in \{0,1\} \), updated synchronously in discrete time steps according to a local transition function. The updated state of each cell depends on the variables of this function which are the previous states of a set of cells, including the cell itself, and constitute its neighborhood. For the \( i \)-th cell, denoted by \( (i) \), we consider the symmetric neighborhood of radius \( r \) which is defined as \( \mathcal{N}_i = \{i-r, \ldots, i, \ldots, i+r\} \). Let \( a_i(T) \) denote the state of \( (i) \) at time \( T \). Then, the local transition function of the cellular automata with radius \( r \) has the following form:

\[
a_i^{(T+1)} = f(a_{i-r}^{(T)}, \ldots, a_{i+r}^{(T)}) , \quad 0 \leq i \leq N - 1
\]

or equivalently,

\[
a_i^{(T+1)} = f(a_{i-r}^{(T)}), \quad 0 \leq i \leq N - 1,
\]

where \( a_i^{(T)} \subset \mathbb{Z}_2 \) stands for the states of the neighbor cells of \( (i) \) at time \( T \). Furthermore, if \( i \equiv j \pmod{N} \), then it is assumed that \( a_i(T) = a_j(T) \) to ensure well-defined dynamics of the CA.

The vector \( C_i(T) = (a_0^{(T)}, \ldots, a_{N-1}^{(T)}) \) is called the configuration of CA at time \( T \) and \( C_i(0) \) is the initial configuration. Moreover, the sequence \( (C_i(T))_{0 \leq T \leq k} \) is called the evolution of order \( k \) of the CA and the set of all possible configurations of the CA is denoted by \( \mathcal{C} \).

The global function of the CA is a linear transformation, \( \Phi : \mathcal{C} \rightarrow \mathcal{C} \), which determines the configuration at the next time step during the evolution of the CA, that is, \( C_i(T+1) = \Phi(C_i(T)) \). For a CA with bijective \( \Phi \), there exists another cellular automaton, called its inverse, with global function \( \Phi^{-1} \), and the CA itself is called reversible. In such CAs the evolution backward is possible [25].

The local transition function of a linear cellular automata (LCA) with radius \( r \) is of the following form:

\[
a_i^{(T+1)} = \sum_{j=-r}^{r} z_{i-j} a_{i+j}^{(T)} \pmod{2}, \quad 0 \leq i \leq N - 1,
\]

where \( z_{i-j} \in \mathbb{Z}_2 \) for every \( j \). Since there are \( 2r+1 \) neighbor cells for \( (i) \), there exist \( 2^{2r+1} \) LCAs and each of them can be specified by an integer \( w \) called rule number which is defined as follows:

\[
w = \sum_{j=-r}^{r} z_{i-j} 2^j,
\]

where \( 0 \leq w \leq 2^{2r+1} - 1 \).

The LCAs considered so far are memoryless, i.e., the updated state of a cell depends on its neighborhood configuration only at the preceding time step. Nevertheless, one can consider cellular automata for which the state of neighboring cells at time \( T \) as well as \( T-1, T-2, \ldots \) contributes to determine the state at time \( T+1 \). This is the concept of the memory cellular automata (MCA) [26]. Hereafter, by a CA, we mean a particular type of MCA called the \( t \)-th order linear MCA (LMCA) whose local transition function takes the following form:

\[
f_i^{(T+1)} = f_i(a_{i-r}^{(T)} + a_{i-r}^{(T-1)} + \cdots + a_{i+r}^{(T-t+1)}) \pmod{2}, \quad 0 \leq i \leq N - 1,
\]

where \( f_i \) is the local transition function of a particular LCA with radius \( r \), for \( 1 \leq i \leq t \). In this case, we require \( t \) initial configurations \( C_i(0), \ldots, C_i(t-1) \) to start the evolution of LMCA. Furthermore, in order for this cellular automaton to be reversible, we have the following proposition. A proof for this proposition can be found in [22].

**Proposition 1.** If \( f_i(a_{i-r}^{(T-t+1)}) = a_{i-r}^{(T-t+1)} \), then the LMCA given by (5) is reversible and its inverse is another LMCA with the following local transition function:

\[
a_i^{(T+1)} = \sum_{m=0}^{t-2} f_m(a_{i-r}^{(T-m)}) + a_{i-r}^{(T-t+1)} \pmod{2}, \quad 0 \leq i \leq N - 1.
\]
order to produce stego images $STG_1, ..., STG_n$, for distributing among participants $P_1, ..., P_n$, respectively. There are two procedures: (1) sharing and embedding, (2) authentication and revealing.

### 2.2.1. Sharing and embedding procedure

First, all pixel values in $SI$ that are greater than or equal to 250 are represented as two values, 250 and the value of the difference between the pixel value and 250, respectively. Therefore, a modified secret image ($MSI$) is generated in which pixel values are ranged from 0 to 250. In addition, a secret key $K$ is used to generate a random bit-stream as the watermark bits. This key is further split as the decimal value of $P_z(x)$ with binary representation to produce stego images as coefficients of a polynomial function of degree $p$. Combined with the current watermark bits to produce check bits $S_{ij}$ pixel with binary representation ($0$–$251$). This process results in modified secret images whose size is greater than the original image (see, e.g. [3,27]). The increase in the size of data to be embedded affects the quality of stego images.

### 2.2.2. Authentication and revealing procedure

In order to restore the secret image, any group of $t$ or more stego images with sub-keys are gathered together. Then, these sub-keys can be used to reveal the secret key $K$ from which the watermark bits can be generated. Next, each of the stego images is divided into a set of four-pixel blocks. The authentication bits and the watermark bits are used to compute the check bits corresponding to the block. If the check bits are identical to the hidden bits of the stego image, the block is verified successfully and a shared pixel is retrieved. This procedure is repeated until all the hidden shared pixels are retrieved. Finally Lagrange’s interpolation is used to get the secret image.

### 3. The proposed scheme

In this section, we first outline why we employ CA and then proceed to describe the proposed scheme.

#### 3.1. Our motivation to use cellular automata instead of Shamir’s sharing scheme

Cellular automata are very powerful computational tools with synchronous update mechanism for their individual components (cells) and are specially adopted for image related works. In particular, in secret image sharing applications in which Shamir’s sharing scheme is used, we must provide distinct input values for Shamir’s secret polynomial. Now, to economize on the size of data to be embedded, these values are usually obtained from pixels of cover images. Therefore, as the number of participants increases, it may become necessary to modify some pixels in cover images and this can introduce a side effect on the visual quality of the resulting stego images. Since we need no input values to start evolutions of CA, this effect is removed when using cellular automata which means that pixels of the cover images can be used more efficiently. Another drawback in Shamir-based sharing schemes is that we are forced to calculate “mod a prime number $p$” while CA do not impose such restrictions. In calculating mod primes, we should perform a preprocessing on the secret image to produce pixel values in proper range (i.e. $0$–$251$). This process results in modified secret images whose size is greater than the original image (see, e.g. [3,27]). The increase in the size of data to be embedded affects the quality of stego images.

### 3.2. The scheme

The scheme consists of four phases: (1) the setup phase, (2) the embedding phase, and (4) the verification and recovery phase. In setup phase, a trusted party called the dealer constructs a reversible LMCA ($M$) of order $t$. The dealer further creates a set of public/private keys that will later be used to add signature on the data for the purpose of authentication. In sharing phase, the dealer divides the secret image into some units and derives from each unit $t$ initial configurations necessary to evolve $M$. The evolutions of $M$ are used to produce shared pixels for participants. These shares plus other information about $M$ and necessary data to reconstruct the secret image are then signed by the dealer and embedded in cover images in the embedding phase and the resulting stego images are given to the participants (Fig. 2). The signature is used so that manipulation of stego images can be detected without processing of pixels in verification and recovery phase. Finally, evolutions of the inverse machine are used to recover the original secret (Fig. 4).

#### 3.2.1. Notations

We use the following notations to describe the scheme.

- $SI$ the secret image, $U_1, ..., U_t$ the $(t – 1)$-pixel units of $SI$, $U_1^{t}, ..., U_t^{t−1}$ the pixels in the $i$-th unit $U_i$.
- $Width$, $Height$ width and height of $SI$.
- $P_1, ..., P_n$ the participants.
- $CI_i$ the cover image corresponding to $P_i$.
- $STGi$ the stego image corresponding to $P_i$.
- $Auti$ authentication string corresponding to $P_i$.
- $SH_j$ the share of $P_i$ from unit $U_j$.
- $ω$ the starting rule number of LMCA.
- $C^{(T)}$ configuration of the LMCA at time $T$. 

![Fig. 1. The block $B^k_i$ of the $k$-th stego image in Chang et al.'s scheme.](image-url)
For each $t$, let $D_{Sk}$ be the digital signature on the bitstring $\langle C \rangle$. We assume that for each $t$, the operator divides the bitstring $\langle C \rangle$ from right to left into substrings of length $i$ and then XORs them to obtain a string of length $i$, with padding done if necessary. $DS_k$ is a digital signature on the bitstring $\langle C \rangle$ with key $k$. $Ver_k (\text{Sign}, m)$ is a verification function of the digital signature on message $m$ and signature $\text{Sign}$ with key $k$. $Seq_i$ is a unique sequence number assigned to $P_i$, $H$ a collision-free hash function, $D$ the dealer, $(PU_{PUD}, PR_{PUD})$ public and private keys of the dealer corresponding to signature scheme DS, $\parallel$ the concatenation operator for strings.

### 3.2.2. The setup phase

In this phase, the dealer $D$ fixes some parameters and constructs a reversible LMCA of order $t$ through the following steps:

1. Assigns a cover image $CI_i$ and a sequence number $Seq_i$ to each participant $P_i$.
2. Generates a set of public/private keys $(PU_{PUD}, PR_{PUD})$.
3. Constructs a reversible LMCA $(M)$:
   
   (a) Selects $1 \leq r \leq 3$ as the radius of the symmetric neighborhood of the LMCA.
   
   (b) Selects a random number $0 \leq w_i \leq 2^{r+1} - 1$ and $t - 2$.
   
   (c) Constructs $M$ of order $t$ by
   
   $a_j^{(t-1)} = f_{w_i} \cdot (f_j^{(T)}) + \cdots + f_{w_{i-1}} \cdot (f_j^{(T-2)}) + (f_j^{(T-1)}) (mod 2),$

   where $0 \leq j \leq 7$ and $f_{w_{i-1}}$ is the local transition function of the LMCA with radius $r$ and rule numbers $w_i + i$, $0 \leq i \leq t - 2$.

The pair $(PU_{PUD}, PR_{PUD})$ will be used to prohibit tampering of stego images. The dealer signs the embedded data with $PR_{PUD}$ and hides the signature together with the corresponding public key $PU_{PUD}$ in stego images for the purpose of verification in the recovery phase.

Note that in our scheme, we consider 1 byte for each pixel. Therefore, the number of cells in each configuration of $M$ is 8 and this is why we assume $1 \leq r \leq 3$. Note also that for a rule number $w$ we must have $0 \leq w \leq 2^{r+1} - 1$.

### 3.2.3. The sharing phase

The details of the sharing phase are depicted in Figs. 2 and 3. First, $D$ divides $SI$ into units $U_1, U_2, \ldots, U_t$, where $U_j$ consists of $(t-1)$ pixels $U_j^1, \ldots, U_j^{t-1}$ (with padding done on the last unit if necessary) (Fig. 2). For each $U_j$, the dealer employs its pixels as initial configurations $C^{(0)}_j, C^{(t-2)}_j$ of $M$. The $r$-th initial configuration $C^{(r)}_j$ is then computed as the hash value of the first $t - 1$ initial configurations.

Finally, the evolutions of $M$ are used to create $n$ shares $SH_1^j, \ldots, SH_n^j$ corresponding to participants. The details are as follows:

**Step 1.**

1. Divides $SI$ into $(t-1)$-pixel units $U_1, U_2, \ldots, U_t$.
2. Repeats for $j = 1, \ldots, t$.
   
   (i) Sets initial configurations $C^{(0)}_j, C^{(t-2)}_j$ of $M$ as the binary representation of $t - 1$ pixels in the unit $U_j$: $(U_j^1, \ldots, U_j^{t-1})$.
   
   (ii) Computes $C^{(r)}_j$ as $(H(C^{(0)}_j), \ldots, H(C^{(t-2)}_j)) \parallel$. The signature and the corresponding public key $PU_{PUD}$ are concatenated to form the authentication string $Aut_j$ which will also be embedded in $CI_i$ to prevent $P_i$ from manipulating his/her share.

   Considering the initial configuration $C^{(t-1)}_j$ as $(H(C^{(0)}_j), \ldots, H(C^{(t-2)}_j)) \parallel$ provides what we call double authentication. Even if the cheaters succeed in forging the dealer's signature on stego images and therefore pass the first authentication check (with very small probability), their manipulation will be revealed by this technique. We elaborate on how this is done in the verification and recovery phase.

   Note that $Seq_j$ is the sequence number devoted to $P_j$ in the setup phase. It is used to prevent participants from exchanging their shares or announcing an incorrect identification number (that is not consecutive) in the recovery phase.

**Step 2.**

1. Repeats for $i = Seq_1, \ldots, Seq_n$.
   
   (i) $\text{Sign}_i = DS_{PR_{PUD}}(PU_{PUD} \parallel \text{Height} \parallel w_i \parallel t \parallel SH_1^i \parallel SH_2^i \parallel \ldots \parallel SH_n^i)$. 
   
   (ii) $Aut_i = PU_{PUD} \parallel \text{Sign}_i$.

So far, with $l$ as the total number of units of $SI$, the string $SH_1^i \parallel SH_2^i \parallel \ldots \parallel SH_n^i$, plus other information about $M$ and $SI$ must be embedded in the $P_i$'s cover image ($CI_i$). In this step, the dealer signs these information with his private key $PR_{PUD}$. The signature and the corresponding public key $PU_{PUD}$ are concatenated to form the authentication string $Aut_i$ which will also be embedded in $CI_i$ to prevent $P_i$ from manipulation of his/her share.

### 3.2.4. The embedding phase

In this phase, the dealer produces final stego images by embedding the data obtained in previous phases into the cover images. The embedding is such that first the visual quality of the results have no serious downturn, and second it is difficult to recognize that any data is hidden in the stego images. Our embedding procedures satisfies both of these requirements.
As described in previous phases, we need to embed in each CI<sub>i</sub> the following data:

- Seq<sub>i</sub>, to ensure consecutive ordering of the participants in the recovery phase.
- wis, r, t, for reconstruction of the LMCA M.
- Width, Height and l, for proper recovery of the secret image.
- The shares assigned to P<sub>i</sub>, i.e. SH<sub>j</sub><sup>i</sup>, 0 ≤ j ≤ l.
- The authentication string Aut<sub>i</sub> corresponding to P<sub>i</sub>, for the purpose of authentication.

We now outline the details of embedding procedure in the i-th cover image CI<sub>i</sub>. The forgoing data, with the same ordering, are considered as an array of bytes. Each byte of data is embedded into one block of CI<sub>i</sub> consisting of 4 bytes. Let (d<sub>1</sub>, ..., d<sub>8</sub>) be the binary representation of the byte to be embedded in block B of CI<sub>i</sub> with pixels X, V, W and Z with binary representation as in Fig. 5. The embedding replaces the least significant bits of X, V, W and Z with d<sub>1</sub>, ..., d<sub>8</sub> as depicted in Fig. 6. Note that the embedding changes at most two of the LSBs in each byte of B. This maintains the quality of stego images.

3.2.5. The verification and recovery phase

The details of this phase are depicted in Fig. 4. Suppose that t consecutive participants, P<sub>x+1</sub>, ..., P<sub>2t+1</sub>, 1 ≤ x ≤ n−t+1, present the stego images STG<sub>x+1</sub>, STG<sub>x+2</sub>, ..., STG<sub>2t+1</sub> to recover the secret
image SI. Each STG_i is divided into a set of blocks with four pixels from which the embedded data can be retrieved.

(1) Retrieve from STG_i, Seq_i, ws, r, t, Height, Width, l, SH^l, 0 ≤ j ≤ l and Aut_j = PU_D||Sign_i for each P_j, x ≤ j ≤ x + t − 1.

(2) Check if the derived sequence numbers Seq_i are consecutive.

(3) Each P_j, x ≤ j ≤ x + t − 1 can verify the information given by P_j, j ≠ i, as follows:
   • PU_D of P_j should equal PU_D of P_i, for x ≤ j ≤ x + t − 1.
   • Ver_PUD[Sign_i, PU_D||Width||Height||||Seq_i||SH^l_i||SH^2_l||...||SH^t_l] = true.

(4) Repeat for j = 1, ..., l (to reconstruct the j-th unit):
   (4.1) Construct the inverse of M, i.e., M, by Eq. (6), with radius r, rule numbers determined by ws, and initial configurations
   \[C^{(0)} = SH^l_{x+1-1}, C^{(1)} = SH^l_{x+2-1}, ..., C^{(t-1)} = SH^l_r,\]
   and evolve \(\hat{M}, t + x - 1\) times to obtain \(C^{(2t+x-2)}, ..., C^{(t+x-1)}\).

(4.2) Check if \(\langle \text{Hash}(C^{(2t+x-2)}, ..., C^{(t+x-1)}) \rangle_8 = \text{Hash}(C^{(t+x-1)})\) or not.

(4.3) The pixels of the j-th unit of SI, that is, \(U_j^1, ..., U_j^{t-1}\), are taken as the binary representation of \(C^{(2t+x-2)}, ..., C^{(t+x-1)}\), respectively.

The stego images which fail the signature verification test are tampered. We have the option to totally reject them and stop, or mark them suspicious and proceed with the recovery phase. Now, tampered stego blocks which are input to \(\hat{M}\) would satisfy, with probability 1/256, the test in step 4.2 of the verification phase and the corresponding unit of the secret image will be invalidated with probability 1/256.

Note that since the shares presented in recovery phase constitute configurations of a reversible t-th order LMCA, then at least t shares are required to compute evolutions of the inverse automata and knowing t − 1 or fewer shares is insufficient.

4. Comparison and experimental results

In [3], the scheme of Chang et al. is shown to outperform the methods presented in [1,2]. Therefore, in this section, we compare the proposed scheme and the scheme of Chang et al. The results of this section show that our proposed method achieves the following merits:

- Our method employs linear cellular automata while best polynomial evaluation and interpolation algorithms are of order \(O(n \log^2 n)\) [24].
- We use at most 2 bits in each pixel of cover images for embedding data while this number is 3 for Chang et al.’s scheme. Moreover, since we employ cellular automata, we are not forced to change any more bits in cover images as Shamir-based methods do. This guarantees a better visual quality for our stego images. We perform some experiments for illustration.
- Given a stego image, our scheme detects whether it is tampered or not without processing of its blocks. Other methods may need to process all blocks of the image to find this. Moreover, each fake stego block is verified successfully with probability 1/256 in contrast to 1/16 for Chang et al.’s method. Experimental results are also provided for comparison.

We first consider visual quality of stego images. We perform experiments for \(n = 4\) and \(t = 3\). We take “Jet-F16” with size 256 × 256 as the secret image while “Lena”, “Pepper”, “Baboon” and “flower” serve as cover images with sizes 512 × 512 (Fig. 7). The criterion for the visual quality of the stego images is the peak-signal-to-noise ratio (PSNR) defined as

\[\text{PSNR} = 10 \times \log_{10} \left( \frac{255^2}{\text{MSE}} \right) \text{dB},\]  

where MSE is the mean-square error between the cover image and the stego image. If the cover image is sized \(f \times g\), MSE is defined as

\[\text{MSE} = \frac{1}{f \times g} \sum_{i=1}^{f} \sum_{j=1}^{g} (x_{ij} - y_{ij})^2,\]  

where \(x_{ij}\) and \(y_{ij}\) denote the cover and the stego pixel values, respectively. The stego images generated by the scheme of Chang et al. and their PSNR are given in Fig. 8(a). The same data are reported for our scheme in Fig. 8(b). The results show that our scheme achieves higher PSNR.

We now compare the two schemes in terms of authentication capabilities. In the scheme of Chang et al., the probability that a fake block can be verified successfully is 1/16. Now, suppose that \(P_i\) modifies a block of his/her stego image. Therefore, the signature \(Sign_i\) hidden in \(STG_i\) cannot be verified successfully. Next, the fake blocks in this stego image are used to compute the evolutions of \(M\) and obtain \(C^{(2t+x-2)}, ..., C^{(t+x-1)}\). The modification makes \(\langle \text{Hash}(C^{(2t+x-2)}, ..., C^{(t+x-1)}) \rangle_8\) different from \(C^{(t+x-1)}\) and the fake block is detected. Hence, the probability that a modified block is verified successfully is about 1/256. Note also that our scheme can detect a tampered stego image with a single test while Chang et al.’s scheme should obtain this information from tampered blocks. We also conducted experiments to confirm authentication abilities of the two schemes. To do this, we follow the method used in [3] and consider \(DR = \frac{NTPD}{NTP}\), the criterion for integrity verification, where \(NTP\) is the number of the tampered pixels, and \(NTPD\) is the number of the tampered pixels that are detected.

We consider again the stego image “lena” constructed by the two schemes and manipulate it twice with two different images: “flower” with size 120 × 112, and “Pepper” of size 120 × 148. These images are added to the top-left corner of the victim stego image, as
shown in Fig. 9. The detection ratios for tampering with “flower” are 0.951 and 0.997 for Chang et al.’s scheme and our proposed scheme, respectively, while tampering with “Pepper” reports 0.974 and 0.997 for the corresponding schemes.

5. Conclusions

In this paper, we propose a new \((t, n)\)-threshold image sharing scheme with steganographic properties. The scheme is based on linear cellular automata and introduces no distortion to the original secret image; however, consecutive shares must be provided to recover the secret image. We change at most 2 bits in each pixel of a given cover image to preserve its visual quality. In order to manipulate even a single block in a stego image and remain unnoticed, a malicious participant has to forge a secure digital signature and even in that case, a double authentication mechanism is employed such that the corresponding information in the secret image will be invalidated with probability 255/256. Moreover,
Fig. 9. Manipulated stego images for the two schemes. (a) Chang scheme (DR = 0.951), (b) our scheme (DR = 0.997), (c) Chang scheme (DR = 0.974), (d) our scheme (DR = 0.997).

the integrity of each stego image can be verified without processing of its blocks.

References


About the Author—ZIBA ESLAMI received her B.S., M.S., and Ph.D. in Applied Mathematics from Tehran University in Iran. She received her Ph.D. in 2000. From 1991 to 2000, she was a resident researcher in the Institute for Studies in Theoretical Physics and Mathematics (IPM), Iran. During the academic years 2000–2003, she was a Post Doctoral Fellow in IPM. She served as a non-resident researcher at IPM during 2003–2005. Currently, she is the Head of and a Professor in the Department of Computer Sciences at Shahid Beheshti University in Iran. Her research interests include design theory, combinatorial algorithms, cryptographic protocols, and steganography.

About the Author—SEYYED HOSSEIN RAZZAGHI received his B.S. degree in Management Information System (MIS) in 2006 from Tabriz University. He is currently an M.S. student of Computer Science in Shahid Beheshti University, Tehran, Iran.

About the Author—JAMAL ZAREPOUR AHMADABADI received his B.S. degree in Management Information System (MIS) in 2006 from Yazd University. He is currently an M.S. student of Computer Science in Shahid Beheshti University, Tehran, Iran.