The Journal of Systems and Software 83 (2010) 2528–2535

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A perfect maze based steganographic method

Hui-Lung Lee, Chia-Feng Lee, Ling-Hwei Chen*

Department of Computer Science, National Chiao Tung University Hsinchu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 8 January 2010
Received in revised form 7 June 2010
Accepted 24 July 2010
Available online 3 August 2010

Keywords:
Steganography in games
Information hiding
Steganalysis
Perfect maze
Solution path

ABSTRACT

In steganography, several different types of media have been used as carriers, such as images, audios and video streams, to hide secret data. Nevertheless, various novel media and applications have been developed due to the rapid growth of internet. In this paper, we select maze games as carrier media to conceal secret data. The original idea of embedding data in a maze is proposed by Niwayama et al. Their method has two disadvantages. One is the very small embedding capacity; the other is that the stego maze is not perfect. Here, we propose an improved algorithm for increasing the embedding capacity and preserving the "perfect" property.

1. Introduction

Steganography (Cox et al., 2008) is a method to conceal and preserve the secrecy of information in a carrier. Traditionally, images, audios and video streams have been used as carriers due to their popularity on the internet. Nevertheless, various novel media and applications have been developed due to the rapid growth of internet. For example, users can play online games with friends, send puzzles to friends' mobile devices (Shirali-Shahreza and Shirali-Shahreza, 2008), or download printable puzzles for kid education (Fun Printable Mazes for Kids, 2009). Hence, some methods (Shirali-Shahreza and Shirali-Shahreza, 2008; Hernandez-Castro et al., 2006; Niwayama et al., 2002; Diehl, 2008; Desoky and Younis, 2009; Kieu et al., 2009; Farn and Chen, 2009a,b; Zander, 2008) have been proposed to conceal message in these novel media. A maze is an excellent educational game (Fun Printable Mazes for Kids, 2009), so many maze games could be found on the internet (Fun Printable Mazes for Kids, 2009; Novel Games, 2009; WORLDVILLAGE KIDZ, 2009; Lost in the maze, 2009). Besides, the maze is a novel carrier in steganography (Niwayama et al., 2002). Discovering new and better carrier can enhance the secrecy of steganography. In this paper, we will propose a steganographic method using a perfect maze to hide secret data. In the following, we will give a brief description for a perfect maze.

A maze (see Fig. 1) basically contains cells, walls, a starting cell, and an end cell. Logically, a maze is a puzzle with complex multipath network, and a player is to find a solution path from the starting cell to the end cell. A rectangular maze has m cells in width and n cells in height and is denoted as $m \times n$ maze, it is called perfect if there exists one and only one path between any two cells (Pullen, 2009). Figs. 2 and 3 (Kirkland, 2009) show a perfect maze and an imperfect maze, respectively. From our observation, most puzzles appearing in websites (Fun Printable Mazes for Kids, 2009; Novel Games, 2009; WORLDVILLAGE KIDZ, 2009; Lost in the maze, 2009) are rectangle and perfect. For security consideration, the created maze should look like a common one (Pullen, 2009). Hence, here we only deal with rectangular perfect mazes.

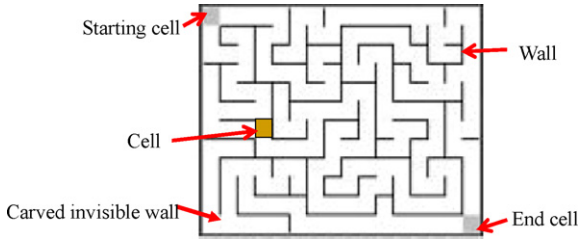

Fig. 1. An example to illustrate a maze structure.

Regarding cells as nodes, carved invisible walls as links, we can express a maze as a graph. Fig. 4 shows an example, a number attached to a link connecting two nodes N and M stands for the number of intermediate nodes appearing in the shortest path from N to M. Based on this representation, we can find that a perfect maze corresponds to a tree (see Fig. 5), such relation can be used to prove whether a generated maze is perfect.

The rest of the paper is organized as follow. Section 2 outlines the previous works. Section 3 describes the proposed algorithm

* Corresponding author. Tel.: +886 3 5712121x54744; fax: +886 3 5721490.
E-mail addresses: huilung@debut.cis.nctu.edu.tw (H.-L. Lee), encounter@debut.cis.nctu.edu.tw (C.-F. Lee), lhchen@cc.nctu.edu.tw (L.-H. Chen).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.07.054

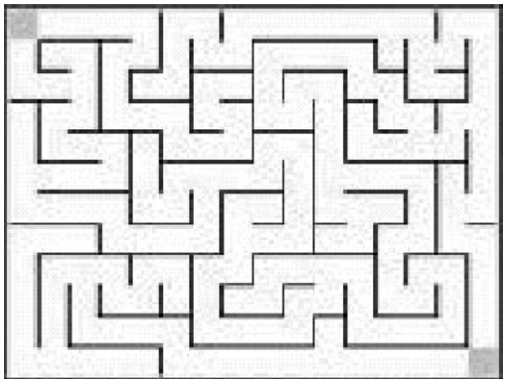

Fig. 2. A 16 × 12 perfect maze.

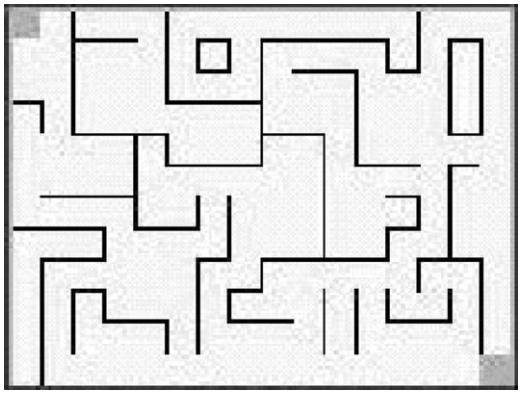

Fig. 3. A 16 × 12 imperfect maze.

and Section 4 is the proof and security analysis for the proposed method. Section 5 illustrates the experimental results, and Section 6 draws conclusions.

2. Previous works

In this section, we will first introduce a typical maze generator, then a steganographic method based on this generator is described.

2.1. Hunt-and-Kill maze generating (HKMG) algorithm

There are a number of maze generating algorithms (Pullen, 2009), Hunt-and-Kill maze generating algorithm (Niwayama et al., 2002) is typical. The HKMG algorithm generates a maze by carving walls. Fig. 6 shows a maze generated by the HKMG algorithm. In

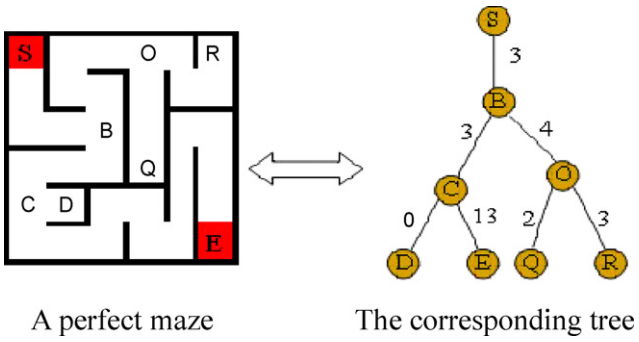

Fig. 5. Correspondence between a perfect maze and a tree.

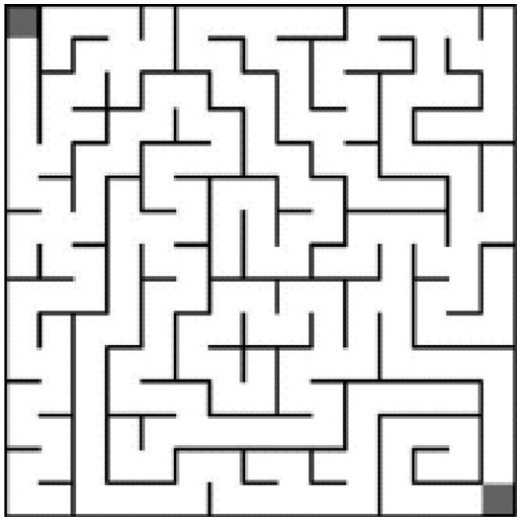

Fig. 6. An maze generated by HKMG algorithm.

HKMG algorithm, there are three types of cells defined as follows:

- 'In' cell (I): a cell that has been processed and always keeps its type.
- 'Frontier' cell (F): a cell that is processed and is a 4-neighbor of a certain "I" cell.
- 'Out' cell (O): a cell not yet processed.

In the following, we will give a brief description for the HKMG algorithm.

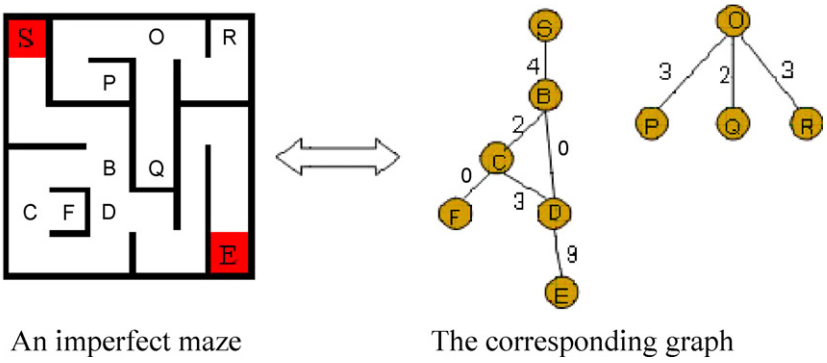

Fig. 4. Correspondence between an imperfect maze and a graph.

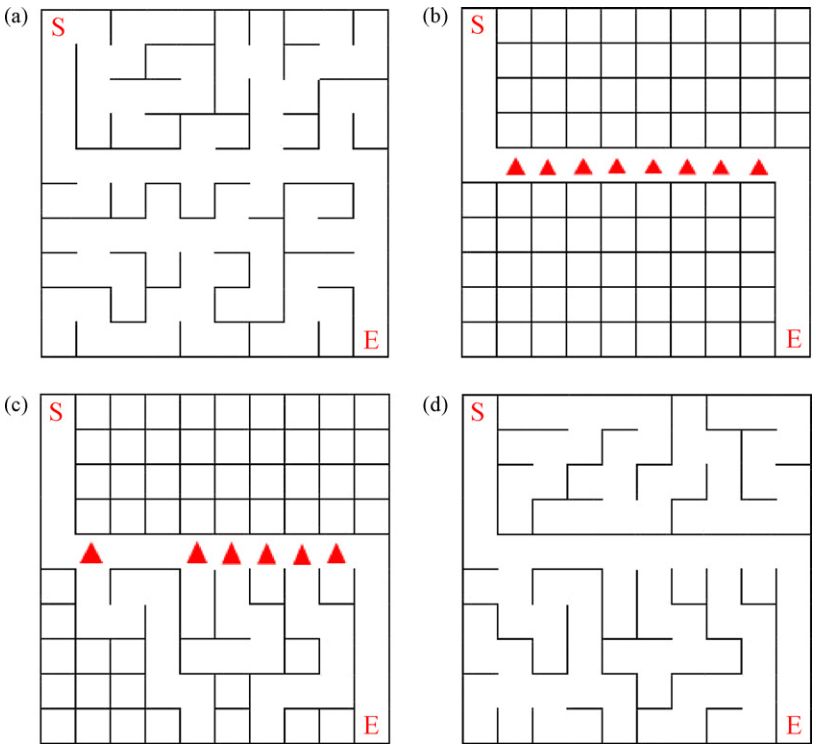

Fig. 7. An imperfect maze generated by the HK embedding algorithm. (a) A perfect maze generated. (b) The solution path remained with eight embeddable cell marked by "▲". (c) Secret data '111111' embedded with six embeddable cells marked by "▲". (d) A maze with an inaccessible area generated based on (c).

1. Mark all cells as O cells.
2. Mark the starting cell as I cell, and mark each O cell in the 4-neighborhood of the I cell as F cell.
3. Choose an F cell around an I cell, and carve the wall between the F cell and the I cell. Mark the F cell as I cell, and mark each O cell in the 4-neighborhood of the I cell as F cell. Repeat step 3 until there is no F cell.
4. End.

As mentioned previously, a maze can be represented as a graph. Therefore, we can show that the HKMG algorithm generates a perfect maze only by only showing that the corresponding graph is a tree. The reason is that any two nodes in a tree have one and only one path. This matches the property of a perfect maze. In the following, we will give a brief proof.

Theorem 1. *A maze generated by the HKMG algorithm is a perfect maze.*

Proof. At first, let set A contain only the starting cell. Create a graph G with the starting cell to be its root node. In step 3 of the HKMG algorithm, when an F cell is chosen, a new node corresponding to the F cell is created and added in G. When a wall between the F cell and its neighbor I cell is carved, a link between the corresponding two nodes (one for I cell in set A, the other for F cell outside A) is added in G, and the F cell is added in set A. This will guarantee that no loop occurs in G. Thus, at the end of the HKMG algorithm, each cell has a corresponding node added to G and connected to a certain node. This means that the corresponding graph G is a tree.

The HKMG algorithm is nearly bias free (Lost in the maze, 2009). A maze generation algorithm is called bias free, if it treats all directions and sides of the maze equally. Since HKMG algorithm is nearly bias free, we could say it generates almost all perfect mazes with the same probability. Besides, the HKMG algorithm does not require extra memory (Lost in the maze, 2009). Hence, the HKMG algorithm can be used to create mazes of larger size or executed on systems with limited processing power.

2.2. Hunt-and-Kill embedding algorithm

Niwayama et al. (2002) provided a data hiding method called HK embedding algorithm, which embeds secret data in a maze generated by the HKMG algorithm. The HK embedding algorithm is described as follows:

1. Locate a solution path from the starting cell to the end cell.
2. Make branches in the solution path depending on embedded data. Concretely, if the embedded data bit is 1 then make a branch on the right, else if the data bit is 0 then on the left.
3. Applying HKMG algorithm to complete the maze creation.

Unfortunately, the HK embedding algorithm may generate some inaccessible sections (Niwayama et al., 2002). Fig. 7 shows an example. A perfect maze (Fig. 7(a)) is first generated by HKMG algorithm. Secondly, all cells are cleared besides the solution path from the starting cell S to the end cell E (see Fig. 7(b)). If the embedded data are '111111', then the solution path makes six right branches (see Fig. 7(c)). After applying HKMG algorithm to Fig. 7(c) to complete the maze creation, an inaccessible section on the upper part of the maze (see Fig. 7(d)) is generated. Each cell in this section does not connect to S and E. Thus, the maze generated by this algorithm is not a perfect maze. This might attract the attention of inspectors. According to Cox et al. (2008), the property of undetectability is the main requirement of steganography. Furthermore, the HK embedding algorithm provides very small embedding capacity. To overcome these problems, we will propose an improved embedding method to generate a perfect maze, and the proposed method will provide more embedding capacity by using multiple paths instead of only one path.

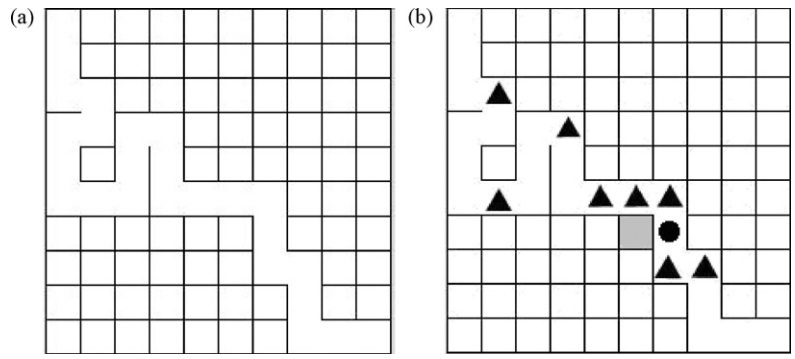

Fig. 8. One example to illustrate embeddable cells. (a) The white path stands for the solution path. (b) Embeddable cells marked by black triangles.

3. Proposed method

The main idea of the proposed method is to consider multipaths rather than only the solution path to gain more embedding capacity. Before describing the proposed method, we will define "embeddable cell" which will be used to embed a bit. Suppose the HKMG algorithm is used to generate a perfect maze, and the solution path from the starting cell to the end cell is located. All cells on the path are marked as I cells, the other cells are reset to be O cells, and all walls are rebuilt, except those in the path (see Fig. 8(a)).

Definition 1. An embeddable cell, A, is an I cell, which is in the solution path with exact two O cells in its 4-neighbors, and each O cell should not be a neighbor of another embeddable cell which appears before A.

Fig. 8 shows an example. In Fig. 8(b), the cells with black triangles are embeddable ones and the gray cell is an O cell and is the overlapped neighbor of two I cells, thus the cell with a black solid circle is not an embeddable cell. Based on the definition, all embeddable cells can be located. According to the embedding bit, we carve the wall between an embeddable cell and one O cell around it, and mark the O cell as I cell. There are six kinds of embeddable cells shown in Fig. 9. Each embeddable cell has two neighboring O cells marked as 1 and 0. If a "1" bit is embedded, then the wall between the embeddable cell and the cell marked "1" is carved, and the "1" cell is marked as I cell. Otherwise the wall between the embeddable cell and the cell marked as "0" is carved, and the "0" cell is marked as I cell.

To increase embedding capacity, we can embed bits into multipaths instead of only one path. In the proposed method, we first generate a perfect maze with HKMG algorithm. Subsequently, we choose some cells as the start cells and one cell as the common end one of the multipaths. Finally, we solve the perfect maze to obtain the corresponding multipaths. These multipaths sometimes will merge at some cells. Fig. 10(a) shows a 10×10 perfect maze generated by the HKMG algorithm. Fig. 10(b) shows two solution paths. The two solution paths start at S and T, respectively, they have the common end cell E. The second path from T to E is merged into the first path from S to E at cell A. Note that the number of solution paths and all the start cells and one end cell are chosen through a random number generator and a seed, which will be considered as the secret key. This part will be further addressed in Section 3.3.

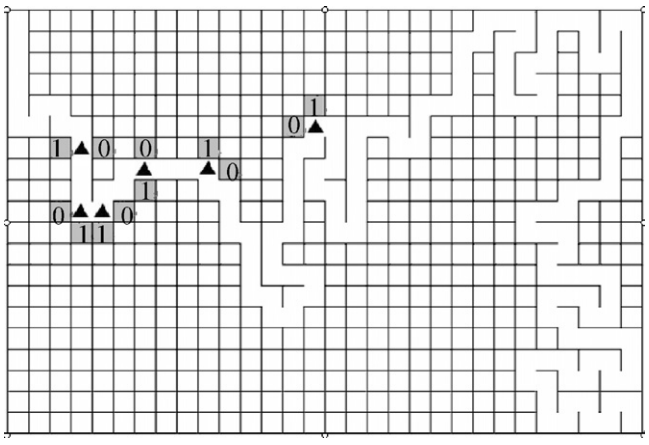

Fig. 9. Six kinds of embeddable cells.

3.1. Embedding algorithm

After obtaining the multiple paths, we order these paths according to their starting cells from top to bottom and then left to right. According to the path sequence, we trace each path from the start cell to locate all embeddable cells. Note that if a subpath of a certain path has already been traced, it will be skipped. As a result, we obtain a sequence of embeddable cells. Then data can be embedded according to this sequence. Here, we will define a new type of cell, 'D' cell, which will be used in the proposed embedding algorithm.

Definition 2. Let A be an embeddable cell, B and C be its two neighboring O cells. If the wall between A and B is carved to embed one bit, cell C is called a D cell.

The details of the proposed embedding algorithm are described as follows:

1. Create a maze using the HKMG algorithm (see Fig. 10(a)).
2. Choose some cells as start cells and one cell as the end cell. Solve multipaths from these starting cells to the end cell (see Fig. 10(b), S and T stand for two starting cells, E stands for the end cell, and A stands for the merging cell).
3. Reset all cells to be O cells and all walls as visible. Set those cells on multipaths to be I cells. Carve each wall between two I cells (see (c)).
4. Find all embeddable cells in each path and order them according to the path sequence. Note that we do not set the cell on boundary to be embeddable cell even if the cell is embeddable (see Fig. 10(d), black triangles stand for embeddable cells).
5. For each embeddable cell, if the embedding bit is 1 (0), the wall between the embeddable cell and its neighboring O cell marked 1 (0) (see Fig. 9) is carved and the cell marked 1 (0) is set as I cell, the other neighboring O cell marked 0 (1) is set as D cell (see Fig. 10(e), D stands for the D cell).
6. Set those O cells around I cells to be F cells.

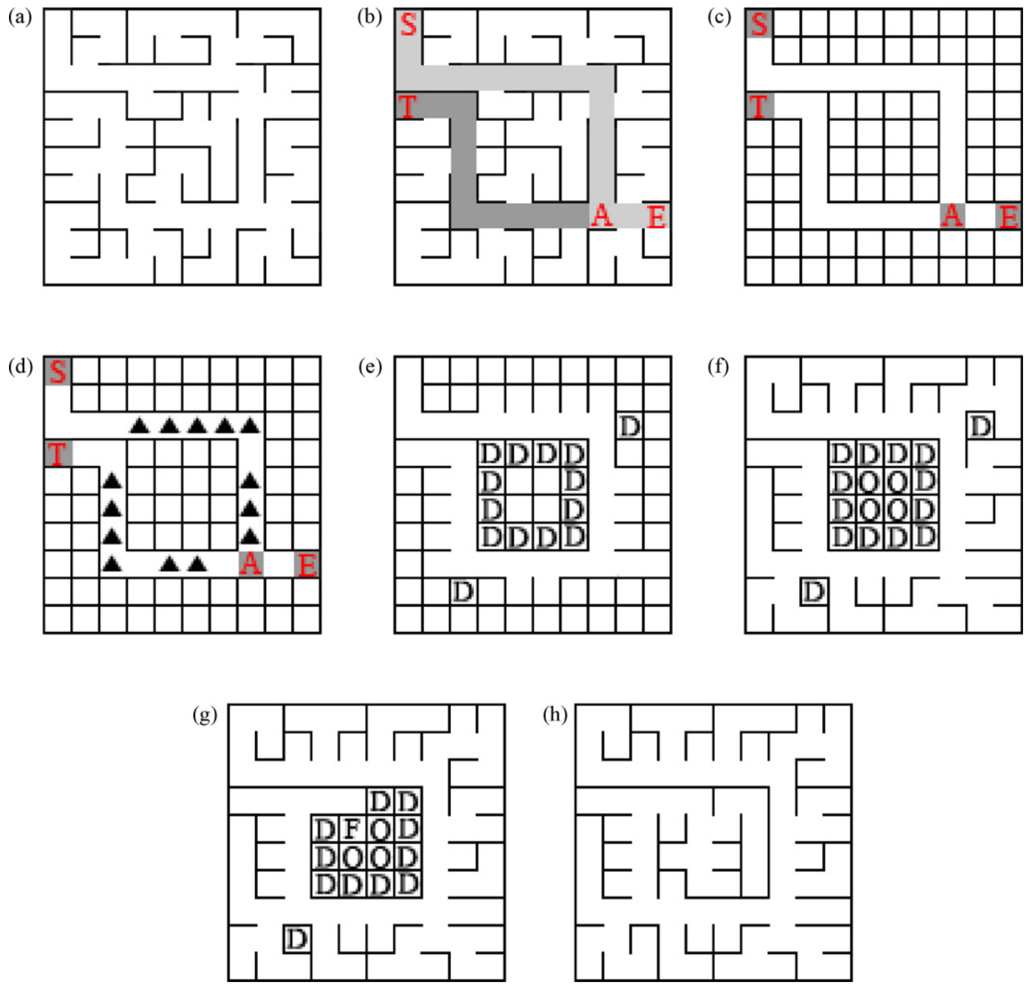

Fig. 10. An example of using our proposed method to embed data in a perfect maze. (a) A perfect maze generated by HKMG algorithm. (b) Two paths from S to E and T to E in (a) located with merged points A. (c) The result after performing Step 3 of the proposed embedding algorithm to (b). (d) All embeddable cells located. (e) The result after embedding data in embeddable cells with D cells marked. (f) The result of applying HKMG algorithm to process F cells. (g) The immediate result of applying Step 8 of the proposed embedding algorithm to (f). (h) The perfect maze generated after processing all Ds.

7. Process these F cells using HKMG algorithm (see Fig. 10(f), O stands for the O cell).
8. Process the D cells.
   (a) Scan the maze, and check if any D cell exists. If none, go to step 9. Otherwise, choose a D cell with one of its neighbors being an un-embeddable I cell and carve the wall between the D cell and the un-embeddable I cell. Mark the D cell as I cell, and mark each O cell in the 4-neighborhood of the I cell as F cell (see Fig. 10(g)).
   (b) Check if any F cell exists. If yes, go to step 7. If none, go to step 8.
9. End (see Fig. 10(h)).

3.2. *Extracting algorithm*

To extract the embedded bits, the receiver must know the number of solution paths and locate the start and end cells of multipaths, which can be extracted through the secret key and a pseudo random number generator. This part can be found in Section 3.3. Since the generated maze is perfect, the receiver can get the original multipaths accurately according to these start and end cells. Then, the receiver can locate all embeddable cells. Finally according to the direction of the branch in each embeddable cell, secret data can be extracted successfully.

3.3. *Maze information generation*

In the proposed method, when the sender wants to embed secret messages in mazes, he should determine a set of maze information including the maze size ($m \times m$), the number of solution paths (k), six types of embeddable cells, the positions of start cells (s_0, s_1, ..., s_k), and the position of end cell (e). In the paper, we provide a method to produce the set of maze information automatically. First, the sender inputs a seed to a pseudo random generator to generate the set of maze information: the maze size ($m \times m$), the number of solution paths (k), six types of embeddable cells, the positions of start cells (s_0, s_1, ..., s_k), and the position of end cell (e), sequentially. Then, the sender uses the set of maze information to generate a perfect maze with secret message embedded. If the capacity of the generated maze is less than the size of the secret message, the sender will continue to generate more perfect mazes in the same way until the whole secret message is embedded. The procedure is repeated until all secret messages are embedded. Note that the secret messages will be padded to the capacity of the generated mazes. Finally, the sender considers the seed used to embed secret messages as a secret key and share it with the receiver in a secure way.

When the receiver receives mazes, the receiver can use the secret key to produce the maze information, and based on the maze information to extract secrets.

Table 1
The embedding capacity of the proposed method.

Number of solution path	HK embedding algorithm (1 path)	Proposed method (1 path)	Proposed method (2 paths)	Proposed method (3 paths)	Proposed method (4 paths)	Proposed method (5 paths)
Average capacity (bits)	64	125	152	213	216	224

4. Proof and security analysis for the proposed method

In this section, we first prove the maze generated by the proposed method is a perfect maze, then the security analysis for the proposed method is described.

4.1. The proof for generating a perfect maze

As mentioned previously, since the maze generated by HKMG algorithm is a perfect maze, we can build a corresponding tree. In the proposed method, we first get the multipaths from some start cells to one end cell. Thus, the corresponding subgraph, T, of these multipaths can be obtained by taking off all nodes and edges, which are not in these multipaths. The subgraph, T, is still a tree, since all cells in the multipaths are connected through the end cell. When we embed one bit into one embeddable cell, A, on the multipaths, only one cell neighboring to A and not in T is added to T, and the link connecting the cell and A are added to T. Thus, the expanded graph T after embedding all bits is still a tree. Subsequently, we use the HKMG algorithm to process F cells. Since the HKMG algorithm is a tree based algorithm, this will make the expanded graph T still be a tree after all F cells are processed. At last, when a D cell with one un-embeddable I neighboring cell is added, the expanded graph T is still a tree. Thus, to prove that the generated maze is perfect, we only need to prove that all Ds are marked as I cells after the proposed algorithm ends. To prove this point, six lemmas are first given; their proof can be found in Appendix A. Here, we briefly describe the idea of proof. For a D cell, its four 4-neighbors have four situations. The first is that each of its four 4-neighbors is not a D cell, the second is that exactly one of its four 4-neighbors is a D cell, the third is that exactly two of its four 4-neighbors are D cells, the fourth is that exactly three of its four 4-neighbors are D cells. Lemmas 3–6 are provided to prove that for each situation, the D cell should become an I cell after applying the proposed embedding algorithm. Lemmas 1 and 2 are provided to aid the proof of Lemmas 3–6.

Based on Lemmas 3–6, we can get the following theorem immediately.

Theorem 2. *In the proposed method, any D cell will become an I cell when the method ends.*

4.2. Security analysis for the proposed method

In steganography, there are two stages to break a steganographic system (Zollner et al., 1998; Katzenbeisser and Petitcolas, 2000): First, an attacker can detect the existence of a secret message in a steganographic system. Second, the attacker can extract the embedded message. Since each maze produced by the HKMG algorithm is perfect, to make a HKMG based embedding method undetectable, the perfect property should be kept. We have proved that all mazes generated by the proposed method are perfect. This makes an attacker not able to distinguish the mazes produced by our proposed method from those produced by the HKMG algorithm. However, the HK algorithm cannot guarantee that each generated maze is perfect, this makes the HK algorithm detectable. On the other hand, if an attacker wants to extract the embedded bits by "brute force", the attacker must guess correct start and end cells and embedded type. Suppose that K is the maximum number of solution paths. The attacker should guess the correct six kinds of embeddable cells (total number of choices is 2^6), the possible positions of start (total number of possible positions is $\binom{n^2}{i}$) and end cells (total number of possible positions is $(n^2 - i)$), and the order of solution paths (total number of possible orders is $i!$ for i solution paths). Hence, the probability of the attack is $\frac{1}{2^6 \times \sum_{2 \le i \le K} (n^2 - i) \times \binom{n^2}{i} \times i!} \approx \frac{1}{n^{2(K+1)}}$. Note that when $n \ge 10$, $K \ge 4$, the probability of the attack is less than 10^{-10}, this is very small. Thus, the proposed method is secure.

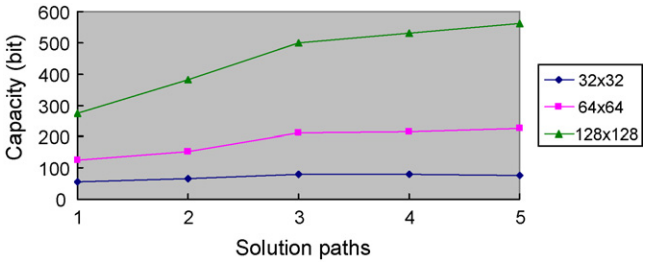

Fig. 11. The capacities of different sizes of mazes.

5. Experimental results

Note that when the HK embedding algorithm embeds a bit into the solution path, a branch path will be generated and may adjoin the solution path (see Fig. 7(c)), this makes some embeddable cells in the solution path become un-embeddable (see Fig. 7(c), two embeddable cells become un-embeddable). In our proposed method, we mark all the embeddable cells and embed bits into these embeddable cells before applying the HKMG algorithm.

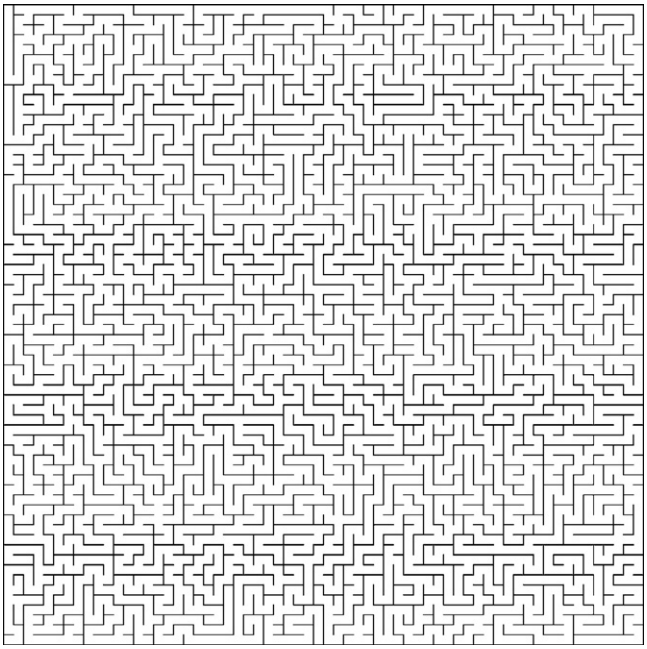

Fig. 12. A maze generated by the HKMG algorithm.

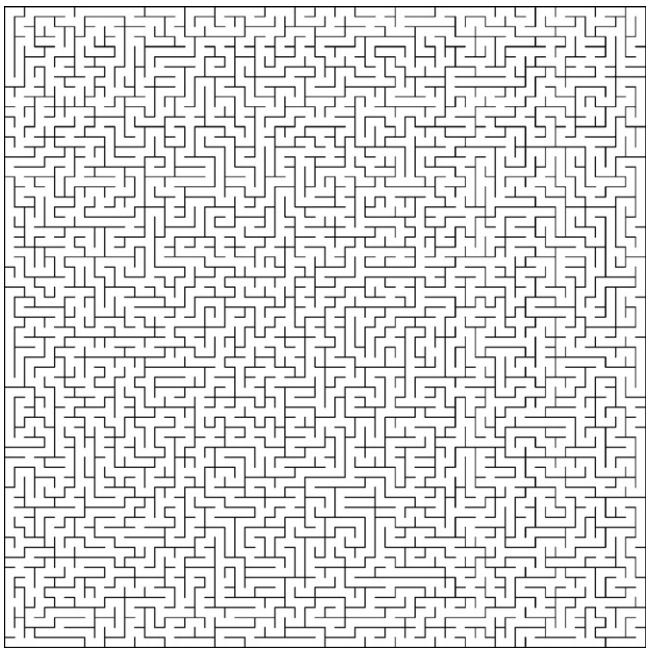

Fig. 13. A maze generated by the proposed method with 221 bits embedded.

Hence, no embeddable cells become un-embeddable. Thus, our proposed method has higher capacity. We also conduct some experiments to show this point. In Table 1, we generate twenty 64×64 mazes and calculate their capacities using different number of solution paths. In the HK embedding algorithm, only 64 bits can be embedded into a 64×64 maze (Niwayama et al., 2002). According to our experiment, the average capacity of the proposed algorithm using one solution path is nearly twice that of the HK embedding algorithm, and the capacity using three solution paths is three times. When embedding bits in more than three solution paths, the capacity of our proposed algorithm will not be increased substantially. Fig. 11 shows the capacity of 32×32, 64×64, and 128×128 mazes which embed bits into one to five solution paths, respectively. Fig. 12 shows a 64×64 maze generated by the HKMG algorithm. Fig. 13 shows a 64×64 maze generated by our proposed method using three paths, in which there is 221 bit embedded. Both figures show perfect mazes, and based on visual inspection we cannot identify the maze that embeds secret data. From our experiments, we can also see that our proposed algorithm actually provides higher embedding capacity.

6. Conclusions

We proposed a novel method to embed secret data into mazes (Fun Printable Mazes for Kids, 2009; Novel Games, 2009; WORLDVILLAGE KIDZ, 2009; Lost in the maze, 2009). It generates perfect mazes (Pullen, 2009) that cannot be distinguished visually by humans from other perfect mazes commonly used. Hence, it significantly improves security over the existing HK embedding algorithm (Niwayama et al., 2002) that does not generate perfect mazes. When encoding bits into one solution path our new method on average provides approximately twice the embedding capacity of the one-path HK algorithm. With larger mazes our algorithm can utilize multiple paths for encoding to further increase capacity, while maintaining its "undetectability" for human vision. In future work, we will focus on other puzzle games.

Acknowledgements

This work is supported in part by National Science Council and Taiwan Information Security Center at NCTU (NSC96-2219-E-009-013).

Appendix A.

Lemma 1. *For a D cell, exact one of its 4-neighbors is an embeddable I cell.*

Proof. Suppose A is a D cell, then A is an O cell. By Definition 2, we know that there is one of A's 4-neighbors to be an embeddable I cell. By Definition 1, we know that any two embeddable cells should not have an O cell as their common 4-neighbors. Thus, only one of A's 4-neighbors could be an embeddable I cell. The proof is completed.

Lemma 2. *For an embeddable I cell, after a bit is embedded, exact three 4-neighbors of the embeddable I cell are I cells.*

Proof. Let A be an embeddable I cell. According to Definition 1, we know that there are exact two 4-neighbors of A to be O cells, that is, there are exact two 4-neighbors of A to be I cells. After a bit is embedded, one of these two O cells will become an I cell. Thus, there are exact three 4-neighbors of A to be I cells.

Lemma 3. *Let A be a D cell, and each of its four 4-neighbors is not a D cell, then A should become an I cell after applying the proposed embedding algorithm.*

Proof. Since A is a D cell, by Lemma 1, we have exact one of its four 4-neighbors to be an embeddable cell. Without loss of generality, suppose that C (see Fig. 14) is an embeddable cell, by Lemma 2, we have exact three of C's four 4-neighbors are I cells. This implies that B is an I cell. Since J is not a D cell, J will become an I cell. Since J is not embeddable, this will make A become an I cell.

Lemma 4. *Let A be a D cell with exact one of its four 4-neighbors being a D cell, then A should become an I cell after applying the proposed embedding algorithm.*

Proof. Suppose that A has exact one of its four 4-neighbors being a D cell, and suppose that C is a D cell (see Fig. 14). Since A is a D cell, from Lemma 1, we have one of J, H, and F to be an embeddable I cell.

(1) If J (or F) is an embeddable I cell, then from Definition 1, we have that H is not embeddable, and from Lemma 2, we have that K (or G) is an I cell. Since H is not a D cell, H will become an I cell, this makes A become an I cell.
(2) If H is an embeddable I cell, then from Lemma 2, we have that K is an I cell, and from Definition 1, we have that J is not embed-

	B	C	E	
	J	A	F	
	K	H	G	

Fig. 14. An example to illustrate Lemmas 3 and 4.

dable. Since J is not a D cell, J will become an I cell, this will make A become an I cell.

Lemma 5. *Let A be a D cell, and exact two of its four 4-neighbors are D cells, then A will become an I cell after applying the proposed embedding algorithm.*

Proof. We prove this Lemma in two different cases (see Fig. 15).

Case 1. Suppose that C and J are D cells (for C and F, J and H, or F and H, the proof is similar), then one of H and F will be an embeddable I cell. If H is an embeddable I cell, then F is not embeddable and G is an I cell (by Lemma 2). This implies that F is an un-embeddable I cell. This make A become an I cell. If F is an embeddable I cell, the proof is similar.

Case 2. Suppose J and F (or C and H) are D cells.

(1) Consider F cell, if E, T and G are not D cells. By Lemma 4, we have F is an I cell, this will imply A to be an I cell.
(2) If one of E and G is a D cell, according to the proof of Case 1, we have F is an I cell, this make A be an I cell.
(3) If E and G (for E and T, G and T, the proof is similar) are D cells, by Lemma 6 (will be proved later), we have F is an I cell, this will imply A to be an I cell.
(4) If exact T is a D cell, we consider S, Z, and U. According to the number of D in S and U cells, we apply the similar proof of (1) to (3) to get T to be an I cell. Since T is an I cell, F will become an I cell, this will imply A to be an I cell. If the proof of (1) to (3) cannot be applied, this means that the horizontal cells such as that from F to M except W are D cells. Since we restrict all boundary points not to be embeddable cell, W is not embeddable, and W is not D due to that L and N are not embeddable cells. This means that M has exact one of its 4-neighbors being D cell. By Lemma 4, M will become an I cell, then we get the left cell of M be an I cell. In the similar fashion, we get all horizontal cells from M to F be I cells, then A will be an I cell.

Lemma 6. *Let A be a D cell, and exact three of its four 4-neighbors be D cells, then A will become I cells after applying the proposed algorithm.*

Proof. If J, H, and F are D cells (see Fig. 15), then C will be an embeddable I cell (by Lemma 1). According to Lemma 2, B, R, and E will be I cells.

(1) Consider F, if G and T are not D cells. By Lemma 4, we have F is an I cell, this will imply A to be an I cell.
(2) If one of G and T is a D cell, by Lemma 5, we have F is an I cell, this will imply A to be an I cell.

	…		R				…		
	…	B	C	E	S		…	Q	L
	…	J	A	F	T	Z	…	M	W
	…	K	H	G	U		…	P	N

Fig. 15. An example to illustrate Lemmas 5 and 6.

(3) If G and T are D cells, then E will be an embeddable I cell (by Lemma 1), and S will be I cells (by Lemma 2). According to the number of D in U and Z, we apply the similar proof of (1) or (2) to get T to be an I cell. This make F become an I cell, then A will be an I cell. If (1) and (2) cannot be applied, this means that we have horizontal cells such as that from F to M are D cells except the end cell W. Since we restrict all boundary points not to be embeddable cell, W is not embeddable, and W is not D due to that L and N are not embeddable cells. This means that M has exact one of its 4-neighbors being D cell. By Lemma 4, M will become an I cell, then we get the left cell of M be an I cell. In the similar fashion, we get all horizontal cells from M to F be I cells, then A will be an I cell.

References

Cox, I.J., Miller, M.L., Bloom, J.A., Fridrich, J., Kalker, T., 2008. Digital Watermarking and Steganography, 2nd ed. Morgan Kaufmann.

Desoky, A., Younis, M., 2009. Chestega: chess steganography methodology. Security and Communication Networks 2 (6), 555–566.

Diehl, M., 2008. Secure covert channels in multiplayer games. In: Proceedings of the 10th ACM workshop on Multimedia and Security, pp. 117–122.

Farn, E.J., Chen, C.C., 2009a. Novel steganographic method based on jigswap puzzle images. Journal of Electronic Imaging 18 (1), 013003.

Farn, E.J., Chen, C.C., 2009b. Jigsaw puzzle images for steganography. Optical Engineering 48 (7), 077006.

Fun Printable Mazes for Kids, 2009. http://www.printactivities.com/Mazes.html (accessed 24.12.09).

Hernandez-Castro, J.C., Blasco-Lopez, I., Estevez-Tapiador, J.M., 2006. Steganography in games: a general methodology and its application to the game of Go. Computers and Security 25 (1), 64–71.

Katzenbeisser, S., Petitcolas, F.A.P., 2000. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, London.

Kieu, T.D., Wang, Z.H., Chang, C.C., Li, M.C., 2009. A sudoku based wet paper hiding scheme. International Journal of Smart Home 3 (April (2)), 1–12.

Kirkland, 2009. Maze Works – How to Build a Maze. http://www.mazeworks.com/mazegen/mazetut (accessed 24.12.09).

Lost in the maze, 2009. http://www.uptoten.com/kids/kidsgames-coordination-kwalamaze.html (accessed 24.12.09).

Niwayama, N., Chen, N., Ogihara, T., Kaneda, Y., 2002. A steganographic method for mazes. In: Proc. of Pacific Rim Workshop on Digital Steganography 2002 (STEG'02), July, Kitakyushu, Japan.

Novel Games, 2009. http://www.novelgames.com/flashgames/game.php?id=17&l=e (accessed 24.12.09).

Pullen, W., 2009. Think Labyrinth, Maze Algorithms. http://www.astrolog.org/labyrnth/algrithm.htm (accessed 24.12.09).

Shirali-Shahreza, M.H., Shirali-Shahreza, M., March 31–April 4 2008. Steganography in SMS by Sudoku puzzle. In: IEEE/ACS International Conference on Computer Systems and Applications, pp. 844–847.

WORLDVILLAGE KIDZ, 2009. http://www.worldvillage.com/kidz/puzzles/maze.htm (accessed 24.12.09).

Zander, S., Armitage, G., Branch, P., 2008. Covert Channels in Multiplayer First Person Shooter Online Games, LCN.

Zollner, J., Federrath, H., Klimant, H., Pfitzmann, A., Piotraschke, R., Westfeld, A., Wicke, G., Wolf, G., 1998. Modeling the security of steganographic systems. In: Proc. Second Workshop on Information Hiding, Portland, Oregon, USA, April, pp. 344–354.

Hui-Lung Lee was born in Taoyuan, Taiwan, R.O.C., on June 16, 1976. He received the M.S. degree in Computer Science and Information Engineering from Fu Jen Catholic University in 2003. Currently, he is a Ph.D. candidate of the Department of Computer Science at National Chiao Tung University. His major research interests include information hiding and watermarking.

Chia-Feng Lee received the M.S. degree in Computer Science from National Chiao Tung University, Hsinchu, Taiwan. His research interests include image processing and information hiding.

Ling-Hwei Chen was born in Changhua, Taiwan, in 1954. She received the B.S. degree in Mathematics and the M.S. degree in Applied Mathematics from National Tsing Hua University, Hsinchu, Taiwan in 1975 and 1977, respectively, and the Ph.D. degree in Computer Engineering from National Chiao Tung University, Hsinchu, Taiwan in 1987. From August 1977 to April 1979 she worked as a research assistant in the Chung-Shan Institute of Science and Technology, Taoyan, Taiwan, From May 1979 to February 1981 she worked as a research associate in the Electronic Research and Service Organization, Industry Technology Research Institute, Hsinchu, Taiwan. From March 1981 to August 1983 she worked as an engineer in the Institute of Information Industry, Taipei, Taiwan. She is now a Professor of the Department of Computer Science at the National Chiao Tung University. Her current research interests include image processing, pattern recognition, Multimedia compression, content-based retrieval and Multimedia Steganography.