Chapter 6

Application of Solving Sudoku Instances

Overview

In this chapter, we like to present some exceptionally important spheres of application, where we may find that how much intrinsic occurrence in solving a given Sudoku instance might be. This is also indicated by different groups of previous researchers who have pointed out mostly the same in last few years of their work. We also have gone through most of these domains of the application area, and pointed out two new areas, namely the Biometric [19] field and the field of Steganography [18], where we have applied our newly developed algorithms designed and introduced in Chapters 3 and 4 of this thesis. In the biometric domain, we have also applied the ideas behind the instance generation techniques described in Chapter 5.

6.1 Application Domains of Solving Sudoku Instances

In this section, we highlight some important fields, where Sudoku puzzle can be used for better performance. After going through several research articles, we have found that Sudoku puzzle have vast application in the following domains that are briefly discussed below.

A. Steganography

The word Steganography is of Greek origin and means concealed writing. Steganography is the art of hiding the existence of information within seemingly harmless carriers such as image, video, or audio. A message in cipher text may arouse suspicion while an invisible message is not. A digital image is a flexible medium used to carry a secret message because a minor modification of some cover image is hard to distinguish by human eyes. But the main drawback of this type of schemes is that, when it is revealed, the covert image may get distorted due to truncation of the greyscale secret. Using an 18×18 Sudoku puzzle and dividing it into eighteen 3×3 blocks we get as minigrids, may provide better quality in revealing the images [64]. This domain of application is advantageous, when the size of the puzzle is small, but unfortunately none of the methods is capable to detect if a cover image gets modified. If the cover image is changed, the message embedded inside the image is also changed.
B. Visual Secret Sharing

*Visual secret sharing* refers to the method for distributing of undisclosed information amongst a group of participants, each of whom is allocated a share of secret, known as *shadow*. The hidden information can only be reconstructed when a sufficient number of shares are combined together, where individual shares are of no use on their own. Sudoku puzzle logic can be used for dividing the secrets into shadows and combining those shadows to reconstruct the original secret information [20].

C. Short Message Service (SMS)

*Short Message Service* (SMS) is one of the popular services provided by mobile telephony system today. In this service people can write short messages and send to a receiver through mobile communications. Securing of short message services is greatly important as there are possibilities of revelation of confidential and personal document during exchange of information among various systems. Better security in hiding the SMS data can be achieved by using Sudoku puzzle and some Steganographic algorithms. The puzzle encapsulating the message is then sent in a way that it does not attract any attention of some possible intruder. After solving the Sudoku puzzle on the receiving end, one may extract the hidden data in accordance with order of numbers 1 through 9 in the sequence of certain rows and/or columns, which is equivalent to the hidden information [22].

C. Digital Watermarking

Distribution of digital data over public networks such as the Internet is not really safe due to copyright violation, counterfeiting, forgery, and deception. Therefore, the protection of digital data, especially for confidential data, is an important contemporary issue. Suitable watermarking scheme may be used to protect the data from copyright violation. But in most of the digital watermarking schemes, hiding high volume of data affects image quality to a considerable extent. Interestingly it has been claimed that if we use the logic of Sudoku puzzle to create a reference matrix for *digital watermarking*, we can attain both the issues [21].

D. Image Authentication

*Image authentication* techniques (or algorithms) have recently attracted researchers’ interest due to their applications in various emergent, military, medical science application areas. Nowadays,
images in digital format are often transmitted over unsecured channels like Internet. So this initiates an urgent need to protect these digital images from active attacks that can tamper the transmitted images. Using the logic of Sudoku puzzle it has been claimed that the part(s) of an image can easily be identified where the tampering have been attempted [23].

C. Data Security

Digital communication among computers and networks paves the highway of information processing. On this highway, there are different types of digital data, which require different security levels according to the sensitivity of the information. To achieve these goals different data encryption techniques have been adopted. Encryption algorithms convert original data (plain text) into encrypted data (cipher text) such that the encrypted data is entirely different from the original one. Different ciphers have been used for this purpose, but the traditional block ciphers such as DES, IDEA, and RSA are not suitable for image data encryption. The bulk data property of some image data makes this conventional method inefficient [24]. Using the rectangular Sudoku matrices of size 4×5 each, a new cipher can be designed that could be used to encrypt an image data in a very efficient manner [65].

D. DNA Sequencing

The mixing and simultaneous sequencing of large number of DNA samples is known as multiplexing. In earlier multiplexing approaches, scientists first tagged each sample with a barcode—a short string of DNA letters known as oligonucleotides, before mixing it with other samples that also had unique tags. A unique barcode generator for each sample, prior to sequencing is a very time consuming and costly process, especially if the numbers are in the order of thousands. In order to overcome this limitation, Erlich and others in the Hannon laboratory came up with an idea of mixing the samples in specific patterns, and thereby creating pools of samples. Instead of tagging the individual samples contained by each pool, the scientists tagged each pool as a whole with one barcode. They have used the logic and number placement rules of Sudoku puzzle for creating the pooling strategy [25].

E. Air Track Maintenance

Maintenance of air tracks is another important domain of applications of Sudoku puzzle, which is essential for the smooth and safe aircraft operation. It includes getting customary aerial
pictures of each air track which can be obtained from the radars present on these zones of the tracks. Therefore, there may be the case that the adjacent radars may chronologically capture the same images of a particular section of the tracks, which leads to redundancy of data and improper bandwidth utilization for sending the data. So, identifying the zones of particular radar is greatly essential. Here, the Sudoku logic can properly be utilized to divide the zones of the tracks in proper way, so that we can overcome these limitations [66].

In this section, we have discussed about the domains where we can find the application of solving Sudoku instances. In the next section, we are going to explore some new algorithms which we have developed, where we can find the application of solving Sudoku instances. Essentially, we have developed two new algorithms for the Biometric and Steganographic domain.

6.2 A Novel Biometric Template Encryption Scheme using Sudoku Puzzle

Identity theft is a growing concern in this digital era. As per the US Federal Trade commission, millions of people are victimized in every year [67]. Traditional authentication methods such as passwords and identity documents are not sufficient to combat ID theft or ensure reliable security system. Such form of identity representations can often be forgotten by the person concern, lost by him, guessed, stolen, or shared with others. On the contrary, biometric systems of identification recognize individuals based on their anatomical traits (e.g., fingerprint, face, palm print, iris, and voice) or behavioral traits (e.g., signature, gait). As such traits are physically linked to the user, biometric recognition is a natural and more reliable mechanism for ensuring that only legitimate or authorized users are able to enter a facility, access a computer system, or cross international borders. Biometric systems also offer unique advantages such as deterrence against repudiation and the ability to detect whether an individual has multiple identity cards (e.g., passports) under different names. Thus, biometric systems impart higher levels of security in case of user recognition and/or authentication. In this chapter, an attempt has been made to use Sudoku puzzle for securing the biometric data. This work is published in the reference given in [19].

Biometric authentication schemes have great potentials in building secured systems since biometric data of the users are bound tightly with their identities, and cannot be forgotten. Typically, a biometric authentication scheme consists of two phases, i.e., enrolment phase and
During the enrolment phase, the sensor module acquires the raw biometric data of an individual in the form of an image, video, audio, or some other signal. The feature extraction module operates on the biometric signal and extracts a salient set of features to represent the signal; during user enrolment the extracted feature set, labeled with the user’s identity, is stored in the biometric system and is known as a template. This template is stored on some central server or a device.

During the authentication phase, user would provide another biometric sample to the sensor. Features extracted from this sample constitute the query, which the system then compares to the template of the claimed identity via a biometric matcher. The matcher returns a matched score representing the degree of similarity between the template and the query. The system accepts the identity claim only if the matched score is above a predefined threshold. The whole scheme is presented in Figure 6.1, where X and Y are biometric templates as discussed below.

Figure 6.1: Sketch generation and template reconstruction of biometric information.

Now we briefly explain the activities performed in Figure 6.1. During the enrolment phase of biometric authentication, the images of the traits are captured using some camera or similar devices. Then features such as characteristics of the captured image such as color, pattern of the image, etc. as required by the algorithm are extracted. Then a biometric template (say, X) is extracted from it. This template contains the required characteristics of the user’s biometric information that could be used to identify the said trait uniquely. Further a sketch is generated from the template (i.e., X). A sketch is essentially a graph like structure from which the template
can be reconstructed, as and when necessary. The generated sketch is stored in the database. This part follows the upper row of the figure up to database. Now during the authentication phase, as usual at first the trait’s image is captured; then in the same way features are extracted from it and another template (say, Y) is constructed. Then the template X is reconstructed from the sketch stored in the database. Afterward, it is matched with the captured template (i.e., Y) by using some matching function. If the templates match up to a predefined level, then we may say that the user is authenticated; otherwise, not.

Thus, a biometric system may be viewed as a pattern recognition system whose function is to classify a biometric signal into one of several identities (viz., identification) possible or into one of two classes—genuine and impostor users (viz., verification). While a biometric system can enhance user convenience and security, it is also susceptible to various types of threats [69, 70, 71] as discussed below in the next section.

6.2 Biometric System Vulnerabilities

A biometric system is vulnerable to two types of failures [72]. A denial of service occurs when the system does not recognize a legitimate user, while an intrusion refers to the scenario in which the system incorrectly identifies an impostor as an authorized user. While there are many possible reasons for these failures, they can broadly be categorized as intrinsic limitations and adversary attacks.

6.2.1 Intrinsic Limitations

Unlike a password based authentication system, which requires a perfect match between two alphanumeric strings, a biometric based authentication system relies on the similarity between two biometric samples. This is because an individual’s biometric samples acquired during enrolment and authentication are seldom identical; a biometric system can make two types of authentication errors. A false mismatch occurs when two samples from the same individual have low similarity and the system cannot correctly match them. A false match occurs when two samples from different individuals have high similarity and the system incorrectly declares them as a match. A false mismatch leads to a denial of service to a legitimate user, while a false match can result in intrusion by an impostor. This is because the impostor need not exert any special effort to fool the system; such an intrusion is known as a zero-effort attack. Most of the research
endeavour in the biometrics community over the past five decades has focused on improving authentication accuracy—that is, on minimizing false mismatches and false matches.

### 6.2.1.2 Adversary Attacks

A biometric system may also fail to operate as intended due to manipulation by adversaries. Such manipulations can be carried out via insiders, such as system administrators, or by directly attacking the system infrastructure. An adversary can circumvent a biometric system by coercing or colluding with insiders, exploiting their negligence (for example, failure to properly logout of a system after completing a transaction), or fraudulently manipulating the procedures of enrolment and exception processing, originally designed to help authorized users.

External adversaries can also cause a biometric system to fail through direct attacks on the user interface (sensor), the feature extractor and matcher modules, the interconnections between the modules, and the template database.

Examples of attacks targeting the system modules and their interconnections include Trojan horse, man-in-the-middle, and replay attacks. As most of these attacks are also applicable to password based authentication systems, several countermeasures like cryptography, time stamps, and mutual authentication are available to prevent them or minimize their impact. Two major vulnerabilities that specifically deserve attention in the context of biometric authentication and ID cards, it is not possible to replace stolen templates with new ones because biometric traits are irrevocable. Finally, the stolen biometric templates can be used for unintended purposes—for example, to covertly track a person across multiple systems or obtain private health information.

There are several possible reasons for these attacks. One of the most common reasons behind this type of attack is stealing of templates and modifying them [70]. So, securing the biometric template is immensely important. At the same time the quality of the template should not be degraded. In our proposed scheme, we have used the puzzle of a 9×9 Sudoku instance as key to encrypt the biometric template which can overcome the said difficulties. The novelty of our proposed scheme lays on the fact that encryption of the template with less distortion as well as it is expected to be incredibly difficult for the intruder to know about the keys and the intruder will not be able to change or distort the template.
6.2.2 Existing Biometric Template Encryption Scheme

An ideal biometric template encryption scheme should have the following three properties [71]:

**Revocability:** The biometric template should be encrypted in such a manner that, we can easily revoke the compromised template.

**Security:** It must be computationally hard to obtain the original biometric template from the secured template, so that the intruder should not be able to reconstruct the template.

**Performance:** The biometric template encryption scheme should not degrade the quality of the template.

The existing biometric template encryption scheme can be broadly classified into two categories: (1) the biometric cryptosystem approach and (2) the feature transformation approach as shown in Figure 6.2. The basic idea of these approaches is that instead of storing the original template, the transformed / encrypted template which is intended to be more secured, is stored. In case the transformed / encrypted template is stolen or lost, it is computationally hard to reconstruct the original template and to determine the original raw biometric data simply from the transformed / encrypted template.

![Figure 6.2: Categorization of biometric template protection scheme.](image)

In the feature transformation approach, a transformation function \( F \) is applied to the biometric template \( T \), and only the transformed template \( F(T, K) \) is stored in the database. The parameter of the transformation function is typically derived from a random Key \( K \) or password. The same transformation function is applied to query features \( Q \) and the transformed
query \((F(Q, K))\) is directly matched against the transformed template \((F(T, K))\). Depending on the characteristics of the transformation function \(F\), the feature transformation schemes can be further categorized as salting and noninvertible transforms. In salting, \(F\) is invertible, that is, if an adversary gains access to the key and the transformed template, he/she can recover the original biometric template (or a close approximation of it). Hence, the security of the salting scheme is based on the secrecy of the key or password. On the other hand, noninvertible transformation schemes typically apply a one-way function on the template and it is computationally hard to invert a transformed template even if the key is known. Figure 6.3 depicts the approach as described.

**Figure 6.3:** Authentication mechanism when the biometric template is protected using feature transformation approach (courtesy to [70]).

Biometric cryptosystems [73, 74] are originally developed for the purpose of either securing a cryptographic key using biometric features or directly generating a cryptographic key from biometric features. However, they can also be used as a template protection mechanism. In a biometric cryptosystem, some public information about the biometric template is stored. This public information is usually referred to as helper data, and hence biometric cryptosystems are also known as the methods based on helper data [75]. While the helper data does not (is not supposed to) reveal any significant information about the original biometric template, it is needed during matching to extract a cryptographic key from the query biometric features. Matching is performed indirectly by verifying the validity of the extracted key. Error correction coding techniques are typically used to handle intruder variations. Figure 6.4 depicts the approach as described above.
Biometric cryptosystems can further be classified as key binding and key generation systems depending on how the helper data is obtained. When the helper data is obtained by binding a key (that is independent of the biometric features) with the biometric template, we refer to it as a key-binding biometric cryptosystem. Note that given only the helper data, it is computationally hard to recover either the key or the original template. Matching in a key-binding system involves recovery of the key from the helper data using the query biometric features. If the helper data is derived only from the biometric template, and the cryptographic key is directly generated from the helper data and the query biometric features, it leads to a key generation biometric cryptosystem.

6.2.3 Biometric Template Encryption Scheme using Sudoku

The existing template encryption scheme can adequately provide security to the biometric template. But it fails, if somebody modifies the stored template. In our proposed investigation scheme, we have applied Sudoku puzzle as a key to encrypt the template, so that any alteration in the template becomes obviously detectable. The entire scheme is described as below:

**Input:** A solved Sudoku puzzle and the biometric template of a trait (e.g., an image).

**Output:** Sudoku embedded biometric template.

**Step 1: Block preparation:** The biometric template is divided into 9×9 equal sized blocks.

**Step 2: Embedding 9×9 Sudoku puzzle:** A solved Sudoku instance and the biometric template with blocks are taken as input.
For each individual block

Make disjoint groups of four pixels each; padding is incorporated, if necessary.

For each group of four pixels

The least significant bit (LSB) of the 8-bit representation of each pixel in block $i$ is added to the associated value present in the corresponding Sudoku cell $i$, $1 \leq i \leq 81$.

As for example, in Figure 6.5, all pixels present in the first block of the first row are to be replaced with 3 in a group of four pixels each, i.e., the LSBs of the first and the second pixel are kept unchanged, whereas 1 is added to the LSB of each of the third and the fourth pixel, as the corresponding binary representation of 3 is 0011.

Step 3: Sudoku instance and key generation: Here a suitable Sudoku instance is generated (by excluding digits from some of the cells and making them blank) from a given solved Sudoku puzzle. Hence, a Sudoku instance is generated by keeping some of the digits as clues and rest of the cells remain blank so that at the end of the process the same solved Sudoku puzzle could be obtained. This process is known as the Digging Hole method for generating Sudoku instance [15]. This method is discussed in detail in Section 5.2 of this thesis. In Figure 6.6, the clues are shown in red colours that are generated (in obtaining an instance of the problem) for some solved Sudoku puzzle, as shown in Figure 6.5. This also can be a different instance other than that has been depicted in this figure.

Then two keys are created from this Sudoku instance: (1) Server key and (2) User key. Server key is created after removing clues from each corner cell. User key is created by storing all the removed corner digits (rowwise); if there is no clue, then 0 is stored as the corresponding key of the cell. For example, in Figure 6.6, the user key we obtain is 0850, as there are no clues in the top-left corner cell as well as bottom-right corner cell; we may note that the top-right corner cell contains an 8 whereas the bottom-left corner cell contains a 5 as clues. In our scheme, we like to keep the server key retained in the database server, while the user key is returned back to the user for future authentication, as and when necessary.

Then the biometric sketch is created, and this sketch along with the server key is stored in the server. So, in our proposed scheme, the server stores encrypted biometric template sketches in the database as well as the server key generated using the Sudoku instance. The user key is kept
by the biometric user. If anybody wants to enter into the biometric system, has to supply the biometric information along with the key generated by the server. The decryption technique of the proposed scheme is as follows.

**Figure 6.5:** Biometric template encryption process: The biometric template is divided and placed over a region of 9×9 blocks. The values of each cell of a (solved) Sudoku puzzle are embedded into the corresponding block of the template. Here red digits are given clues of the problem instance and black digits are inserted to get a solved solution of the instance.

**Figure 6.6:** The process of key generation.

### 6.2.4 Biometric Template Decryption Technique

**Step 1: Submission of keys:** User has to submit its own key along with biometric data to the server. The server places each number sequentially to each corner of the server key as
shown in Figure 6.7. The first leftmost value is placed in the top-left corner of the server key. The next value is placed in the top-right corner, whereas the next two values are placed in the bottom-left and bottom-right corner of the server key, respectively. Then the original instance of the Sudoku is computed, wherefrom we can reach to the original solved Sudoku puzzle, which is ultimately embedded in the template. If 0 is found in the user key, then the value is replaced by a blank cell in the server key.

**Step 2:** Here the Sudoku instance is solved by the server to get the complete Sudoku puzzle.

**Step 3:** Then in the similar way during the encryption process, the reconstructed templates are divided into 81 blocks.

**For each individual block:**

Make groups of four pixels each.

**For each group:**

The associated value present in a Sudoku cell is subtracted from the least significant bit (LSB) of each pixel present in the corresponding group of the block, and then padding is also removed, if added earlier.

Then the original biometric template (i.e., $X$, in Figure 6.1) is recreated.

**Figure 6.7:** Merging of Server key with User key.
Now this is obvious that an efficient Sudoku solving algorithm is also very essential to speed up the process of encryption and decryption. Hence, the Sudoku solvers developed in the thesis (in Chapters 3 and 4) can be applied for this purpose.

6.2.5 Experimental Results: Analysis and Discussion

Here, we have tested our developed an approach of biometric template encryption scheme, using the Essex Faces 94 face database (E94 database), which is publicly available and essentially created for face recognition related research studies [76]. The E94 database contains images of 152 distinct subjects, with 20 different images for each subject where the size of each JPEG image is 180×200. In our approach, we have transformed each of these images to a matching 8-bit gray level image and then used the gray level image in our experiment. For each subject, we have randomly chosen 12 out of 20 samples for enrolment and the remaining 8 sample face images are used for authentication. Some sample images from the E94 database are given in Figure 6.8.

Figure 6.8: Sample E94 database.

Then we have done histogram analysis of this proposed scheme. We have noticed a very less distortion in the template. Sample histograms are shown in Figures 6.9 and 6.10; Figure 6.9 shows the histogram of the template before embedding Sudoku and Figure 6.10 shows the same after embedding Sudoku.
From the histogram analysis, we can eventually notice that there is incredibly less distortion in the image after embedding the Sudoku puzzle, which is one of the prime requirements of any template encryption scheme.

The advantages of our proposed scheme that has been developed in this chapter are discussed in brief in next subsequent sections.

**A. Computation Time**

As we have performed only simple add operation on the LSB’s, it takes constant amount of computational time (as the size of the Sudoku puzzle under consideration is bounded by a constant). The average computational time for the operation is around 30 msec, which is small enough in comparison to other encryption techniques. Feature transformation takes in an average of 30-35 msec, whereas biometric cryptosystem takes in an average of 40-45 msec of computation time.

**B. Robustness**

In feature transformation approach, a noninvertible feature transformation function is applied to the biometric template whereas in the case of biometric cryptosystems biometric templates are encrypted using keys. But still there are scopes to modify the template which ultimately leads to
DOS (Denial-of-Service) attack. In our devised scheme, we are embedding a Sudoku puzzle inside the biometric template and each such Sudoku puzzle is of size 9×9 only having the numbers 1 through 9 in each row, column, and minigrid exactly once. Thus, no modification is possible in this approach, as any modification in the template in due course leads to modification in the Sudoku puzzle itself, which leads to the violation of Sudoku constraints.

C. More Number of Keys

There exist as many as 6,670,903,752,021,072,936,960 distinct Sudoku puzzles of size 9×9 each [15]. That means these many numbers of different keys can be used. Now the average computing time of a Sudoku instance is of the order of 29 msec; that means, around 6,134,456,139,289 years are required to solve all the Sudoku instances available in practice. So guessing the key is almost impossible in our devised method, and the brute force attack is also not feasible in our projected scheme of encryption. Thus, this approach leads to an invincible encryption scheme.

Our anticipated encryption scheme is novel in the following sense. As we are embedding a Sudoku puzzle inside the template it is almost impossible for an intruder to modify the template as each row, column, and minigrid of the Sudoku instances containing 1-9 uniquely. Any changes in these data ultimately lead to an error in the Sudoku puzzle. If somebody changes the whole Sudoku puzzle, then also the user of the system is capable to find the alteration, as the original Sudoku instance is stored as key in the server as well as with the user. We are modifying the least significant bit (LSB) of each pixel. Hence, the probability of image distortion is significantly reduced. We have distributed the key used for the encryption into two parties, i.e., user and receiver server. So, without getting these two keys nobody will be able to decrypt the template. Therefore, we can claim that the proposed scheme is exceedingly robust with least distortion in the quality of the template.

We have also developed an application of Sudoku puzzle in the Steganographic domain. In the next section, we are going to discuss about it.

6.3 A Novel Steganographic Scheme using Sudoku

As has already been discussed, the word ‘Steganography’ is derived from Greek words ‘stegos’ meaning ‘cover’ and ‘graphia’ meaning ‘writing’ [77]. Steganography conceptually implies that the message to be transmitted is not visible to the normal eye. It is an art of hiding information
inside a medium. The main objective of Steganography is mainly concerned with the protection of contents in the form of the hidden information. A message in cipher text may arouse suspicion while an invisible message is not. A digital image is a flexible medium used to carry a secret message because the slight modification of a cover image is hard to distinguish by human eyes.

In this section of the thesis, we have proposed a Steganographic scheme using Sudoku puzzle. The novelties of the work lies that nobody could think of and modify the cover image during transmission where the message is embedded using a Sudoku reference matrix. The work is published in [18].

Steganography is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows the existence of the message. The following formula provides a very generic description of the pieces of the Steganographic process [77]:

$$\text{cover\_medium} + \text{hidden\_data} + \text{stego\_key} = \text{stego\_medium}$$

In this context, the cover\_medium is the file in which we hide the hidden\_data, which may be encrypted using the stego\_key. The resultant file is known as stego\_medium (which is, of course, the same type of file as the cover\_medium). Here, the cover\_medium and the stego\_medium are typically image or audio files. In this investigation, we have focussed on image files and, therefore, the cover\_medium and stego\_medium are referred to as the cover\_image and stego\_image, respectively.

Before discussing as how information is hidden in an image file, it is worth to have a fast review of how images are stored in a computer. An image file is merely a binary file containing a binary representation of the colour or light intensity of each picture element (pixel) which is the only constituent of an image.

Images typically use gray and colour levels which can be represented by 8-bit or 24-bit (or by some other suitable value) [64]. When using 8-bit colour, there is a definition of up to 256 colours forming a palette for this image, each colour is denoted by an 8-bit value. In a similar fashion, in a 24-bit colour scheme, as the term suggests, uses 24 bits per pixel and provides a much better set of colours (as there are $2^{24}$ colour combinations now). In this case, each pixel is represented by three bytes, each byte representing the intensity of the three primary colours red, green, and blue (RGB), respectively.
The simplest approach in hiding data within an image file is called the least significant bit (LSB) insertion [78]. In this method, we can take the binary representation of the hidden_data and overwrite the LSB of each byte within the cover_image. If we are using 24-bit colour, the amount of change is minimal and indiscernible to the human eye. As an example, suppose that we have three adjacent pixels (nine bytes) with the following RGB encoding:

```
10010101 00001101 11001001 | 10010110 00011111 11001010 | 10011111 00010000 11001011
```

Now suppose we want to hide the following nine bits of data (the hidden data is usually compressed prior to being hidden): 101101101. If we overlay these nine bits over the LSB of the nine bytes above, we get the following (where the red bits in bold have been changed):

```
10010101 00001100 11001001 | 10010111 00001110 11001010 | 10011111 00010000 11001011
```

One of the potential problems of this type of method is that any intruder can easily modify the cover image. Then it will be impossible for the receiver to extract the hidden message in the cover image. So, detecting whether a cover image is modified during transmission is very much important.

In our proposed scheme, we have embedded an 8×8 Sudoku puzzle into the cover image to prevent and detect the modification in the cover image, if any. Along with it, we have introduced an 18×18 Sudoku reference matrix as a key for message embedding into the cover image.

### 6.3.1 Existing Steganographic Scheme using Sudoku

Initial work on Steganography using Sudoku puzzle was done by Chang, Chou, and Kieu [79]. The basic idea in the method is to use a Sudoku puzzle to generate a reference matrix (M) and alter the values at selected pixels in the cover image according to the values represented in the reference matrix. For an 8-bit cover image, the size of the reference matrix is 256×256.

#### 6.3.1.1 Chang et al.’s Method [79]

**Step 1:** Create a tile matrix (T) by using Sudoku puzzle solution and subtracting 1 from all the fields in the puzzle as shown in Figure 6.11. The initial puzzle contained data values ranging from 1 to 9. After subtracting 1 from each of the values in the solution of the puzzle, we obtain the solution that contains values ranging between 0-8.
Figure 6.11: (a) An instance of 9×9 Sudoku problem. (b) A solution of the Sudoku instance shown in Figure 6.11(a), where a digit / symbol occurs exactly once in each row, column, and minigrid. (c) The tile matrix (T) for creating the reference matrix (M) by subtracting 1 from each of the values in each cell of the solved Sudoku grid obtained in Figure 6.11(b).

Figure 6.12: Pixel tuple from carrier image.

Step 2: Generation of the reference matrix (R): Replicate the tile matrix on both axes to create a matrix of size 256×256. The reference matrix, M is then consisting of an $m\times m$ tiling of copies of T, where $m = \lceil 256/9 \rceil + 1$. The overflowing fields are ignored and M is truncated to 256×256 matrix.
Step 3: Data embedding: Convert the cipher text to base-9 numeral system. So that the cipher text is converted as: $S = S_1 S_2 S_3 \ldots S_n$, where $n$ is the number of converted secret digits and $S_k \in [0, 8]$, $1 \leq k \leq n$. The converted cipher text can now be pointed into the reference matrix as the values in this matrix also lie in the same range (i.e., 0-8).

Step 4: The cover image is partitioned into $R$ nonoverlapping blocks of size $1 \times 2$ as shown in Figure 6.12. Each tuple contains one pair of values, in which each value ranges from 0 to 255. This value pair is treated as X and Y coordinates for the reference matrix. This is used to traverse the reference matrix for data embedding.

Step 5: Find the value in the reference matrix at that position pointed by the pixel tuple values using pixel pair $(X, Y)$ as x- and y-coordinates in $256 \times 256$ reference matrix. To conceal in the cover image the base-9 value of the cipher text (that we like to embed) is selected and then it is looked up at the closest position of $(X, Y)$ pointed in the reference matrix containing the said base-9 value; the coordinate $(X', Y')$ of the cell where it is found is selected.

Step 6: This new coordinate $(X', Y')$ is then updated in place of the original tuple value $(X, Y)$. This provides the location of the concealed data in the reference matrix with a deviation of maximum two bits in each value of the tuple. To obtain the data back, the receiver needs to generate the reference matrix again and selects the pixel pair sequentially to map the values in it.

But there are several limitations or disadvantages of this method.

(i) The works are performed only on greyscale images, and

(ii) With the increase in pixel size, the size of the reference matrix multiplies. Hence, it is infeasible for RGB images.

This method is further modified by Shetty et al. [78]. They have used a $27 \times 27$ reference matrix for the same purpose. The later method is slightly advantageous than the scheme proposed by Chang et al. as the size of the matrix is smaller. But unfortunately none of the methods is capable to detect if the cover image has been modified. If the cover image is changed, the message embedded inside the image will also be changed. They have used Sudoku reference matrices of size $256 \times 256$ (by Chang et al.) and $27 \times 27$ (by Shetty et al.), which are higher in dimension and also computation time intensive. In this respect, our devised technique uses only $18 \times 18$ sized
reference matrix that require much less time to perform the necessary computations. Besides, our method has also introduced an added layer of security by embedding an 8×8 solved Sudoku puzzle into the cover image to prevent any sorts of modification during transmission of the image. These have been elucidated in the following section.

6.3.2 A New Steganographic Scheme using Sudoku

As discussed earlier, the detection and prevention of modification in cover image is very much essential. The size of the reference matrix, used as a key for the hidden message, is also of higher dimension in all other earlier methods. In our proposed approach, we have used one 8×8 Sudoku for detecting modifications in the cover image and an 18×18 Sudoku reference matrix, as key, which are lesser in size than that of the previous methods discussed. The present finding has been published in [18].

The image used in our proposed method is a 24-bit coloured image. Initially, the cover image is assumed whereon an 8×8 solved Sudoku is embedded using the LSB embedding technique [64].

![Figure 6.13: (a) An instance of 8×8 Sudoku puzzle. (b) A solution of the Sudoku instance.](image)

6.3.2.1 Embedding of an 8×8 Sudoku Matrix in a Cover Image

Initially, the cover image is divided into 64 blocks. In each block, we have created several groups of three adjacent pixels each (in some fashion).

Then, we extract the values of B (Blue) components of each pixel, which is represented by eight bits.
Now we take an 8×8 solved Sudoku puzzle, wherein we subtract 1 from the value present in each cell. Then the value in each cell is ranged in 0-7, each of which can be represented by three bits. An instance of 8×8 Sudoku puzzle and its possible solution are shown in Figure 6.13.

![Sudoku Puzzle Image](image)

**Figure 6.14:** Embedding of the 8×8 Sudoku solution into the cover image.

![Embedding Image](image)

**Figure 6.15:** An 18×18 reference matrix (M) is created using tile matrix (T), introducing the solved Sudoku puzzle shown in Figure 6.11(c).

Now, in each block we embed the value present in each cell of the solved Sudoku puzzle. The embedding of these values is done in the following manner.
For each three pixels’ group of a block, we insert the values in the LSB of B (Blue) components; thus, only one bit is changed (at most). The embedding procedure of the given puzzle is shown in Figure 6.14. The sender is supposed to send the encrypted 8×8 Sudoku instance whose solution is embedded in the cover image; the receiver needs to solve the instance to recover the cover image. The benefit of using the 8×8 Sudoku is that if by any means the secret message or the cover image is tampered, then the 8×8 Sudoku will not be solved at the receiver’s side, and hence, the attack is detected.

After that, an 18×18 reference matrix is generated from a 9×9 solved Sudoku puzzle. The reference matrix is created by replicating 9×9 Sudoku solution in a (2×2) square form. The values of the reference matrix lie between 0 through 8 as 1 is subtracted from each cell value, and the reference matrix behaves as a reference look-up table for embedding the secret message. A sample 18×18 Sudoku reference matrix is generated, as shown in Figure 6.15, using the Sudoku instance shown in Figure 6.11.

6.3.2.2 Embedding of Hidden Message

It has already been stated in the previous section that the cover image is divided into 64 equal sized blocks and in each block three adjacent pixels are assumed in a group, where the number of such groups depends upon the size as well as the resolution of the image. Each group carries a character of the secret message and a value of the 8×8 (solved) Sudoku puzzle, which is determined by the corresponding block. The R (Red) and G (Green) components hold the secret character whereas the B (Blue) component holds the value of the Sudoku. Now we discuss in successive steps below how the hidden message could be embedded in a cover message.

(i) Conversion of secret message into base-9

All the characters of the secret message are converted into base-9 having three digits each so that the digits of the characters fall in the range of 0-8 and can be referred to the 18×18 reference matrix. The necessary steps of the algorithm are as follows:

Read character one-by-one from the message.

Convert the character to ASCII code (decimal).

Convert the decimal system (in ASCII code) to base-9.
Add padding bits (0’s) in front of the number in base-9, if necessary, so that each of the numbers now becomes of three-digit length.

(ii) Finding out the base-9 values in reference matrix

For each digit of the three-digit base-9 character code

Read pixel (h, f) /*h and f are random location of pixel*/

\[ X = (R \mod 6) + 6 \]
\[ Y = (G \mod 6) + 6. \]

Locate the cell \((X, Y)\) in the reference matrix. \(X\) can be considered as row number and \(Y\) as column number.

Now consider nine cells in the same row, keeping the cell \((X,Y)\) in the middle of it and store them in \(C_r\), i.e., we like to consider only four left-most and four right-most cells surrounding \((X,Y)\), or keeping the cell location \((X, Y)\) at the middle as shown in Figure 6.15, for example.

Similarly, select nine cells in the same column, keeping the cell \((X,Y)\) in the middle of them, and store them in \(C_c\). Select the minigrid, where the cell \((X,Y)\) belongs to and store them in \(C_m\).

If cell \((X_i, Y_i)\) contains a base-9 digit, then

Locate \((X_r, Y_r), (X_c, Y_c), (X_m, Y_m)\) in \(C_r, C_c, \) and \(C_m\) of the reference matrix, respectively. In location \((X_r, Y_r), (X_c, Y_c),\) and \((X_m, Y_m)\) of the reference matrix, the cell contains the same value of the base-9 digit.

Calculate deviation of each of \((X_r, Y_r), (X_c, Y_c), (X_m, Y_m)\) from \((X, Y)\), separately.

Select the cell from the above three candidate elements which has minimum deviation from \((X, Y)\).

(iii) Updating the R and G values of each pixel

Suppose, \((X_c, Y_c)\) has minimum deviation from \((X, Y)\), then

The pixel \((h, f)\) has now the newly embedded data with R-G-B values as:

\[ R = R - (X - X_c), \]
\[ G = G - (Y - Y_c), \]
where B holds the 8×8 Sudoku value embedded previously.

Let us take an example; say after conversion from R-G values, using formula (in Section 6.3.2.2(ii)), say for a particular pixel we get X = 12 and Y = 7. The corresponding value in position (12, 7) is 7 (as shown in Figure 6.15), marked by a circle. Say we want to encrypt a character ‘A’. The ASCII equivalent of ‘A’ is 65. If we convert it into base-9, it becomes 027. So, 0 is required to be embedded in the first pixel, 2 in the second pixel, and 7 in the third pixel amongst the three-pixel group. Let us try to embed 0 in the first pixel. At first, we have to found out the elements of C\_r, C\_c, and C\_m. The elements of C\_r, C\_c, and C\_m are marked (or encircled) with dotted coloured lines green, red, and blue, respectively, as shown in Figure 6.15. Now we can clearly mark the presence of 0 in C\_r, C\_c, and C\_m. We find the least deviation in C\_r, which is also true for C\_m that has been computed later on. Then we can update the R and G values accordingly. We can observe that the value of (X-X\_c) or (Y-Y\_c) is at most four. So, there is a very negligible amount of distortion in the cover image.

### 6.3.2.3 Checking the Integrity of the Cover Image

For checking whether the cover image has been modified or not, we first need to send an 8×8 Sudoku instance to the receiver. Receiver solves this puzzle and stores it. Then the receiver divides the cover image into 64 blocks and in each of the blocks, a group of three pixels be prepared, as mentioned above. Next, the last bit of each three-pixel group is taken and the equivalent decimal values are kept in the 8×8 matrix. If this matrix matches with the already solved Sudoku puzzle, then the integrity is maintained; otherwise, we may conclude that the image has been modified.

### 6.3.2.4 Hidden Message Extraction

For extracting the hidden message from the stego image, we first need to transfer the instance of 9×9 Sudoku puzzle. Then after receiving it the receiver solves this puzzle and an 18×18 reference matrix is generated.

Then, for each bit in a three-pixel group, we compute the following:

\[
X = (R \mod 6) + 6 \quad \text{and} \quad Y = (R \mod 6) + 6.
\]
Figure 6.16: The proposed Steganographic scheme.
Find the value present at (X, Y) from the reference matrix.

End for

Concatenate the three values from the three pixels.

Convert the result to base-10. This is an ASCII value of the hidden character message.

Get the character equivalent to the decrypted ASCII code.

The sender of the secret message needs to send an 8×8 and a 9×9 Sudoku instance to the receiver. Sender encrypts the instances and sends them to the receiver.

The whole process (i.e., the proposed Steganographic scheme) is shown in Figure 6.16.

**Figure 6.17:** A sample cover image.

**Figure 6.18:** Stego image after embedding the sample cover image shown in Figure 6.15, in an 8×8 Sudoku and secret message.
6.3.2.5 Experimental Results and Analysis

The quality of stego image is evaluated using histogram comparison in MAT Lab and embedding capacity in terms of characters. A sample result has been shown as stated below.

After performing histogram analysis, as shown in Figures 6.19 and 6.20, we may observe that there is significantly less distortion in the cover image.

The embedding capacity for the image shown in Figure 6.17 is calculated in the following manner:

Sample image size = 229 KB.

Number of pixels in the image = 78089.

Number of pixels required for hiding one character = 3.

Therefore, 78089/3=26029 characters.

Thus, this method can hide approximately 26 KB secret messages in the image shown in Figure 6.18.

We have also performed steganalysis of the stego image; we can find that the devised scheme in this chapter is safe against geometrical attack, noise based attack, and so on.
A Brief Comparison with Existing Method

The Chang et al.‘s method [79] has used a reference matrix of size 256×256, whereas in the Shetty et al.’s method [78], a 27×27 reference matrix has been utilized. In this respect the reference matrix that we have introduced is of size 18×18, which is even smaller in size. Use of smaller reference matrix also reduces the required computation time. Moreover, we have embedded a separate 8×8 Sudoku matrix into the blue components of each pixel in each block, which provides an additional layer of security.

We have proposed a Steganographic scheme, where we have used an 18×18 Sudoku reference matrix for embedding the message and we also have embedded an 8×8 Sudoku, for checking whether the cover image has been modified. If somehow the cover image gets modified, it can by far be detected as we have already embedded the 8×8 Sudoku matrix inside it (i.e., the cover image). It can also prevent any modification, as each Sudoku puzzle matrix should have values 0 through 8 only once in the same row, in the same column, and in a minigrid. An 18×18 Sudoku reference matrix is used for hiding the secret message, whereas the entire earlier existing methods used 256×256 or 27×27 reference matrix. That is why, less computation is involved in our method. So, it can eventually be claimed that our developed scheme is more robust where the amount of necessary computation is also reduced.

6.4 Summary

In this chapter we have discussed about some important domains of application of solving Sudoku puzzles. In the present investigation we have identified two new domains, namely Biometric and Steganography, where we can apply our devised technique much efficiently. The advantage of using Sudoku in these domains is that nobody will be capable in changing or modifying the content except the intended users.

We know that a Sudoku instance might have two or more solutions as well, and such a situation has its own merits and applications, especially in the domain of Steganography. If a given Sudoku instance is having only one valid solution, then that instance may fail to conceal some information while transmitting the instance that an intruder may attack. On the other hand, if there are two or more solutions for a given Sudoku puzzle, then such attackers may confuse in extracting the hidden information.