Chapter 3
Covert Communication, Watermarking, Data Authentication, and Secret Sharing by Digital Puzzle Images

3.1 Overview of Proposed Method

3.1.1 Information Hiding in Digital Puzzle Image

In this chapter, we embed information into certain features of a digital puzzle image during the digital puzzle image creation process proposed in Section 2.2. We will describe how to hide data in the orientations, sizes, and angles of digital puzzle images in this chapter. The data could be a secret message, a watermark, or authentication signals, and we can embed any one kind of the secret data into any one of the feature of a digital puzzle image, by choice. We will embed a secret message by puzzle piece orientation modification in Section 3.3, embed a watermark by puzzle piece size modification in Section 3.4, and embed authentication signals by puzzle piece angle modification in Section 3.5, respectively. Of course, a puzzle piece feature detection process will also be discussed.

In principle, we embed data in the horizontal lines (denoted as $Xaxis_i$) of the digital puzzle image first, and then the rest of the data will be embedded in the vertical lines (denoted as $Yaxis_i$) of it. That is, we embed the data in the south side of each puzzle piece except the puzzle pieces which are located in the last row of the digital puzzle image, and embed the remaining data in the east side of each puzzle

piece except the puzzle pieces which are located in the rightmost column of the digital puzzle image. As illustrated in Figure 3.1, each number written on a purple circle indicates the data embedding order of each region.

Figure 3.1 The purple circles indicate the regions where the data are embedded.

3.1.2 Secret Sharing by Digital Puzzle Image

By applying the digital puzzle image decomposition and reconstruction processes proposed in Section 2.2.2.2 and Section 2.3, respectively, to deal with data-embedded digital puzzle image, we can fulfill a concept of secret sharing using the digital puzzle image. A flowchart of the proposed secret sharing concept is shown in Figure 3.2.

We randomly divide the data-embedded digital puzzles into several groups by applying the digital puzzle image decomposition process proposed in Section 2.2.2.2. In a sense, this process is similar to cutting apart a real treasure map. Each secret sharing participant will receive one group of the puzzle pieces, like receiving a piece of a torn map. After all of the puzzle pieces have been collected from the participants, we can perform the digital puzzle image reconstruction process proposed in Section 2.3, and the complete data-embedded digital puzzle image will be recovered automatically. Secret information embedded in the digital puzzle image will not be extracted successfully unless all of the participants have returned their own parts of

puzzle pieces in this process.

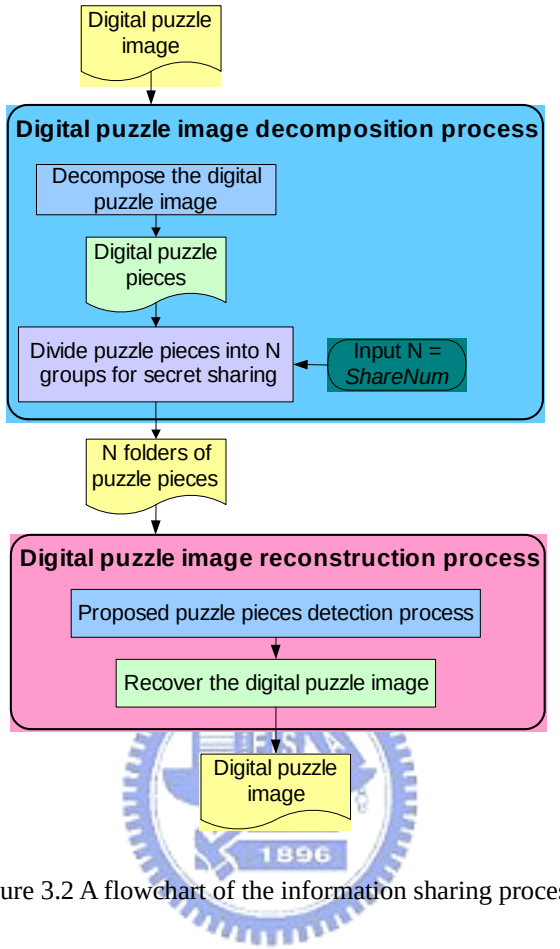


Figure 3.2 A flowchart of the information sharing process.

3.2 Proposed Puzzle Piece Feature Detection Techniques

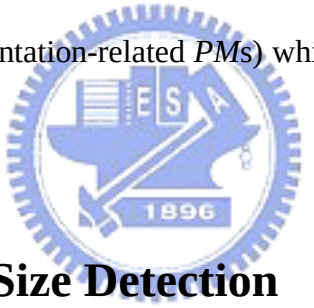
In this section, we will refer to all of the parameters which were defined in Chapter 2.

3.2.1 Puzzle Piece Orientation Detection

Before performing the proposed digital puzzle image reconstruction process proposed in Section 2.3, we have to figure out the *orientation values* (denoted as *OVs*)

of the four sides (denoted as *N, E, W,* and *S*, respectively) of each *puzzle piece region map* (denoted as *PM*). Then, we can apply the digital puzzle piece orientation detection process proposed in Section 2.3.2.1 when performing the puzzle piece orientation detection process in this chapter to find out the *orientation pixels number* (denoted as *OPN*) and the *OV* of each detected *PMs*.

However, different from the digital puzzle pieces orientation detection process proposed in Section 2.3.2.1, we do not have to scan all sides of each *PM* of a digital puzzle image. We only scan *S* of each *PM* except those *PMs* whose corresponding puzzle pieces are located in the last row of the digital puzzle image, and scan *E* of each *PM* except those *PMs* whose corresponding puzzle pieces are located in the rightmost column of the digital puzzle image in this section. We denote these kinds of *PMs* as *ORPM*s (meaning orientation-related *PM*s) while performing the puzzle piece orientation detection process.



3.2.2 Puzzle Piece Size Detection

We utilize the parameter *BigR* mentioned in Section 2.2.2.1 to modify the feature of size (denoted as *SZ*), which is the area of the big circle as shown in Figure 2.5. The value of the parameter *BigR* is either "*BR*" or "*SR*," and the values of *BR* and *SR* are derived by Formulas (2.1) and (2.2), respectively.

We only scan *S* of each *PM* except the *PMs* whose corresponding puzzle pieces are located in the last row of a digital puzzle image, and scan *E* of each *PM* except the *PMs* whose corresponding puzzle pieces are located in the most east column of the digital puzzle image, and we denote these kinds of *PMs* as *SRPM*s (meaning size-related *PM*s) while performing the puzzle piece size detection process.

While scanning each $SRPM_i$, we calculate the number of *black pixels* (denoted as

BPs) within the red rectangles as shown in Figure 3.3(b) and (d), and calculate the number of *white pixels* (denoted as *WPs*) within the red rectangles as shown in Figure 3.3(a) and (c), and we denote *BP* and *WP* as *size pixels numbers* (*SPNs*). We utilize the *SPN* of each $SRPM_i$ to figure out a *total size pixel number* (denoted as *SPNtotal*), and an *average size pixel number* (denoted as *SPNavg*).

The derived *SPNavg* will be utilized in the watermark extraction process proposed in Section 3.4.3 to figure out a *size value* (denoted as *SV*) of each detected *PM*. We denote the *SV* as *SVSmall* when we find out that the parameter *BigR* is equal to *SR*, and denote the *SV* as *SVBig* when we find out that the parameter *BigR* is equal to *BR*.

Algorithm 3.1: Puzzle piece size detection process.

Input: Digital puzzle piece images and their corresponding *SRPMs*.

Output: An *average size pixel number* (denoted as *SPNavg*).

Steps:

Step 1 Applying the puzzle piece orientation detection process proposed in Section 3.2.1, and figure out an *orientation value* (denoted as *OV*) of *E* and *S* of each $ORPM_i$ first.

Step 2 Scan each $SRPM_i$.

Case 1. : If the *OV* of *S* of the $ORPM_i$ is equal to -1, perform the following steps

2.1 Perform a raster scan of the $SRPM_i$, and calculate the number of the *WPs*, and denote this value as size pixels number (*SPN*). The scanning region is illustrated in Figure 3.3(a).

2.2 Scan all of the $SRPM_i$ in Case 1, and sum up the *SPNs* of them.

Case 2. : If the *OV* of *S* of the $ORPM_i$ is equal to 1, perform the following

steps.

2.3 Perform a raster scan of the $SRPM_i$, and calculate the number of the *BPs*, and denote this value as *SPN*, too. The scanning region is illustrated in Figure 3.3(b).

2.4 Scan all of the $SRPM_i$ in Case 2, and sum up the *SPNs* of them.

Case 3. : If the *OV* of *E* of the $ORPM_i$ is equal to -1, perform the following steps.

2.5 Perform a raster scan of the $SRPM_i$ and calculate the number of the *WPs*, and denote this value as *SPN*, too. The scanning region is illustrated in Figure 3.3(c).

2.6 Scan all of the $SRPM_i$ in Case 3, and sum up the *SPNs* of them.

Case 4. : If the *OV* of *E* of the $ORPM_i$ is equal to 1, perform the following steps.

2.7 Perform a raster scan of the $SRPM_i$, and calculate the number of the *BPs*, and denote this value as *SPN*, too. The scanning region is illustrated in Figure 3.3(d).

2.8 Scan all of the $SRPM_i$ in Case 4, and sum up the *SPNs* of them.

Step 3 Figure out a *total size pixel number* (denoted as *SPNtotal*) by summing up all of the *SPNs* derived from the four cases discussed in Step 2.

Step 4 Figure out an *average size pixel number* (denoted as *SPNavg*) by dividing the *SPNtotal* by the number of $SRPM_i$.

(a) (b) (c) (d)

Figure 3.3 Scanning regions of the puzzle piece size detection process are within the red rectangles.

3.2.3 Puzzle Piece Angle Detection

We utilize the parameter θ mentioned in Section 2.2.2.1 to modify the feature of angle (denoted as *AG*), which is a notation shown in Figure 2.5. The value of θ is taken to be either "40^o" or "48^o" in this study.

In this section, we only scan *S* of each *PM* except the *PMs* whose corresponding puzzle pieces are located in the last row of a digital puzzle image, and scan *E* of each *PM* except the *PMs* whose corresponding puzzle pieces are located in the rightmost column of the digital puzzle image, and we denote these kinds of *PMs* as *ARPM*s (meaning angle-related *PM*s) while performing the puzzle piece angle detection process.

While scanning *S* and *E*, respectively, of each $ARPM_i$, we calculate the number of *black pixels* (denoted as *BPs*) of each side, and we denote the value of *BP* as an *angle pixel number* (*APN*).

The derived *APN* will be utilized by the authentication signal extraction process proposed in Section 3.5.3 to figure out an *angle value* for each detected *PM*. We denote the *angle value* as *AVSmall* when we find out that the parameter θ is equal to 40^o, and denote it as *AVBig* when we find out that the parameter θ is equal to 48^o.

Algorithm 3.2: Puzzle piece angle detection process.

Input: Digital puzzle piece images and their corresponding *ARPMs.*

Output: A parameter *angel pixels number (APN).*

Steps:

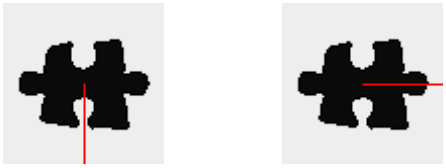
Step 1 Scan *S* of each $ARPM_i$ and perform the following step. The scanning region is shown in Figure 3.4(a).

1.1 Calculate the number of the *BPs* on the scanning line, and denote this

value as APN_i.

Step 2 Scan E of each $ARPM_i$ and perform the following step. The scanning region is shown in Figure 3.4(b).

2.1 Calculate the number of the *BPs* on the scanning line, and also denote this value as APN_i.



(a) (b)

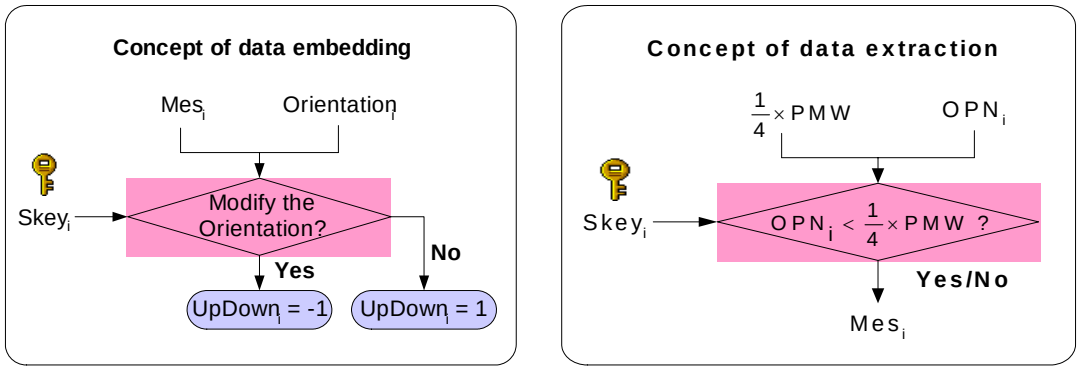
Figure 3.4 The scanning region of the puzzle piece angle detection process is on the red line.

3.3 Proposed Secret Hiding Method by Puzzle Orientation Modification

3.3.1 Core Concept

The core concept of secret data hiding method by puzzle orientation modification is shown in Figure 3.5(a). Assume that we want to embed a bit sequence of a secret message (denoted as Mes_i) into an $ORPM_i$, which was defined in Section 3.2.1. As shown in Figure 3.5(a), by checking the input bit sequence of a secret key (denoted as $SKey_i$), we can know whether we need to modify the *orientation value* (denoted as *OV*) or not. If the answer is "Yes," we will assign a value "-1" to the parameter *UpDown*, which was mentioned in Section 2.2.2.1, and assign a value "1" to the parameter *UpDown* while the answer is "No."

As shown by the flowchart of the data extraction process in Figure 3.5(b), for each $ORPM_i$, we try to figure out whether it is an indent or an outdent side by comparing the value of $\frac{1}{4} \times PMW$ with the value of OPN_i. By the checking result and the input $SKey_i$, we can get the embedded Mes_i.



(a) (b)

Figure 3.5 Concepts of data embedding and data extraction by puzzle orientation modification.

3.3.2 Secret Message Embedding Process

The proposed information embedding process is similar to the digital puzzle image creation process proposed in Section 2.2.2.1. When performing the digital puzzle image creation process, we randomly assign a value "1" or "-1" to a parameter *UpDown*, but when performing this proposed process, the value of the parameter *UpDown* is decided by an input secret message, and a secret key (denoted as $SKey_i$).

Algorithm 3.3: Information embedding by digital puzzle image orientation modification.

Input: A digital image, secret message, and $SKey_i$.

Output: A secret message embedded digital puzzle image.

Steps:

Step 1 Transform the secret message into a bit sequence (denoted as Mes_i).

Step 2 Append an *ending pattern* (sixteen successive 0s) at the end of Mes_i, that is, $Mes_i = 0$, if $n-16 < i \le n$.

Step 3 Decide a value of $UpDown_i$ by performing a bitwise Exclusive OR operation to Mes_i and $SKey_i$. A diagrammatic explanation is shown in Figure 3.6.

Step 4 Execute the digital puzzle image creation process proposed in Section 2.2.2.1 by utilizing the value of $UpDown_i$ derived from Step 3.

In Step 2, by utilizing the ending pattern, we can determine where the secret information ends in a sequence of extracted bits after performing the information extraction process.

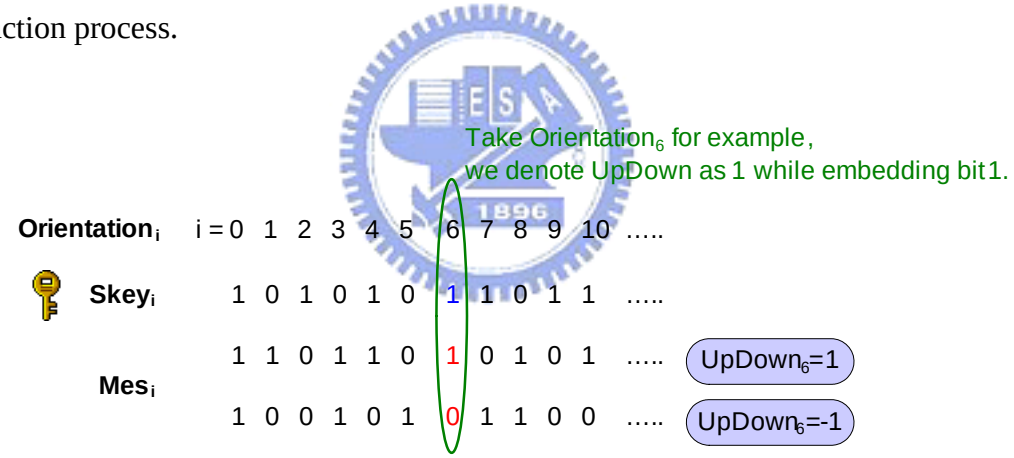


Figure 3.6 A bitwise Exclusive OR operation applied to Mes_i and $SKey_i$.

3.3.3 Secret Message Extraction Process

The proposed secret information extraction process is also an inverse of the embedding process. A flowchart of the secret information extraction process is shown in Figure 3.7.

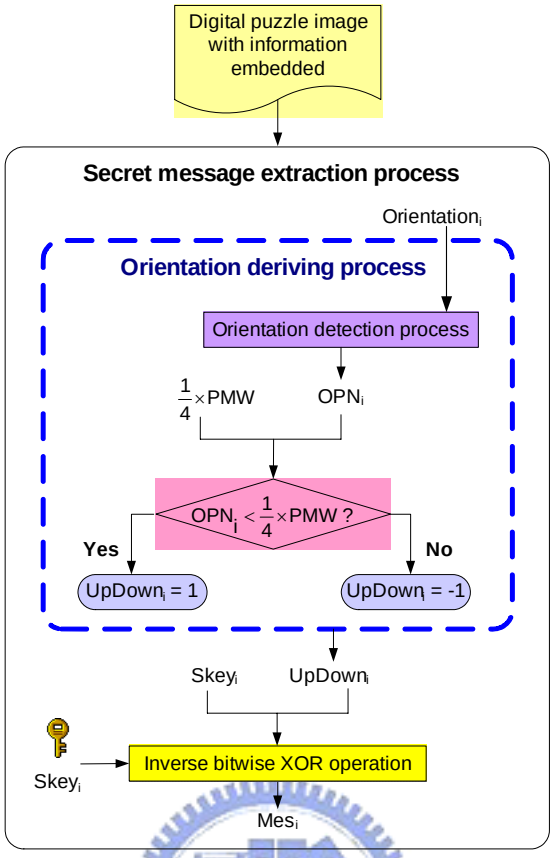


Figure 3.7 A flowchart of the secret information extraction process.

In this section, we will describe how we apply the puzzle piece orientation detection process proposed in Section 3.2.1, and the algorithm of the information extraction process is discussed as follows.

Algorithm 3.4: Information extraction by digital puzzle image orientation detection process.

Input: A secret message embedded digital puzzle image, an $ORPM_i$, and an $SKey_i$.

Output: A Mes_i.

Steps:

Step 1 Detect each $ORPM_i$ of the secret message embedded digital puzzle image

Step 2 Derive an OPN_i by performing the orientation detection process proposed in Section 3.2.1.

Step 3 Compare each OPN_i with $\frac{1}{4}\times PMW$ in the following way.

3.1 If $OPN_i < \frac{1}{4}\times PMW$, set $UpDown_i = 1$.

3.2 If $OPN_i > \frac{1}{4}\times PMW$, set $UpDown_i = -1$.

Step 4 Decide the value of Mes_i by performing an inverse bitwise Exclusive OR operation to the $UpDown_i$ and the $SKey_i$, and a diagrammatic explanation is shown in Figure 3.8.

Step 5 Search the Mes_i for the *ending pattern* (16 successive 0s) and truncate the redundant bits at the rear of the Mes_i.

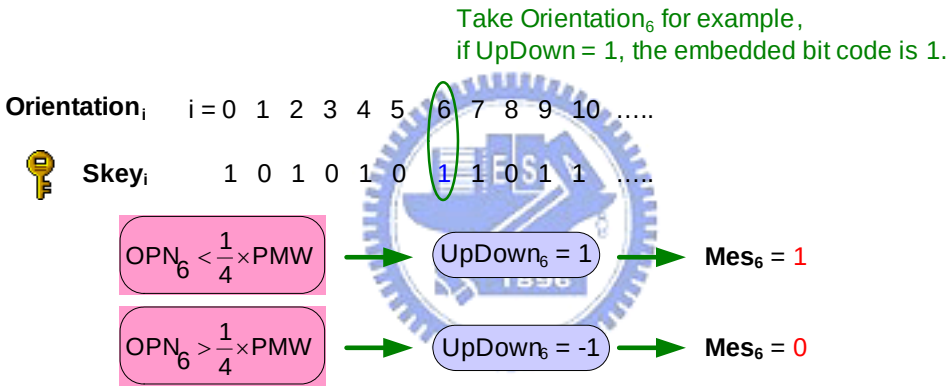
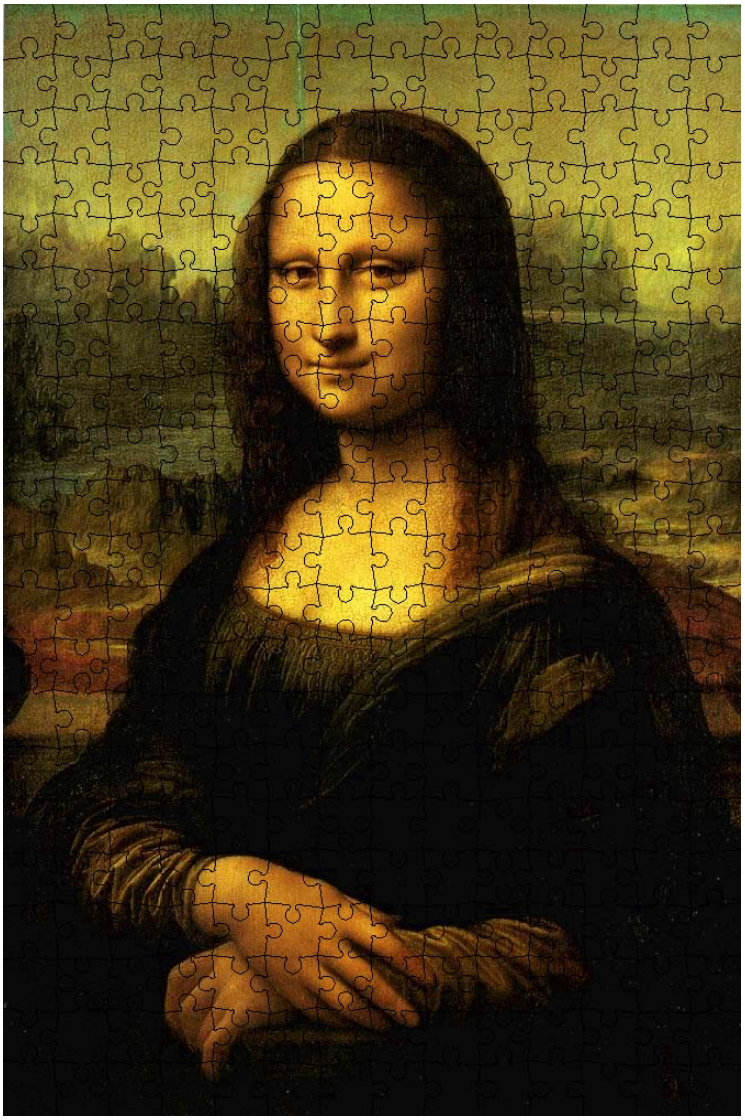


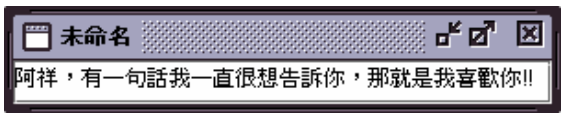
Figure 3.8 The inverse bitwise Exclusive OR operation applied to $UpDown_i$ and $SKey_i$.

3.3.4 Experimental Results

Some experimental results are shown in Figure 3.9. Figure 3.9(a) is a digital puzzle image with the secret message, “阿祥，有一句話我一直很想告訴你，那就是我喜歡你!!,” embedded. Figure 3.9(b) shows the secret message extracted from Figure 3.9(a). Figure 3.9(c) is the secret message extracted from Figure 3.9(a) with a wrong key. Therefore, Figure 3.9(c) shows the secret message is protected by the key properly.



(a)



(b)

(c)

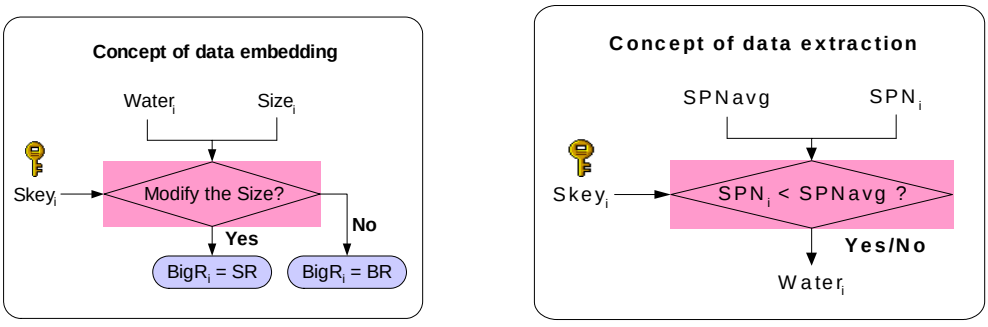
Figure 3.9 Experimental results. (a) A digital puzzle image with a secret message embedded. (b) The secret message extracted from (a) with a correct key. (c) The secret message extracted from (a) with a wrong key.

3.4 Proposed Watermarking Method by Puzzle Size Modification

3.4.1 Core Concept

The core concept of the proposed watermarking method by puzzle size modification is shown in Figure 3.10(a). Before embedding a watermark in a digital puzzle image, we transform a watermark into a bit sequence (denoted as $Water_i$) first, and embed it into each $SRPM_i$, which is defined in Section 3.2.2. By checking an input secret key (denoted as $SKey_i$), we can know whether we need to modify the feature of size or not. If the answer is yes, we denote the parameter $BigR_i$ as *SR*, and if the answer is no, we denote the parameter $BigR_i$ as *BR*.

As shown by the flowchart of the data extracting process in Figure 3.10(b), by applying the puzzle piece size detection proposed in Section 3.2.2, we can figure out the *size pixels number* (denoted as SPN_i) of each $SRPM_i$ and the *average size pixel number* (denoted as *SPNavg*). If the SPN_i is bigger than *SPNavg*, we can figure out that the *size value* (denoted as *SV*) of $SRPM_i$ is *SVBig*, and if the SPN_i is smaller than *SPNavg*, we can figure out that the *SV* of $SRPM_i$ is SVSmall. By the checking result and the input $SKey_i$, we can get the embedded $Water_i$.



(a) (b)

Figure 3.10 Concepts of data embedding and data extraction by puzzle size modification.

3.4.2 Watermark Embedding Process

When performing the digital puzzle image creation process proposed in Section 2.2.2.1, we randomly assign the value $PPS \times 4.2/27$ (SR) or $PPS \times 4.2/27 + 1$ (BR) to the parameter *BigR*, but when performing the proposed process in this section, the value of the parameter *BigR* is decided by the input $Water_i$ and a $SKey_i$.

Algorithm 3.5: Watermark embedding by digital puzzle image size modification.

Input: A digital image, a watermark, and a $SKey_i$.

Output: A watermark embedded digital puzzle image.

Steps:

Step 1 Transform the watermark into a bit sequence (denoted as $Water_i$).

Step 2 Append an *ending pattern* (sixteen successive 0s) at the end of $Water_i$, that is, set $Water_i = 0$ if $n - 16 < i \le n$.

Step 3 Decide a value of $BigR_i$ by performing a bitwise Exclusive OR operation to $Water_i$ and $SKey_i$. A diagrammatic explanation is shown in Figure 3.11.

Step 4 Perform the digital puzzle image creation process proposed in Section 2.2.2.1 by utilizing the parameter $BigR_i$ derived from Step 3.

In Step 2, by utilizing the *ending pattern*, we can determine where the watermark ends in a sequence of extracted bits after performing the information extraction process.

Figure 3.11 A bitwise Exclusive Or operation between $Water_i$ and $SKey_i$.

3.4.3 Watermark Extraction Process

The proposed watermark extraction process is an inverse of the embedding process. A flowchart of the watermark extraction process is shown in Figure 3.12.

In this section, we describe how we apply the secret message extraction process and the puzzle piece size detection process proposed in Section 3.3.3 and 3.2.2, and the algorithm of the watermark extraction process is described as follows.

Algorithm 3.6: Process of watermark extraction by digital puzzle image size detection.

Input: A watermark-embedded digital puzzle image, an $SRPM_i$, and an $SKey_i$.

Output: A $Water_i$.

Steps:

Step 1 Detect each $SRPM_i$ of the watermark-embedded digital puzzle image.

Step 2 Derive an *average size pixel number* (denoted as *SPNavg*) and a *size pixels number* (denoted as SPN_i) of each $SRPM_i$ by performing the puzzle piece size detection process proposed in Section 3.2.2.

Step 3 Compare the SPN_i with the *SPNavg*, and perform the following steps.

3.1 If $SPN_i > SPNavg$, that is, if *size value*$_i$ (denoted as SV_i) is *SVBig*, then set $BigR_i = BR$.

3.2 If $SPN_i < SPNavg$, that is, if the SV_i is *SVSmall*, then set $BigR_i = SR$.

Step 4 Decide the $Water_i$ by performing an inverse bitwise Exclusive OR operation to the $BigR_i$ and the $SKey_i$, and a diagrammatic explanation is shown in Figure 3.13.

Step 5 Search the $Water_i$ for the *ending pattern* (16 successive 0s) and truncate the redundant bits at the rear of the $Water_i$.

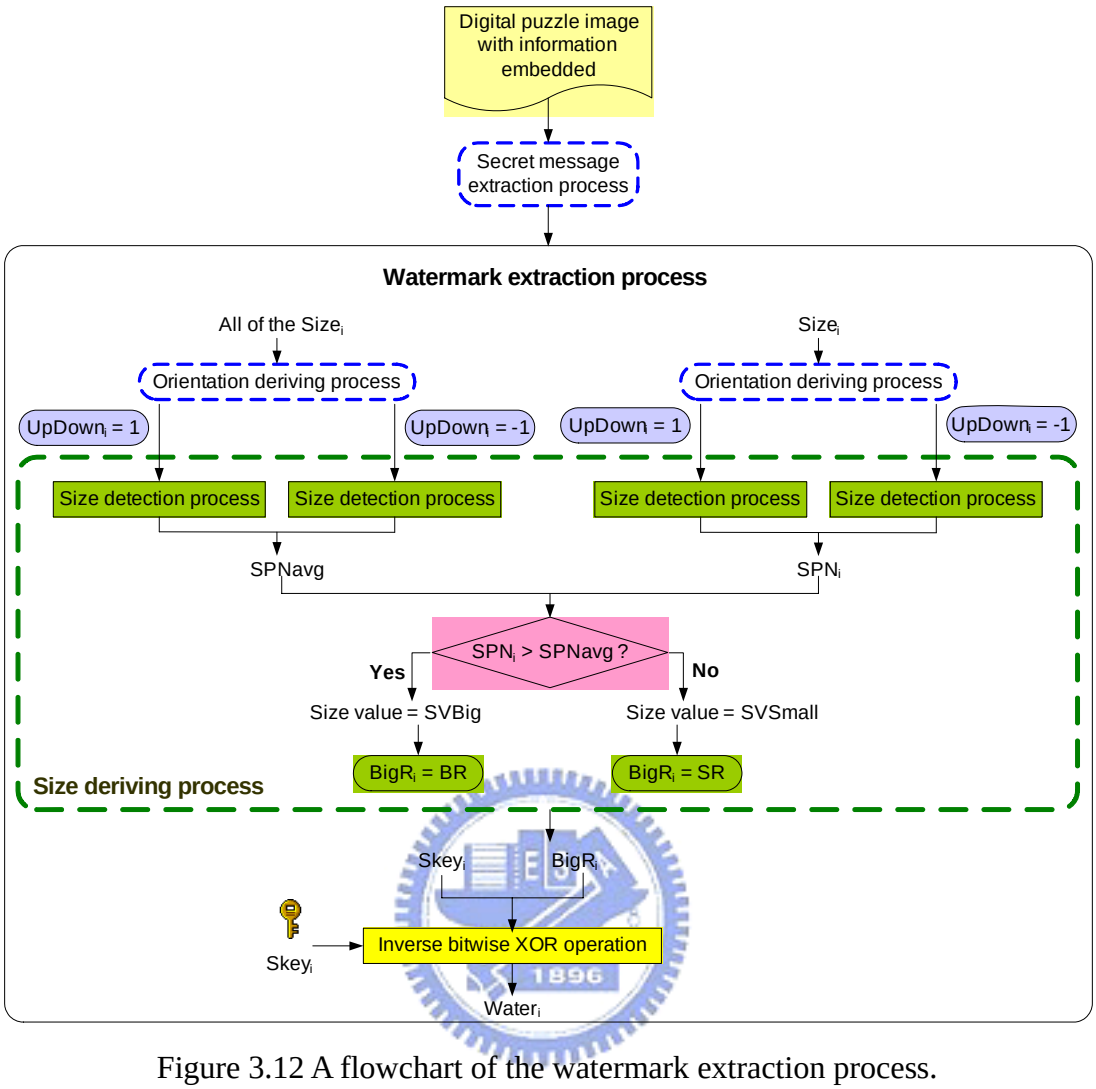


Figure 3.12 A flowchart of the watermark extraction process.

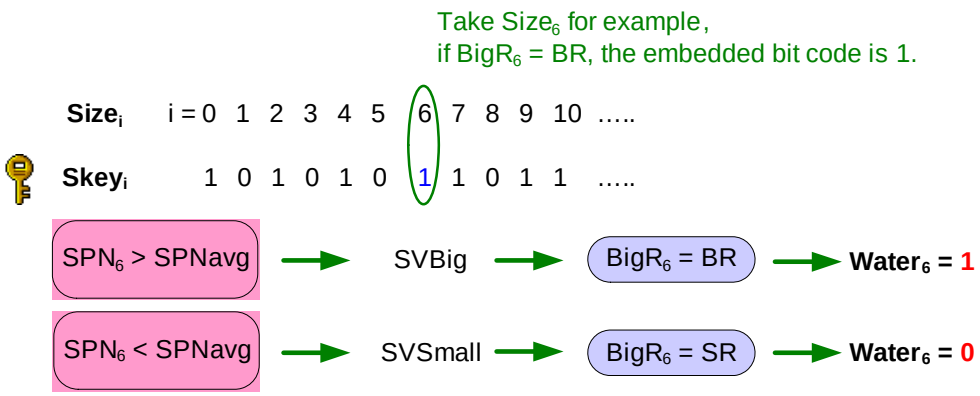
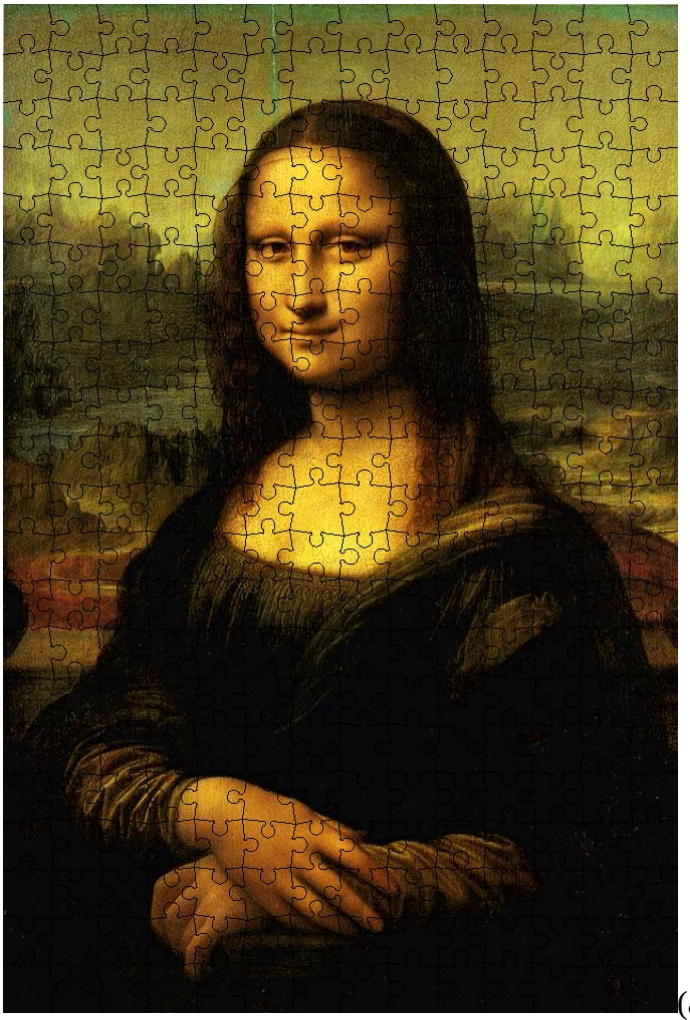


Figure 3.13 An inverse bitwise Exclusive OR operation applied to $BigR_i$ and $SKey_i$.

3.4.4 Experimental Results

Some experimental results are shown in Figure 3.14. Figure 3.14(b) is an input watermark. Figure 3.14(a) is a digital puzzle image with the watermark Figure 3.14(b) embedded. Figure 3.14(c) is a watermark extracted from Figure 3.14(a). From Figure 3.14(d), we can see that the watermark can-not be extracted with a wrong key.


(a)


(b)


(c)


(d)

Figure 3.14 Experimental results. (a) A digital puzzle image with the watermark (b) embedded. (b) An input watermark. (c) A watermark extracted from (a) with a correct key. (d) A watermark extracted from (a) with a wrong key.

3.5 Proposed Authentication Method by Puzzle Angle Modification

3.5.1 Core Concept

The core concept of the proposed authentication method by puzzle angle modification is shown in Figure 3.15. We propose the use of a hash function to derive authentication signals (denoted as AS_i) first, and then embed them into each $ARPM_i$, which was defined in Section 3.2.3. By checking the AS_i, we can know whether we need to modify the feature of angle or not.

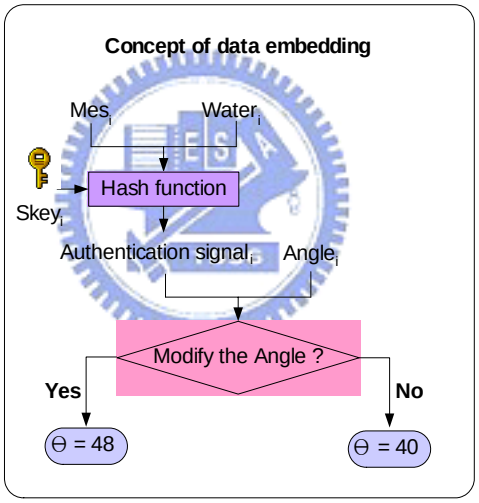


Figure 3.15 Concepts of data embedding process by puzzle Angle modification.

As shown by the flowchart of the data extracting process in Figure 3.16, by applying the puzzle piece angle detection process proposed in Section 3.2.3, we can figure out an *angel pixels number* (denoted as *APN*) of each $ARPM_i$. Before utilizing the APN_i to derive the embedded AS_i, we apply the puzzle piece orientation and size detection processes proposed in Section 3.2.1 and 3.2.2, respectively first, and then derive the values of *UpDown* and *BigR*. By utilizing the values of *UpDown* and *BigR*,

we can divide the $ARPM_i$ into four kinds and start to detect the authentication signals. By the checking result, we can get the embedded AS_i.

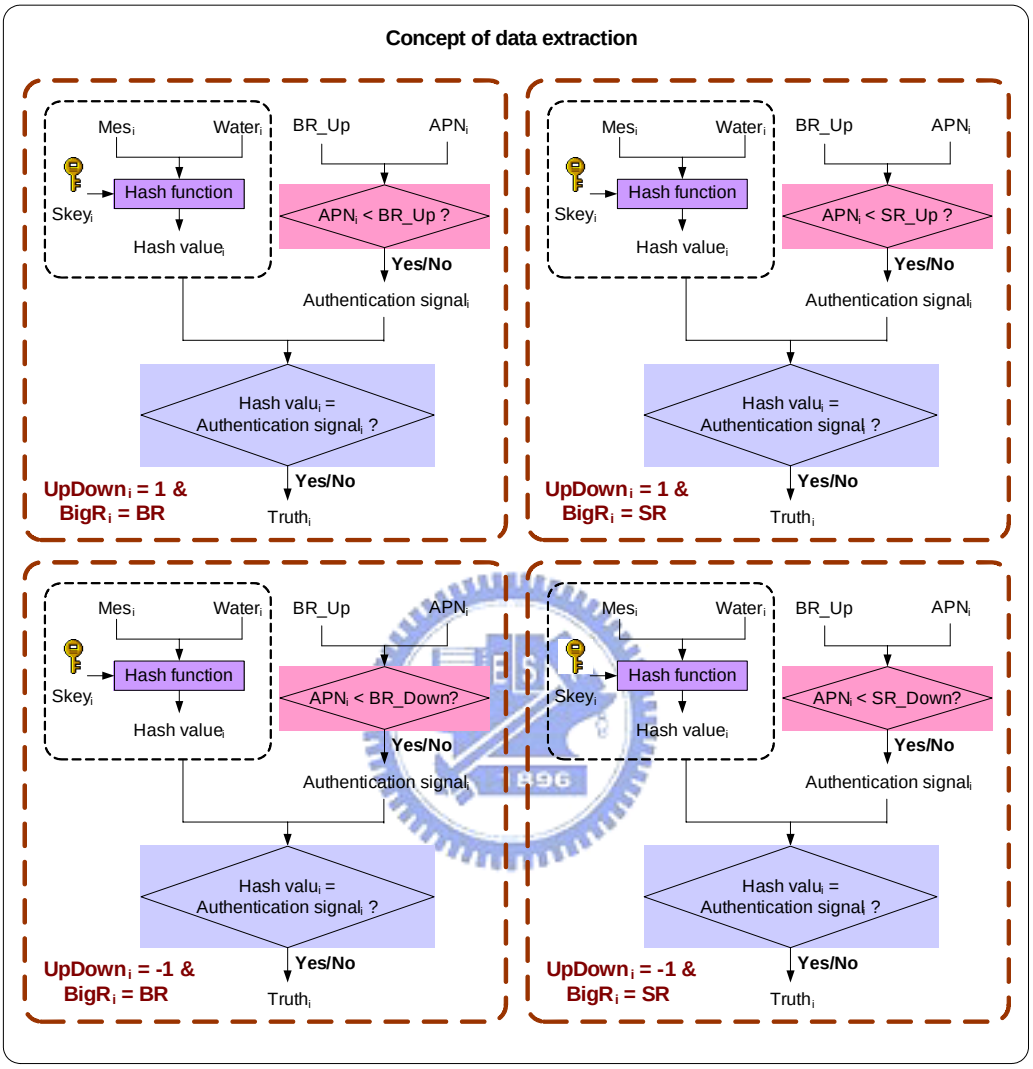


Figure 3.16 Concepts of data extraction process by puzzle angle modification.

3.5.2 Authentication Signal Embedding Process

When performing the digital puzzle image creation process proposed in Section 2.2.2.1, we randomly assign a value "40^o" or "48^o" to a parameter θ, but when performing the proposed process in this section, the value of the parameter θ is decided by input authentication signals (denoted as AS_i).

Algorithm 3.7: Authentication signal embedding by digital puzzle image angle modification.

Input: A digital image, a Mes_i, a $Water_i$, a $Skey_i$., and an empty array of $OrientationHash_i$, and an empty array of $SizeHash_i$.

Output: An authentication-signal-embedded digital puzzle image.

Steps:

Step 1 Derive authentication signals (denoted as AS_i) by performing the following steps.

1.1 Read in the bit code Mes_i, and assign the bit value of each Mes_i to each $OrientationHash_i$ in order.

1.2 Read in the bit code $Water_i$, and assign the bit value of each $Water_i$ to each $SizeHash_i$ in order.

1.3 Derive a hash value (denoted as $Hash_i$) by applying the hash function as shown in Formula (3.1) below:

$$Hash_i = ((OrientationHash_i + 1) \times (SizeHash_i + 1) \times Random(Skey_i)) \bmod 3 \bmod 2 . \quad (3.1)$$

1.4 Read in the $Hash_i$, and assign the bit value of each $Hash_i$ to each AS_i in order.

Step 2 Decide a value of θ_i by checking the AS_i in the following way.

2.1 If the AS_i is 1, assign a value "40^o" to the θ_i.

2.2 If the AS_i is 0, assign a value "48^o" to the θ_i.

Step 3 Performing the digital puzzle image creation process proposed in Section 2.2.2.1 by utilizing the value of the θ_i derived from Step 3.

3.5.3 Authentication Signal Extraction Process

The proposed authentication signal extraction process is an inverse of the embedding process. A flowchart of the authentication signal extraction process by puzzle angle modification is shown in Figure 3.17.

In this section, we describe how we apply the secret message extraction process, the watermark extraction process, and the puzzle piece angle detection process proposed in Section 3.3.3, 3.4.3, and 3.2.3, respectively. And the algorithm of the proposed authentication signal extraction process is described as follows.

A parameter *neck value* is used to represent the height ($\overline{AM}$) of the triangle shown in Figure 2.5. We denote the parameter *neck value* as *NVSmall* when the value of θ shown in Figure 2.5 is $40°$, and denote it as *NVBig* when θ is $48°$. We can derive the values of *NVSmall* and *NVBig* by applying Formulas (3.1) and (3.2) below, respectively:

$$NVSmall = \frac{PPS/6}{\tan 40^{\circ}}; \tag{3.2}$$

$$NVBig = \frac{PPS/6}{\tan 48^{\circ}}. \tag{3.3}$$

Before performing the information extraction process in this section, we should figure out the values of some parameters. We derive the values of *BR*, *SR*, *NVSmall*, and *NVBig* by applying Formulas (2.1), (2.2), (3.2), and (3.3), respectively first, and the values of *STheta_BR*, *BTheta_BR*, *STheta_SR*, *BTheta_SR*, *BR_Up*, *SR_Up*, *BR_Down*, and *SR_Down* by applying Formulas (3.4) through (3.11), respectively, below:

$$STheta\_BR = NVSmall + BR; \tag{3.4}$$

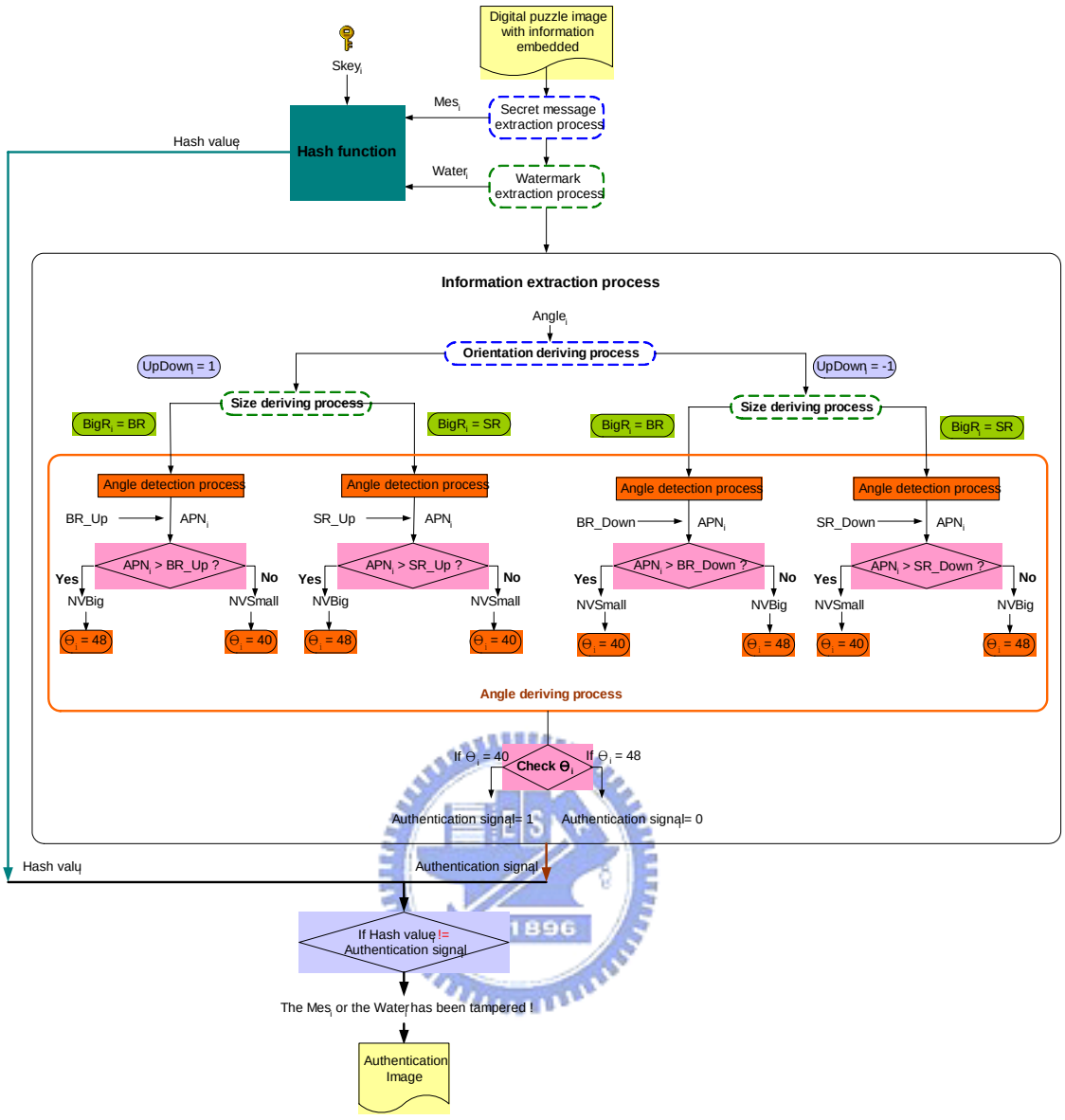


Figure 3.17 A flowchart of the Authentication signals extraction process by puzzle angle modification.

$$BTheta\_BR = NVBig + BR; \tag{3.5}$$

$$STheta\_SR = NVSmall + SR; \tag{3.6}$$

$$BTheta\_SR = NVBig + SR; \tag{3.7}$$

$$BR\_Up = \frac{PPS - STheta\_BR - BTheta\_BR}{2}; \tag{3.8}$$

$$SR\_Up = \frac{PPS - STheta\_SR - BTheta\_SR}{2}; \quad (3.9)$$

$$BR\_Down = \frac{PPS + STheta\_BR + BTheta\_BR}{2}; \quad (3.10)$$

$$SR\_Down = \frac{PPS + STheta\_SR + BTheta\_SR}{2}. \quad (3.11)$$

Algorithm 3.8: Process of authentication signal extraction by digital puzzle image angle detection.

Input: A digital puzzle image in which a secret message, a watermark, and authentication signals are embedded; an $Skey_i$; and an $ARPM_i$.

Output: Authentication signals (denoted as AS_i), and an authenticated image.

Steps:

Step 1 Apply the secret message and the watermark extraction processes proposed in Section 3.3.3 and 3.4.3, respectively, in the following

1.1 Apply these processes to extract Mes_i and $Water_i$.

1.2 Apply these processes to figure out the values of $UpDown_i$ and $BigR_i$ of each $ARPM_i$.

Step 2 Derive a Hash value (denoted as $Hash_i$) by applying the hash function shown in Formula (3.1) utilizing the extracted Mes_i and $Water_i$.

Step 3 Figure out the values of *BR_Up*, *SR_Up*, *BR_Down*, and *SR_Down* by applying Formulas (3.8) to (3.11), respectively.

Step 4 Derive an *angle pixels number* (denoted as APN_i) of each $ARPM_i$ by performing the puzzle piece angle detection process proposed in Section 3.2.3.

Step 5 Compare each APN_i with the value of *BR_Up*, *SR_Up*, *BR_Down*, or

SR_Down in the following four kinds of cases.

Case 1. : If the parameter $UpDown_i$ of $ARPM_i$ is 1, and the parameter $BigR_i$ of the $ARPM_i$ is *BR*, then compare APN_i with *BR_Up* in the following way.

5.1 If $APN_i > BR\_Up$, that is, if the *neck value* is *NVBig*, then set $\theta_i = 48°$.

5.2 If $APN_i < BR\_Up$, that is, if the *neck value* is *NVSmall*, then set $\theta_i = 40°$.

Case 2. : If the parameter $UpDown_i$ of $ARPM_i$ is 1, and the parameter $BigR_i$ of the $ARPM_i$ is *SR*, then compare APN_i with *SR_Up* in the following way.

5.3 If $APN_i > SR\_Up$, that is, if the *neck value* is *NVBig*, then set $\theta_i = 48°$.

5.4 If $APN_i < SR\_Up$, that is, if the *neck value* is *NVSmall*, then set $\theta_i = 40°$.

Case 3. : If the parameter $UpDown_i$ of $ARPM_i$ is -1, and the parameter $BigR_i$ of the $ARPM_i$ is *BR*, then compare APN_i with *BR_Down* in the following way.

5.5 If $APN_i > BR\_Down$, that is, if the *neck value* is *NVSmall*, then set $\theta_i = 40°$.

5.6 If $APN_i < BR\_Down$, that is, if the *neck value* is *NVBig*, then set $\theta_i = 48°$.

Case 4. : If the parameter $UpDown_i$ of $ARPM_i$ is -1, and the parameter $BigR_i$ of the $ARPM_i$ is *SR*, then compare APN_i with *SR_Down* in the following way.

5.7 If $APN_i > SR\_Down$, that is, if the *neck value* is *NVSmall*, then

set $\theta_i = 40°$.

5.8 If $APN_i < SR\_Down$, that is, if the *neck value* is *NVBig*, then set $\theta_i = 48°$.

Step 6 If $\theta_i = 40°$, then denote the bit value of the AS_i as 1, and if $\theta_i = 48°$, then denote the bit value of the AS_i as 0.

Step 7 Compare the $Hash_i$ with the AS_i: if the $Hash_i$ is not equal to the AS_i, decide that the i_{th} bit of the Mes_i or the $Water_i$ has been tampered.

Step 8 Create an authentication image according to the above comparison results.

3.5.4 Experimental Results

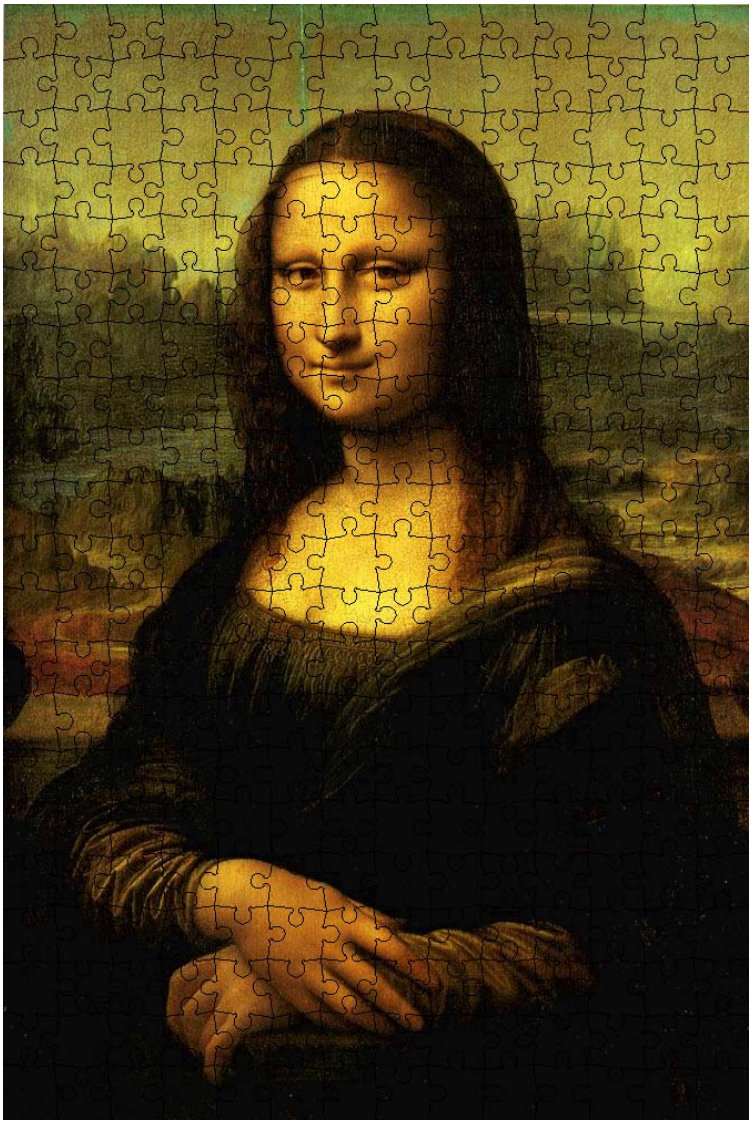
As discussed in the previous sections, we can verify the fidelity of embedded secret information by the proposed digital puzzle image authentication system. Because we embed only one bit into a puzzle piece angle, it is $\frac{1}{2}$ in probability that we may miss to detect a puzzle piece that has been damaged. However, if one damaged bit has successfully been detected by this system, we can regard this digital puzzle image as being tampered with.

Figure 3.18(a) is a digital puzzle image with a secret message, a watermark, and authentication signals embedded. Figure 3.18(b) is a secret message extracted from Figure 3.18(a), and Figure 3.18(c) is a watermark extracted from Figure 3.18(a). Figure 3.18(d) is an authentication image, which shows a verification result of Figure 3.18(a). In Figure 3.18(d), we can see that there are only black dots on it, which means that all of the puzzle pieces have not been modified. By Figure 3.18(b), (c), and (d), we can claim that a secret message, a watermark, and authentication signals

can be embedded in a digital puzzle image simultaneously by utilizing the proposed methods.

Figure 3.18(e) is a copy of Figure 3.18(a) with some puzzle pieces being tampered with. Figure 3.18(h) shows a verification result of Figure 3.18(e), we can see that there are some red dots appearing on it. The locations of the red dots indicate those puzzle pieces which have been modified. Figure 3.18(i) shows two pieces of original digital puzzle pieces, and Figure 3.18(j) shows two pieces of digital puzzle pieces which have been tampered with, and the yellow notations shown in Figure 3.18(k) indicate the locations of the puzzle pieces shown in Figure 3.18(i) and (j). If people who know the proposed secret message embedding process plan to tamper with the secret message, they can replace some original puzzle piece images with some modified puzzle piece images, such as replacing the puzzle pieces shown in Figure 3.18(i) with those shown in Figure 3.18(j). Figure 3.18(f) shows a secret message extracted from Figure 3.18(e), and Figure 3.18(g) shows a watermark extracted from Figure 3.18(e). We can see that both the secret message and the watermark have been extracted successfully, which means that the watermark has not been modified, and the secret message “might” not have been modified. However, because some modified digital puzzle images are found by the proposed authentication process, users can decide whether he/she will accept the extracted secret message or not by the authentication image shown in Figure 3.18(h).

(a)

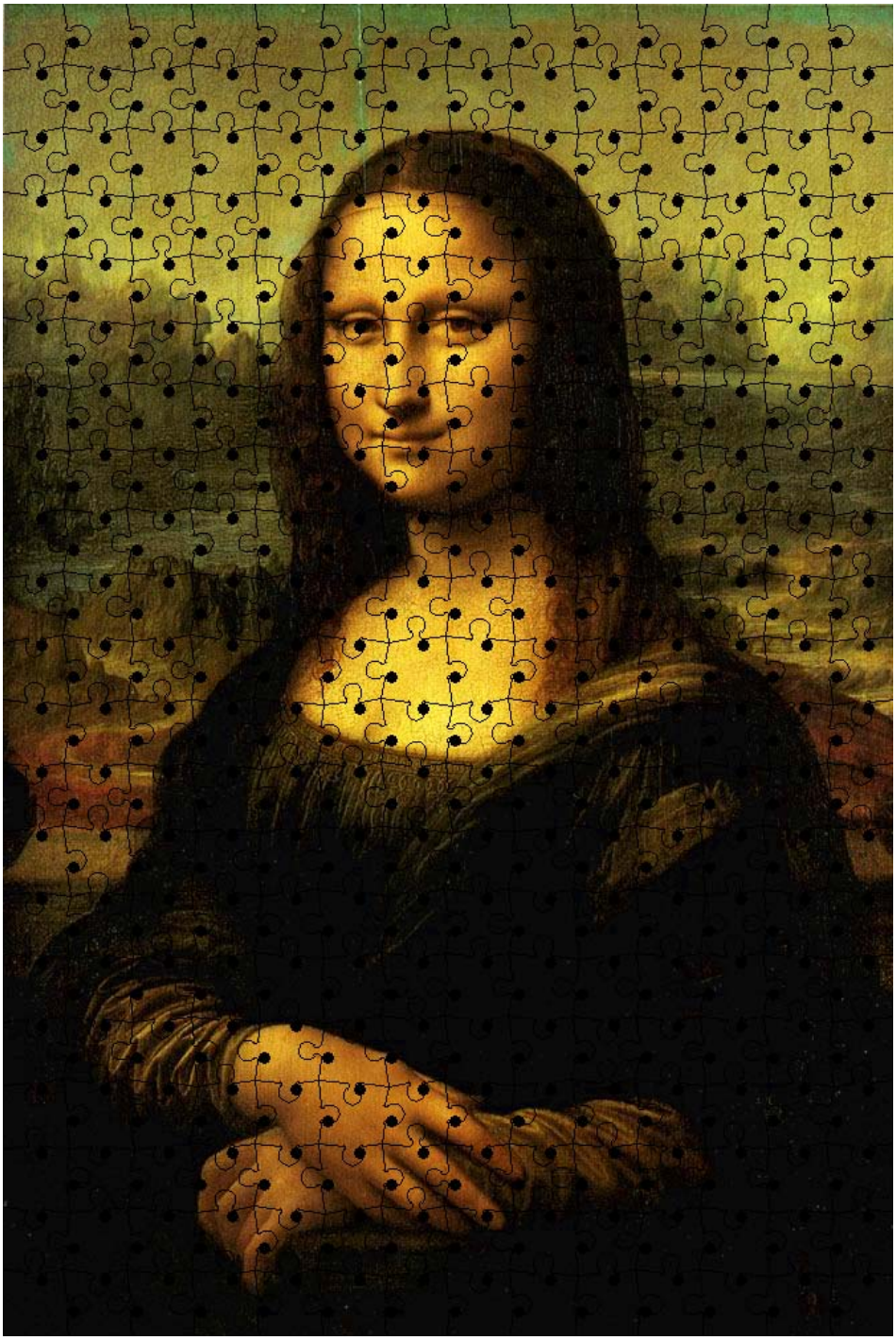


(b)

(c)

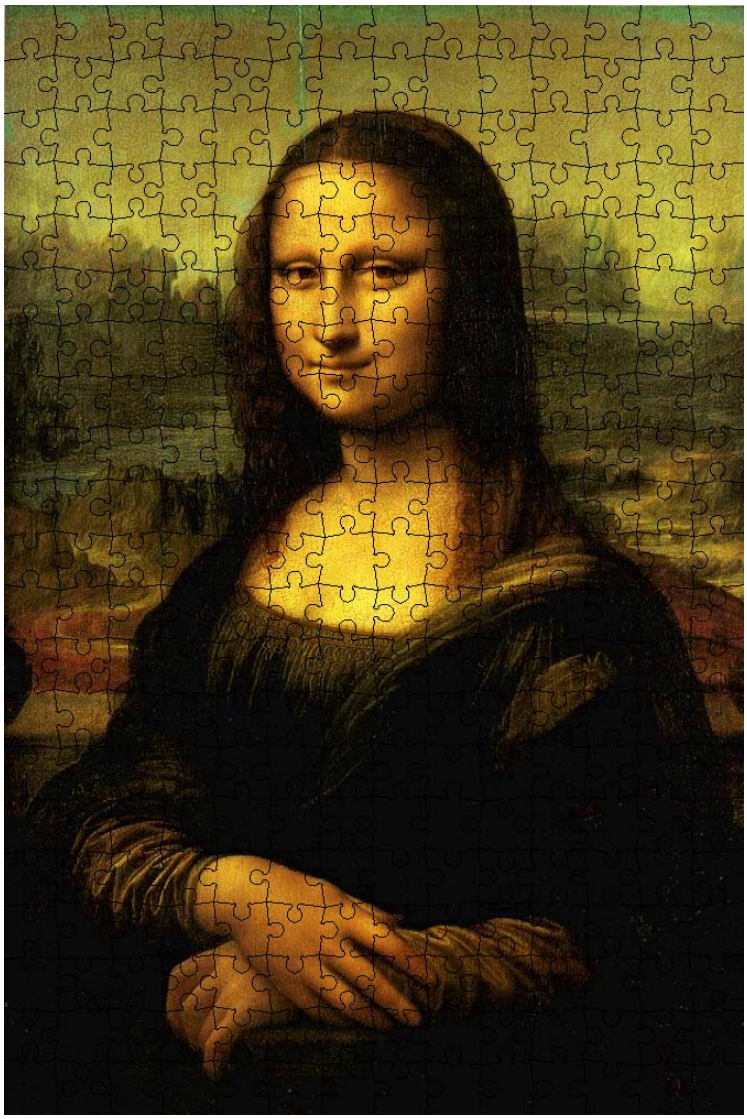
(d)

Figure 3.18 Experimental results. (a) An original image. (b) A digital puzzle image with a secret message, a watermark, and authentication signals embedded. (c) A secret message extracted from (b). (d) A watermark extracted from (b). (e) A verification result of (b). (f) A copy of (b) with some tampered puzzle pieces. (g) A secret message extracted from (f). (h) A watermark extracted from (f). (i) A verification result of (f). (j) Two pieces of original digital puzzle pieces. (k) Two pieces of tampered digital puzzle pieces. (l) The yellow notations indicate locations of (j) and (k).



(e)

Figure 3.18 Experimental results. (a) An original image. (b) A digital puzzle image with a secret message, a watermark, and authentication signals embedded. (c) A secret message extracted from (b). (d) A watermark extracted from (b). (e) A verification result of (b). (f) A copy of (b) with some tampered puzzle pieces. (g) A secret message extracted from (f). (h) A watermark extracted from (f). (i) A verification result of (f). (j) Two pieces of original digital puzzle pieces. (k) Two pieces of tampered digital puzzle pieces. (l) The yellow notations indicate locations of (j) and (k). (continued).

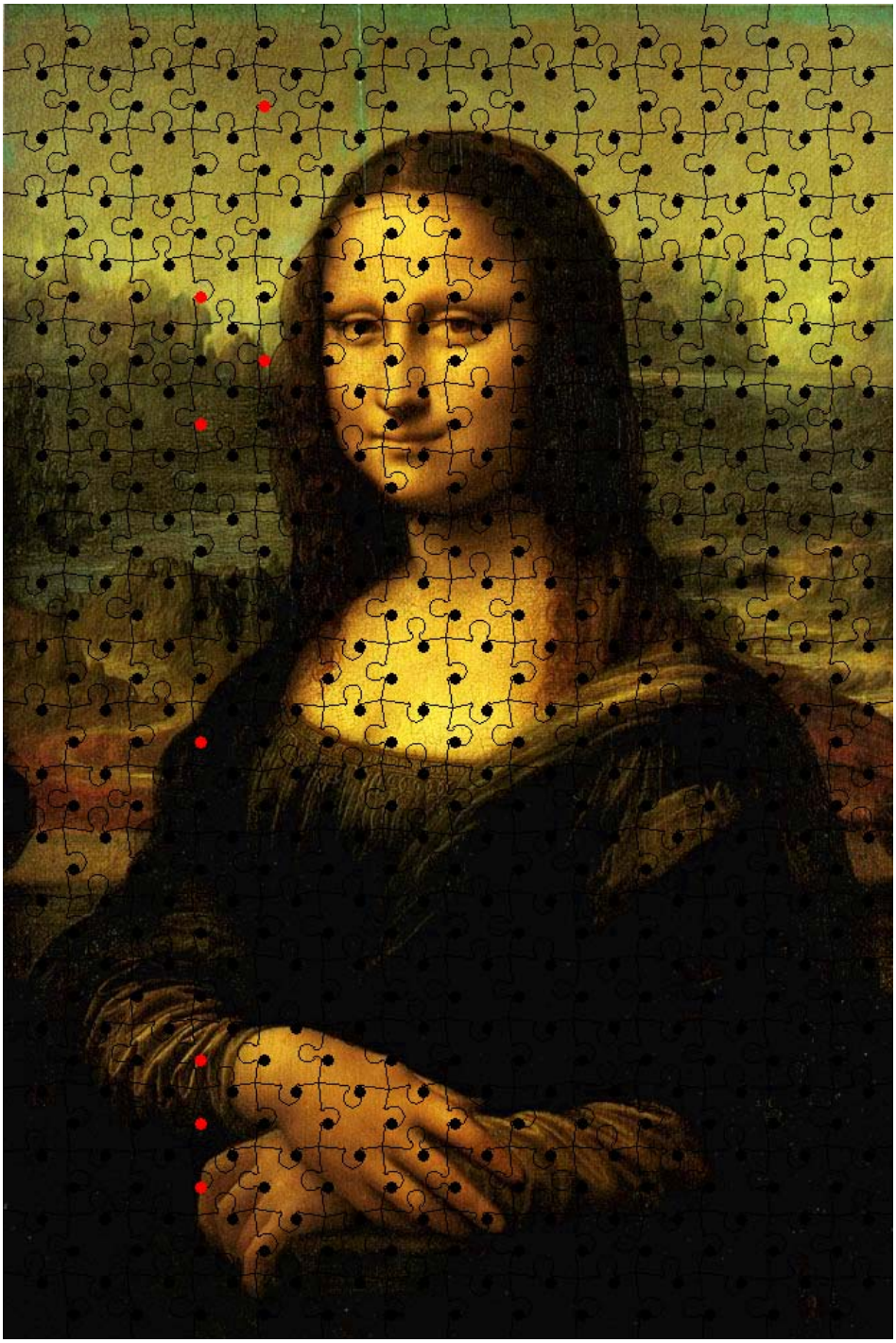


(f)

(g)

(h)

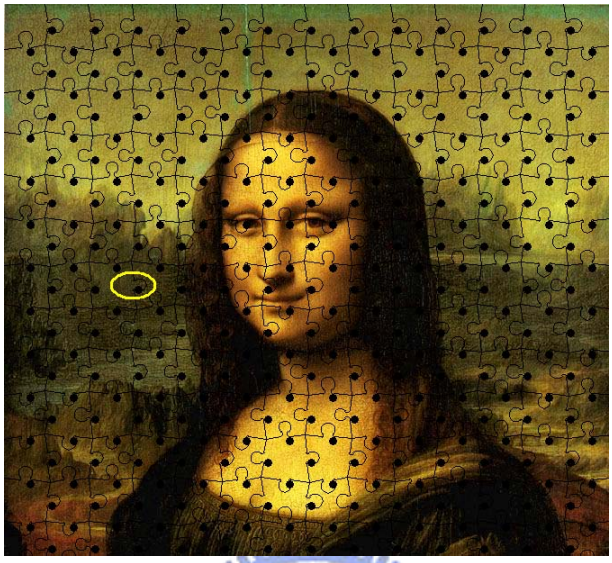
Figure 3.18 Experimental results. (a) An original image. (b) A digital puzzle image with a secret message, a watermark, and authentication signals embedded. (c) A secret message extracted from (b). (d) A watermark extracted from (b). (e) A verification result of (b). (f) A copy of (b) with some tampered puzzle pieces. (g) A secret message extracted from (f). (h) A watermark extracted from (f). (i) A verification result of (f). (j) Two pieces of original digital puzzle pieces. (k) Two pieces of tampered digital puzzle pieces. (l) The yellow notations indicate locations of (j) and (k). (continued).

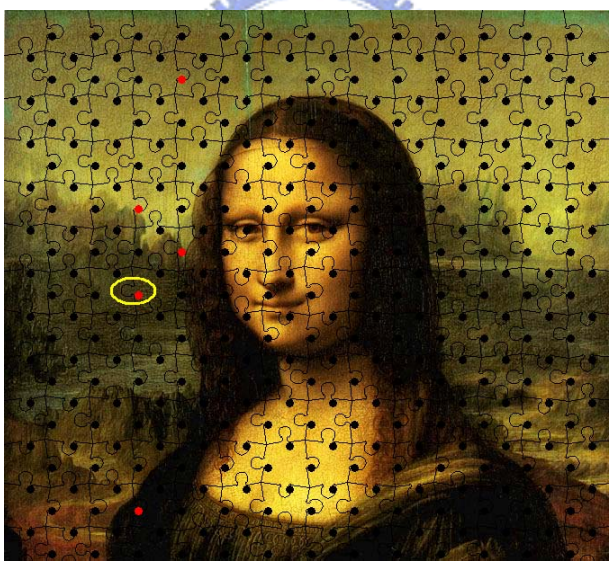


(i)

Figure 3.18 Experimental results. (a) An original image. (b) A digital puzzle image with a secret message, a watermark, and authentication signals embedded. (c) A secret message extracted from (b). (d) A watermark extracted from (b). (e) A verification result of (b). (f) A copy of (b) with some tampered puzzle pieces. (g) A secret message extracted from (f). (h) A watermark extracted from (f). (i) A verification result of (f). (j) Two pieces of original digital puzzle pieces. (k) Two pieces of tampered digital puzzle pieces. (l) The yellow notations indicate locations of (j) and (k). (continued).

(j) (k)





(l)

Figure 3.18 Experimental results. (a) An original image. (b) A digital puzzle image with a secret message, a watermark, and authentication signals embedded. (c) A secret message extracted from (b). (d) A watermark extracted from (b). (e) A verification result of (b). (f) A copy of (b) with some tampered puzzle pieces. (g) A secret message extracted from (f). (h) A watermark extracted from (f). (i) A verification result of (f). (j) Two pieces of original digital puzzle pieces. (k) Two pieces of tampered digital puzzle pieces. (l) The yellow notations indicate locations of (j) and (k). (continued).

Figure 3.19 shows the performing time of embedding information in digital puzzle images and separating them, and the performing time of reconstructing digital puzzle images and extracting the embedded information from them.

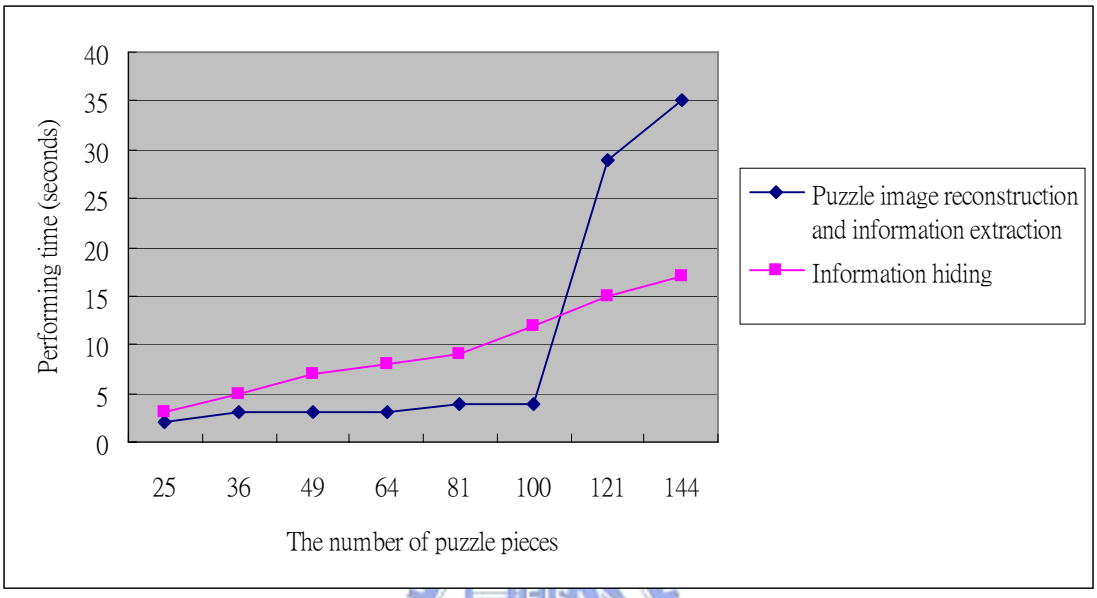


Figure 3.19 The performing time of experimental result