

Signals and Communication Technology

Bin Yan
Yong Xiang
Guang Hua

Improving Image Quality in Visual Cryptography

 Springer

Signals and Communication Technology

Series Editors

Emre Celebi, Department of Computer Science, University of Central Arkansas,
Conway, AR, USA

Jingdong Chen, Northwestern Polytechnical University, Xi'an, China

E. S. Gopi, Department of Electronics and Communication Engineering, National
Institute of Technology, Tiruchirappalli, Tamil Nadu, India

Amy Neustein, Linguistic Technology Systems, Fort Lee, NJ, USA

H. Vincent Poor, Department of Electrical Engineering, Princeton University,
Princeton, NJ, USA

This series is devoted to fundamentals and applications of modern methods of signal processing and cutting-edge communication technologies. The main topics are information and signal theory, acoustical signal processing, image processing and multimedia systems, mobile and wireless communications, and computer and communication networks. Volumes in the series address researchers in academia and industrial R&D departments. The series is application-oriented. The level of presentation of each individual volume, however, depends on the subject and can range from practical to scientific.

“Signals and Communication Technology” is indexed by Scopus.

More information about this series at <http://www.springer.com/series/4748>

Bin Yan · Yong Xiang ·
Guang Hua

Improving Image Quality in Visual Cryptography

Bin Yan
College of Electronics, Communication
and Physics
Shandong University of Science
and Technology
Qingdao, Shandong, China

Yong Xiang
School of Information Technology
Deakin University
Melbourne, VIC, Australia

Guang Hua
School of Electronic Information
Wuhan University
Wuhan, Hubei, China

ISSN 1860-4862 ISSN 1860-4870 (electronic)
Signals and Communication Technology
ISBN 978-981-13-8288-8 ISBN 978-981-13-8289-5 (eBook)
<https://doi.org/10.1007/978-981-13-8289-5>

© Springer Nature Singapore Pte Ltd. 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd. The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

To my beloved family.

—Bin Yan

To my beloved Shan, Angie, and Daniel.

—Yong Xiang

To my beloved ones.

—Guang Hua

Preface

Visual cryptography (VC) is a new form of secret sharing technique, where no computation is needed for decryption. The decryption can be done by stacking the shares or simple OR/XOR operations. This type of crypto-system is especially attractive for computation-limited environment, such as mobile devices or authentication without computation. It has found many applications including but not limited to privacy protection for biometric identification, bar code security, online evaluation and electronic voting, anti-counterfeiting and commodity tracing, etc. This book comprehensively covers the important efforts in improving the visual quality of images in visual cryptography, with a focus on the cases with grayscale images. We not only cover schemes in traditional VC and extended VC for binary secret image, but also cover topics for grayscale secret image and latest development in analysis-by-synthesis approach.

This book distinguishes itself from the existing literature in three ways. First, it not only reviews traditional VC for binary secret image, but also covers recent efforts in improving visual quality for grayscale secret image. Second, not only traditional quality measures are reviewed, but also measures that were not used for measuring perceptual quality of decrypted secret image, such as Radially Averaged Power Spectrum Density (RAPSD) and residual variance, are employed for evaluating and guiding the design of VC algorithms. Third, unlike most visual cryptography books following a mathematical formal style, this book tries to make a balance between engineering intuition and mathematical reasoning. All the targeted problems and corresponding solutions are fully motivated by practical applications and evaluated by experimental tests, while important security issues are presented as mathematical proof. Furthermore, important algorithms are summarized as pseudo-codes, thus enables the readers to reproduce the results in the book. Therefore, this book serves as a tutorial for readers with engineering background as well as for experts in related areas to understand the basics and research frontiers in visual cryptography.

An open source project was built on GitHub to accompany this book, which includes implementation of most algorithms covered in this book in MATLAB and related images and data. Please visit the following URL: <https://github.com/yanbinhit/GrayscaleVisualCryptography>.

The key features of this book include:

1. First book focusing on the topic of perceptual quality in VC.
2. First book comprehensively reviewing and discussing perceptual quality improvement of decrypted grayscale secret images.
3. First book employing perceptual quality measures, such as RAPSD for halftone image, to guide the design and evaluation of grayscale VC.
4. Good balance between engineering intuition and mathematical reasoning.
5. Suitable for both engineers targeting at fast implementation and researchers targeting at catching up the latest development in VC.
6. Matlab code for almost all major algorithms.

Qingdao, China
Melbourne, Australia
Wuhan, China
February 2019

Bin Yan
Yong Xiang
Guang Hua

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) (No. 61272432) and the Shandong Provincial Natural Science Foundation (No. ZR2014JL044). Yong Xiang's work is partially supported the Australian Research Council under Grant LP170100458. Guang Hua's work is partially supported by Hubei Provincial Natural Science Foundation of China (No. 2018CFB225).

Contents

1	Introduction	1
1.1	Brief History of Visual Cryptography	1
1.2	Applications of Visual Cryptography	3
1.2.1	Online Transaction Security	3
1.2.2	Privacy Protection	6
1.2.3	Barcode Security	8
1.2.4	Electronic Voting System Security	9
1.2.5	E-Commerce Security	10
1.3	Classification of Visual Cryptography Algorithms	10
1.4	Remark and Introduction to Chapters	11
	References	12
2	Basic Visual Cryptography Algorithms	15
2.1	Framework of Visual Secret Sharing	15
2.1.1	A Note on Color Convention	15
2.2	Deterministic Visual Cryptography	16
2.2.1	Introduction	16
2.2.2	Definition	18
2.2.3	Constructions	19
2.3	Probabilistic Visual Cryptography	20
2.4	Random Grid Visual Cryptography	23
2.4.1	Generalized Random Grid	27
2.5	Security Issue in Visual Cryptography	28
2.5.1	Strong Security and Weak Security	28
2.6	Summary	32
	References	32
3	Digital Halftoning	35
3.1	Introduction to Digital Halftoning	35
3.2	Bi-level Quantization	36

3.3	Ordered Dithering	37
3.3.1	Clustered Dot Dithering	38
3.3.2	Dispersed Dot Dithering	40
3.4	Error Diffusion and Its Mathematical Model	41
3.4.1	Error Diffusion	41
3.4.2	Mathematical Model	43
3.5	Direct Binary Search	47
3.5.1	Direct Implementation	48
3.5.2	Fast Implementation	49
3.6	Quality Measures for Halftone Image	49
3.6.1	Fidelity Measures	50
3.6.2	Blue Noise and Spectral Characterization	50
3.6.3	Residual Variance	51
3.7	Summary	52
	References	52
4	Improving Visual Quality for Share Images	55
4.1	Binary Visual Cryptography with Meaningful Shares:	
	Extended VC	55
4.1.1	Basic Extended VC	55
4.1.2	User-Friendly Random Grid	58
4.1.3	Pixel Swapping Algorithm	59
4.2	Error-Diffusion Based Scheme	61
4.2.1	SIPs and ABPs	62
4.2.2	Constrained Error Diffusion	64
4.3	Extended VC with a Hidden Watermark	65
4.3.1	Simultaneous Encoding of Secret and Watermark	66
4.3.2	Extension to $N > 2$ Shares	68
4.3.3	Constrained Error Diffusion	69
4.3.4	Experimental Test	69
4.3.5	Attacks on SIPs and ABPs	72
4.4	Summary	72
	References	73
5	Improving Visual Quality for Probabilistic and Random Grid Schemes	75
5.1	(k, n) -Threshold VC for Grayscale Image	75
5.2	Probabilistic VC for Grayscale Secret Image	78
5.2.1	Wang's Algorithm	78
5.2.2	AbS-Based Probabilistic VC	81
5.3	Random Grid VC for Grayscale Secret Image	87
5.3.1	Applying Binary Scheme to Grayscale Image	87
5.3.2	Blue Noise Approach	87
5.3.3	AbS-Based Algorithm for Random Grid Visual Cryptography	91

5.4	Remarks	93
	References	94
6	Improving Visual Quality for Vector Schemes	97
6.1	Vector Visual Cryptography	97
6.1.1	Hou's Block Encoding Algorithm	98
6.1.2	Lee's Block Encoding Algorithm	102
6.1.3	Vector VC for Binary Secret Image	105
6.2	Local Blackness Preserving Visual Cryptography	107
6.2.1	VC Encryption and Local Blackness Preservation	107
6.3	AbS-Based Vector VC	110
6.4	Remarks	115
	References	115
7	Conclusion and Future Works	117
7.1	Summary and Conclusion	117
7.2	Future Works	118
7.2.1	Block Encoding	118
7.2.2	Color VC	118
7.2.3	VC for QR Code	119
7.2.4	Grayscale Secret and Cover Images	119
7.2.5	Tradeoff Between Security and Perceptual Quality	119
7.2.6	Printer Model and HVS Model	119
	References	120

Acronyms and Abbreviations

ABP	Auxiliary Black Pixel
AbS	Analysis-by-Synthesis
AM	Amplitude Modulation
BER	Bit Error Rate
CIP	Cover Information Pixel
DBS	Direct Binary Search
DC	Direct Current
DDF	Directional Distribution Function
FRGVSS	Friendly Random Grid Visual Secret Sharing
HPSNR	Human Peak Signal-to-Noise Ratio
HVS	Human Vision System
IID	Independent and Identically Distributed
MSE	Mean Squared Error
MSSIM	Mean Structure Similarity
NTF	Noise Transfer Function
PDF	Probability Density Function
PSD	Power Spectral Density
PSF	Point Spread Function
PSNR	Peak Signal-to-Noise Ratio
QR	Quick Response
RAPSD	Radially Averaged Power Spectrum Density
RG	Random Grid
RNBED	Random Noise Balanced Error Diffusion
SIP	Secret Information Pixel
SNR	Signal-to-Noise Ratio
STF	Signal Transfer Function
VC	Visual Cryptography
VSS	Visual Secret Sharing

Notations

Matrices/Vectors

α, β, \dots	Coefficients or scalar parameters
a, b, \dots	Scalars
\mathbf{A}, \mathbf{B}	General sets
\mathbb{N}	Set of nonnegative integer numbers, i.e., natural numbers $\{0, 1, 2, \dots\}$
\mathbb{Z}_n	Set $\{0, 1, \dots, n - 1\}$
\mathbb{C}	Complex number
\mathbb{R}	Real number
i, j, k, l, m, n	General index to vector/matrix, limits of index
M, N, L, K	Dimension and length variables, $M, N \in \mathbb{Z}_+$
\mathbf{a}, \mathbf{b}	Boldface lowercase letters, vectors, e.g., $\mathbf{a} = (a_1, \dots, a_N)^T$
\mathbf{A}, \mathbf{B}	Boldface uppercase letters, Matrices, digital images
\mathbf{a}_k	The k -th column of a matrix
$\mathbf{A}_{k,l}$	The (k, l) -th sub-matrix of matrix \mathbf{A} , the (k, l) -th block of image \mathbf{A}
$\{\cdot\}^T$	Transpose operator
$\{\cdot\}^*$	Complex conjugate operator
$\langle \cdot, \cdot \rangle$	Inner product operator
$\ \cdot\ _p$	ℓ_p -norm

Probability

$\Pr(\mathbf{A})$	Probability of event \mathbf{A} that is a subset of sample space
$\mathcal{U}(\mathbf{A})$	Uniform distribution over the set \mathbf{A}
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution, mean μ and variance σ^2

Operators

Script letters $\mathcal{F}(\cdot)$, $\mathcal{G}(\cdot)$	General operators or functions
\otimes	Convolution operator
$\text{round}\{\cdot\}$	Rounding to the nearest integer
$\lfloor \cdot \rfloor$	Rounding to the nearest integer towards $-\infty$
$\mathbf{1}$	An all-one vector with an appropriate length
\odot	Stacking operation
\triangleq	Equal by definition

Signal Processing

$a[i]$	The i -th components of a signal vector \mathbf{a} having finite or infinite length
$a[i,j]$	The (i,j) component of a two-dimensional signal or image \mathbf{A}
$a[\mathbf{n}]$	A sample at position \mathbf{n} , where $[\mathbf{n}] = [i,j]$

Boolean Operation

$a \vee b$	Boolean OR operation between two Boolean variables a and b
\bar{a}	Boolean NOT operation on a Boolean variable a

Chapter 1

Introduction



1.1 Brief History of Visual Cryptography

The history of visual cryptography (VC) or visual secret sharing (VSS) can be traced back to 1987, when the random grid algorithms were introduced in Kafri and Keren's work [26]. But this new type of secret sharing was first formally introduced by Naor and Shamir in their seminal work in [35], where the VC problem with pixel expansion is defined, constructed and analyzed. Naor and Shamir constructed several (n, n) -threshold and (k, n) -threshold schemes, with optimality in terms of contrast and pixel expansion. Later, in 1996, the (k, n) -threshold scheme was extended to more general access structure in Ateniese et al.'s work [3]. The operation in decoding these shares is stacking, which is equivalent to logical 'OR' operation if black pixel is represented by '1'. Another type of 'stacking' operation, XOR (exclusive or), is introduced by Tuyls et al. by considering a system based on light polarization [43]. This type of VC also attracts attentions from many researchers due to its perfect recovery property [15, 45, 50].

Three important problems in VC act like pushing forces to advance this research area, including:

- Reducing pixel expansion;
- Improving quality of the shares and the stacking result of the shares;
- Security problem.

To reduce pixel expansion, there are three basic routes: the probabilistic VC, the random grid VC and the block VC.

1. **Probabilistic VC.** The first route is the probabilistic VC approach. It transformed the Naor and Shamir's pixel-expanded VC to a probabilistic one [23]. Yang formally defined the probabilistic VC problem and provided constructions that don't rely on the deterministic VC [54]. Wang et al. extended the probabilistic VC to encode the grayscale secret image directly [5, 32, 46].
2. **Random grid VC.** In Kafri and Keren's 1987 paper, three $(2, 2)$ -threshold random grid VC algorithms were proposed [26]. These algorithms were extended to gray/color image and (n, n) -threshold by Shyu [40, 41] in 2007 and 2009. Later, the more general $(2, n)$, (n, n) and (k, n) -threshold algorithms were constructed by Chen and Tsao [8, 9] in 2009 and 2011. In 2013, Wu and Sun generalized

the classical random grid VC, where the probability of white pixels on the first share can be larger than the default value $1/2$ that is assumed by previous algorithms [47]. Using this extension, the contrast for $(2, n)$ schemes can be further improved. In 2014, Hou et al.'s work further pushed the quality to the theoretical maximum [21].

3. **Block VC.** Block VC was developed for solving the VC problem for grayscale images. Using block VC, the local structures can be taken into consideration when designing VC algorithm [10, 13, 20, 42]. In 2005, Hou et al. extended the basic Ito algorithm to vector VC [20]. In 2007, Tu and Hou improved block VC by block classification. Their algorithm can ensure that the frequency of basis matrix for black pixel is related to the local feature of this block. So the contrast is guaranteed for each type of block globally. For grayscale secret image, Chen et al. designed vector VC for grayscale image directly [10]. However, each block is processed independently so that the loss of contrast in one block cannot be compensated by other blocks. In 2014, Lee used an implicit histogram equalization in block encoding to extend the histogram of the target image [27]. From 2016 to 2018, Yan et al. developed AbS-based approach, that can improve the quality of the recovered secret image directly [51, 52]. The loss of contrast in one block can be compensated by other adjacent blocks.

The second driving force is improving visual quality in VC. Due to the strict security condition imposed on VC, the quality of the recovered secret image has a lower contrast than the secret image. The loss of contrast is especially severe when one uses probabilistic schemes (probabilistic VC and random grid VC), and when one uses grayscale secret and cover images. To improve visual quality, there are two branches: improving visual quality for binary secret/cover image, and improving visual quality for halftone and grayscale secret image.

1. **For binary secret/cover image.** In 2012, Liu et al. comprehensively investigated the image quality improvement problem [31], mainly for binary secret image. In 2013, Wu et al. proposed algorithms for improving the quality of random grid schemes, using noise balanced error diffusion and void and cluster algorithms, respectively [48, 49]. The noise balanced error diffusion was further improved by Yan et al. in [53].
2. **For halftone and grayscale secret image.** There are two different schemes, sample-based scheme and block-based scheme. For sample-based schemes, in Muecke's thesis and Blundo et al.'s 2000 paper, they constructed VC for grayscale image having more than two levels [5, 32], which is later improved by Wang et al. and transformed to a probabilistic one [46]. These schemes designed basis matrices for each gray level of a sample. The block-based schemes designed basis matrices for blocks with different features, such as the vector VC reviewed in previous section on reducing pixel expansion. For the same pixel expansion, in general, the block-based approach provides better visual quality.

The third driving force is various security problems in VC. Security is a central issue of VC, as with other cryptographic tools. Here we briefly outline some research focuses along this direction.

1. **Weak security.** In 2012, Iwamoto's work shows that if we relax requirement on security, then it is possible to improve contrast and reduce pixel expansion [22, 24]. Similar suggestions are also proposed by Liu et al., when studying block encoding VC [31].
2. **Cheating problem.** Participants may collude to generate fake shares, such that, when stacking the fake shares with the shares from victim participant, a fake secret is reconstructed. To combat such an attack, in 2006, Horng et al. designed two authentication based cheating prevention VC algorithms [12, 19, 30]. In 2010, Prisco and Santis designed $(2, n)$ -threshold and (n, n) -threshold cheating prevention schemes [38]. In 2011, Liu et al. extended the design to general access structure, which also avoided some of the drawbacks in previous cheating prevention VC algorithms.

The research of VC has attracted attentions of researchers from diverse fields, including cryptography, image processing, digital halftoning and multimedia security. Various different new types of VC algorithms are proposed as solutions to apply in different scenarios, such as progressive VC [18, 25, 28], VC with reversing [16, 44], etc. Considering the focus of this book, improving visual quality of recovered image, we direct the readers to other monograph for compressive review of these important topics [15, 29].

1.2 Applications of Visual Cryptography

In recent years, VC has been developed to be one of the building blocks in many solutions to security problems. In this subsection, we review some of the applications of VC in payment security, privacy protection, source tracing and its combination with QR (Quick Response) code.

1.2.1 Online Transaction Security

Nowadays, online transaction is one of the major transaction methods due to the widely use of e-commerce. Such transactions include online banking, online payment on an e-commerce platform, payment by scanning a QR code using a payment App, and payment using an online social network App. For these transactions to be secure, one must ensure that both the hardware and the software are secure so that there are no Trojan horses residing in it. A Trojan horse is a malicious program, in hardware or in software, residing in a client or a sever, trying to compromise the security of a computer system or a communication system.

However, considering the diverse sources of Apps and softwares installed on a computer or cellphone, one cannot guarantee that his/her device is Trojan-free. Using such insecure devices, how to ensure that the transaction is safe? A Trojan horse may

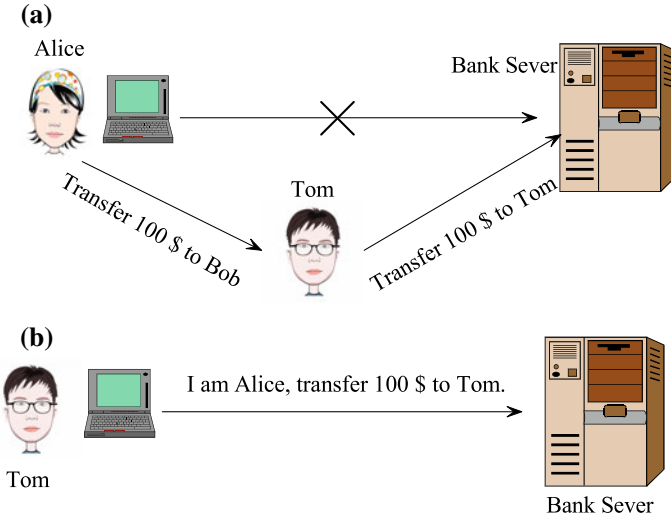


Fig. 1.1 A Trojan horse may modify or forge messages between client (user's computer) and bank sever. **a** A Trojan horse (Tom) modifies a message from user Alice's computer to bank sever. **b** A Trojan horse (Tom) pretends to be Alice and forges a message to bank sever

monitor all messages from bank sever to a user's device and may be able to modify these messages. There are two common attacks on the communication between the bank sever and the client computer, as illustrate in Fig. 1.1. In Fig. 1.1a, an honest user Alice uses her computer to send a message to the bank sever, asking it to transfer 100\$ to Bob. But this message is intercepted by a Trojan horse residing in her computer. Here we denote the Trojan horse as Tom. Trojan horse modified Alice's message to be 'Transfer 100\$ to Tom'. In a second scenario, as shown in Fig. 1.1b, the Trojan horse Tom pretends to be Alice, and sends a message to bank sever, asking it to transfer 100\$ from Alice's account to Tom's account. The first scenario asks for message *authentication* to ensure that the messages between the sever and the client are not modified. The second scenario asks for person *identification*, so that the sever can ensure that it is talking to the right person.

Visual cryptography provides solutions to combating these attacks, thus can ensure that we can make online transactions on an insecure device, such as computer, smart phone, smart card terminal, etc. This protocol was first proposed by Naor and Pinkas [34] and was further developed by Borchert and Reinhardt in [6], and by Pape in [37]. We briefly outline their solutions to authentication and identification in the following steps.

1. Before the transaction starts, the bank sever generates a random grid (random black/white dots) as the first share, which is then printed on a transparency, and distributed to Alice. Alternatively, the random grid image can be sent to Alice in a secure email, and Alice can print it out on a transparency. Such a transparency can be seen in Fig. 1.2a.

2. To transfer 100\$ to Bob, Alice starts the transaction by logging into the online bank webpage (or App), and filling in the receiver's (Bob) bank account and the amount to transfer, as shown in Fig. 1.2b. Alice's bank account is 0001 and the receiver's bank account is 0002. The amount to transfer is indicated by a \$ sign. Then Alice sends this message to the bank sever.
3. After receiving the request from Alice, the bank sever generates a TAN (transaction authentication number) as part of the secret message, and sends back the second share of this secret message, asking Alice to verify the message she sent and identifying if Alice is a person or a Trojan horse. The second share has totally random appearance, as shown in Fig. 1.2c. A Trojan horse, without share 1, cannot obtain the plain text encrypted in this random image. If the share 1 is distributed to Alice in electronic form and is stored in computer disk as a file, then the Trojan horse should have chance to read and use it. So, the key to using this protocol is the printed share, which is later stacked on screen by human user. This blocks the possibility of stealing by the Trojan horse.

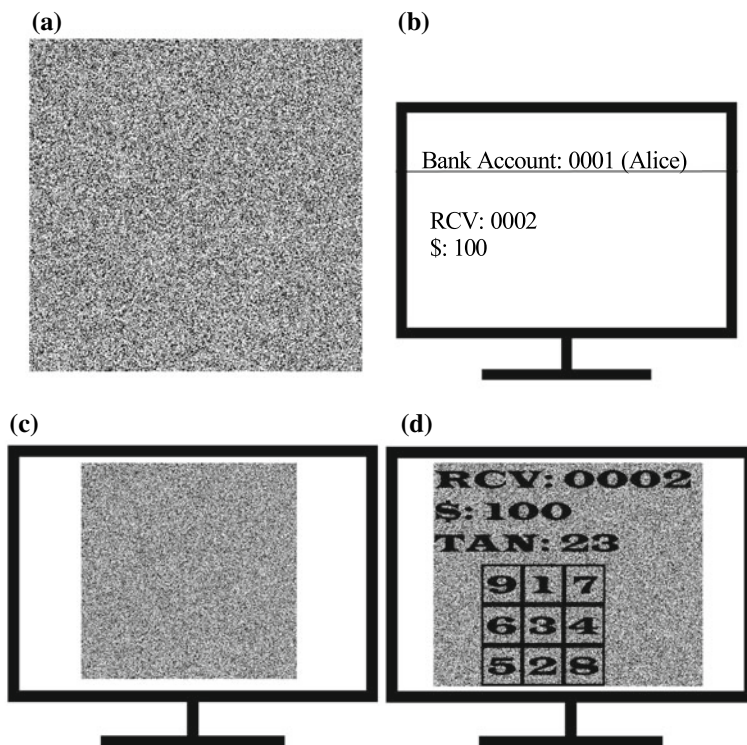


Fig. 1.2 A solution to online transaction on an insecure device using VC. **a** The share transparency Alice obtained from bank. **b** Transaction request from Alice to bank sever. **c** Message from sever to Alice in encrypted form. **d** Decrypted message by stacking the first share owned by Alice (a transparency) with the second share received from the sever

4. After receiving the share 2 in Fig. 1.2c, Alice stacks her share (share 1) with the share on the screen (share 2). The secret verification request will reveal itself, as shown in Fig. 1.2d. This request from the bank asks Alice to confirm the destination of the money transfer, i.e., Bob's bank account, and also the amount to be transferred. It also asks Alice to input a TAN using a soft keyboard that is part of this secret request. The arrangement of the digits in the keyboard is random, so that a Trojan horse cannot figure out the TAN from the positions of the mouse click by Alice. Alice inputs the TAN (here it is 23) by clicking this random keyboard, and the bank will receive the position coordinates of the clicks.
5. After the bank sever receives the position coordinates of the clicks, it can check its local soft keyboard and decode the TAN. By comparing the decoded TAN and the local TAN, the sever is able to judge if it is talking to Alice, not the Trojan. Furthermore, the sever can make sure that the message from Alice is not modified by either the Trojan or an active attacker monitoring the communication link between Alice and itself.

The advantage of this protocol is its ability to combat any Trojan residing in user's device. In this aspect, it is advantageous to short message based authentication, where the short message can also be read by an App or a Trojan horse in the same smart phone. It is also safer than the CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), since an image recognition algorithm can be built into a Trojan horse to help it to recognize the CAPTCHA. Drawbacks of this protocol are the distribution, storage, and alignment of the shares during decoding.

1.2.2 Privacy Protection

Another application of VC is privacy protection. Personal privacy, such as biometric data (fingerprint, face template, finger vein, iris, etc.), medical information and personal financial information, contains sensitive personal information that should be owned only to the owner himself/herself. The issue here is that, these information is also important to access control and disease diagnosis. So, privacy protection addresses the ways to protect personal privacy without hindering legitimate usage of these information.

Visual cryptography provides solutions to privacy protection in several scenarios. This subsection reviews some applications in privacy protection for biometric data and medical data [1, 2, 4, 39, 56].

The biometric feature is unique and cannot be easily changed during life time. So, unlike password, once it is leaked, one cannot come up with another one. But face and fingerprint are used so common today, such as paying by face scanning. For these applications, there are two steps involved: the enrollment step and the authentication step. During the enrollment step, the biometric data is recorded and the features are extracted. These features, usually referred to as template, are stored in a database. During authentication, such as entrance control by fingerprint recognition,

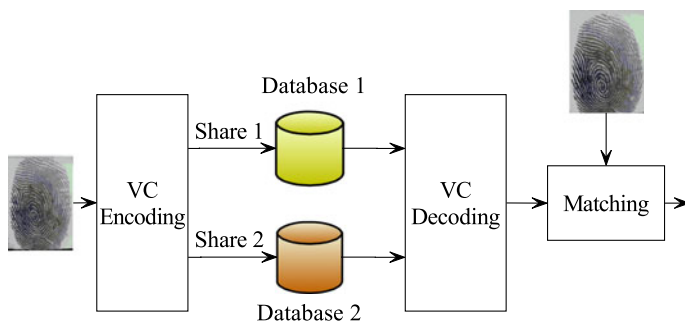


Fig. 1.3 A visual cryptography solution to privacy protection of fingerprint image

a live fingerprint is taken from the visitor and the biometric features are extracted. These live features are then compared with the template stored in the database. The biometric templates are usually stored in a central database owned by a company [39]. Any successful attack to these databases may lead to leakage of user's privacy. Ross and Othman proposed to use VC to split a biometric image into two shares to protect privacy. For example, for fingerprint recognition, this idea is shown in Fig. 1.3.

During enrollment step, the fingerprint image is split into two meaningless shares by VC encoding algorithm. Each share contains totally random patterns and is stored in a separate database. Even if one of the database is attacked, only one share is leaked to the attacker. However, from only one share, it is not possible to gain any information about the fingerprint. During authentication process, such as person verification, the two fingerprint shares are loaded from the two databases. After VC decoding, one may reconstruct the fingerprint enrolled, i.e., the template. Then this template is matched to a live fingerprint. Using such a system, the privacy of the fingerprint is ensured even though one of the database is not secure. In general, for (k, n) -threshold VC, this system may resist successful attacks to up to $k - 1$ databases.

Similar system can also be used in securing medical or health care data [4, 56]. Yang et al. considered the scenario that two or more hospitals need to share their medical images, such as CT (computed tomography), X-ray, etc. But sharing these images between two private systems directly is usually difficult. On the other side, uploading all images to a public system is insecure. Yang et al. proposed to use a random grid type algorithm to generate the two shares of the medical image.

Referring to Fig. 1.4, hospital 1 has a medical equipment. After the medical equipment generates the medical image, it uses a random key K to generate the first share image, the *master share*. Using master share and the medical image, the VC encoding algorithm can generate the second share, which is called the *key share*. The master share is not totally random, but is generated by a pseudo random number generator. Anyone having the key K can generate the master share. The Master share, or the key K , is stored in private database 1 in hospital 1. The key share is uploaded to public system. Thus, leakage of key share from the public system won't lead to lose of private information. When another hospital, hospital 2, needs to access the

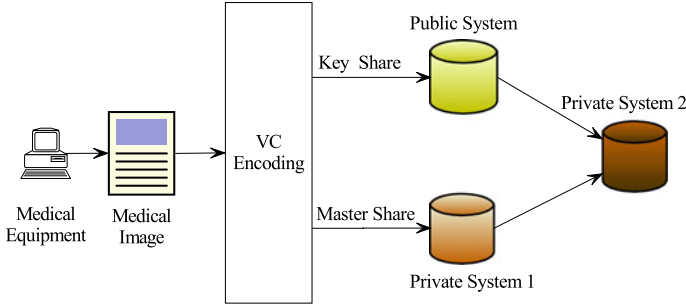


Fig. 1.4 A visual cryptography solution to privacy protection of medical image

medical image, and after authentication, the key share is sent to private system in hospital 2. The key K , which is significantly shorter than the master share, is also sent to the private system 2. Using this key, private system 2 can generate the master share. Finally, private system 2 is able to stack the two shares to recover the medical image. This system has two advantageous features. First, the medical image is secure to information leakage from the public database. Second, only the key that is used to generate the master share needs to be sent, which is fast and efficient.

1.2.3 Barcode Security

QR code is widely used for online and offline payment, especially in China and other east Asian countries. A QR code is a two-dimensional bar code, which can store numbers, URL, or even text. In [17], Fang designed a VC scheme to share a QR code. The QR code image is used as the secret image to VC encoder. The two shares contain random binary pattern, but the function patterns are reserved. By scanning one share, the scanning App cannot find anything and reports a decoding failure. But after stacking the shares, the contents of the QR code can be decoded by a standard QR decoder.

Similar ideas are also used in [11, 14]. A secret QR code is split into several share QR codes. Each share QR code is a legitimate one. So each QR code can be scanned and decoded. The decoded message is irrelevant to the secret message. Once we stack all the shares, we get another QR code image. Scanning this QR image, we may decode the secret message. The advantage of Chow's and Cheng's scheme over Fang's is the cover QR codes, which behave like normal QR code, as illustrated in Fig. 1.5.

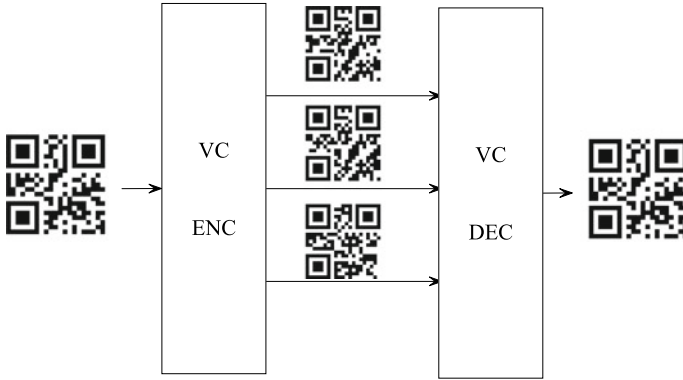


Fig. 1.5 Application of VC to QR code sharing

1.2.4 Electronic Voting System Security

For online voting, voting by participants not living in a neighboring region, attackers may use phishing attack (a fake voting website) to collect personal confidential information. To prevent such an attack, the login password for the legal voting website is split into two shares. One share is sent to the voter's email address, and the other share is stored within the voting website. To log into the voting website, the voter needs to provide his user name and the first share. From the user name, the website will find the second share in its database. After stacking the two shares, the password will reveal itself [36].

To prevent multiple voting, each voter is provided with a unique voting ID card. He/She can execute voting only after his/her identity is verified. Naidu et al. proposed using visual cryptography to design such a mechanism [33]. Each user is identified by his/her fingerprint. During enrollment process, the voter's fingerprint is taken and split into two shares. One share is stored in a voting ID card (a smart card), and the other share is stored in the voting sever. This can ensure that if either the voting ID card is lost or the databased is attack, no information about the user's privacy is leaked. During voting process, the two shares are gathered in a voting station and stacked to produce a fingerprint image. This fingerprint is compared with the live fingerprint taken at the voting station.

In a similar scenario as [33], Chaum used VC to design a mechanism such that the voter will get a receipt after voting [6, 7]. Using this receipt, he/she can ensure that his/her voting is counted. But this receipt should not reveal which candidate the voter has voted.

1.2.5 *E-Commerce Security*

An E-commerce system may have a built-in deposit account and is usually protected by login username-password pair. This database of password is usually the target of attacks since it contains hash value of the login information. To secure this password, Yang et al. proposed a system using VC and optical character recognition (OCR) [55]. Only one share is stored in the database of the website, the other share is hold by the user. To log into the website, the user provides his/her share that will be stacked with the share stored in the database. An OCR system then can be used to recognize the password from the stacking result.

One major concern of online shopping is fake commodity. For example, cellphone is usually assigned with a unique serial number (SN), so that after receiving the cellphone, the buyer may verify the SN printed at the back of the cover from the manufacturer's website. But this SN number is in plain text and can be easily copied to fake cellphone. In [57], Yang et al. designed an anti-counterfeiting mechanism for cellphone. The SN image is split into two shares. One share is printed within the cellphone, and the other share is stored in the manufacturer's database. To verify his cellphone, the user upload his share to verification website and the staking result is sent back to the user's cellphone. This stacking result reveals the true SN number, and can be compared to the one printed on the cellphone cover.

As a secret sharing mechanism, visual cryptography is suitable for applications involving splitting one secret into multiple shares and distributing to multiple participants. Furthermore, decoding involves collaboration between these participants and no key is needed. This is different from encryption, where a key is needed for encryption and decryption.

We end this subsection by noting that in some of these applications, either the secret image or the cover images are grayscale images. For applications involving recognition on the stacking result, the image quality must be good enough. For example, in the application where one needs to do face recognition, fingerprint recognition or OCR on stacking result, the stacking result must retain high fidelity with the original secret image. This motivates our focus in this book: improving image quality in visual cryptography.

1.3 Classification of Visual Cryptography Algorithms

Considering the diverse types of VC algorithms developed over the last thirty years, it is difficult to classify a VC algorithm into just one type. As discussed above, there are three important factors to consider when designing a VC algorithm: pixel expansion, perceptual quality, and security. So the VC algorithms can be classified according to the combination of the above properties.

Based on the way the basis matrices are used and pixel expansion, one can classify the VC algorithms to deterministic, probabilistic or random grid. The deterministic

VC uses all columns of a basis matrix so it has pixel expansion. The probabilistic and random grid approach have no pixel expansion.

Based on the type of ‘stacking’ operation, we may classify the algorithms as based on ‘OR’ operation or ‘XOR’ operation.

Based on the type of the secret image, one may classify the VC algorithms as designed for binary secret image, halftone secret image, grayscale secret image or color secret image.

Based on the whether the shares are meaningful, one may classify a VC algorithm as ordinary VC having meaningless shares, or extended VC having meaningful shares.

1.4 Remark and Introduction to Chapters

In this book, we focus on VC algorithms satisfying strong security and no pixel expansion. The secret image is a grayscale image, whereas the shares are meaningless. Under these conditions, we review important efforts in improving the visual quality of the reconstructed secret image.

This book consists of seven chapters and the topics are arranged as follows.

In Chap. 2, after introducing the basic framework of visual cryptography, we review the three basic VC schemes, the classic deterministic VC, the probabilistic VC and the random grid VC. The security issue including the weak security and replacement attacks will also be briefly reviewed.

In Chap. 3, we introduce the problem of digital halftoning and important halftoning schemes, including simple bi-level quantization, ordered dithering, error diffusion and direct binary search. The quality measures for halftone image are also reviewed. Furthermore, we introduce the residual variance measure that will be used to evaluate the quality of recovered secret image in VC. The purpose of this chapter is to introduce to the reader a self-contained background for digital halftoning. For the VC algorithms we are focusing to, the secret images and/or cover images are grayscale images.

In Chap. 4, we introduce important algorithms in improving visual quality for the share images in extended VC, i.e., VC with meaningful shares. For binary cover image, we introduce three important algorithms, the basic extended VC, user-friendly random grid, and pixel swapping algorithm. For halftone shares, we introduce error diffusion based algorithms, and an improvement of this algorithm by including a hidden watermark, for authentication purpose.

In Chap. 5, we introduce the VC problem for grayscale secret image. Then the efforts in improving size-invariant VC are reviewed. For probabilistic VC, Wang’s size-invariant algorithm and our analysis-by-synthesis algorithms will be introduced. For random grid VC, we introduce the blue noise approach and our analysis-by-synthesis approach.

In Chap. 6, the vector VC approach will be comprehensively reviewed. For halftone secret image, we introduce Hou’s block encoding algorithm, Lee’s block

encoding algorithm, and our local blackness preserving algorithm. From Lee's work, we formally define the vector VC scheme, which is then improved by a vector VC in the analysis-by-synthesis framework.

The Chap. 7 concludes this book with a short summary and discussion of future works.

References

1. Abdul, W., Ali, Z., Ghouzali, S., Alfawaz, B., Muhammad, G., Hossain, M.S.: Biometric security through visual encryption for fog edge computing. *IEEE Access* **5**, 5531–5538 (2017)
2. Abdullah, M.A.M., Dlay, S.S., Woo, W.L., Chambers, J.A.: A framework for iris biometrics protection: a marriage between watermarking and visual cryptography. *IEEE Access* **4**, 10180–10193 (2016)
3. Ateniese, G., Blundo, C., Santis, A.D., Stinson, D.R.: Visual cryptography for general access structures. *Inf. Comput.* **129**(2), 86–106 (1996)
4. Basavegowda, R., Seenappa, S.: Electronic medical report security using visual secret sharing scheme. In: 2013 UKSim 15th International Conference on Computer Modelling and Simulation, pp. 78–83 (2013)
5. Blundo, C., Santis, A.D., Naor, M.: Visual cryptography for grey level images. *Inf. Process. Lett.* **75**(6), 255–259 (2000)
6. Borchert, B., Reinhardt, K.: *Visual Cryptography and Secret Image Sharing*, pp. 329–350. CRC Press (2012). Chapter Applications of Visual Cryptography
7. Chaum, D.: Secret-ballot receipts: true voter-verifiable elections. *IEEE Secur. Priv.* **2**(1), 38–47 (2004)
8. Chen, T.H., Tsao, K.H.: Visual secret sharing by random grids revisited. *Pattern Recognit.* **42**(9), 2203–2217 (2009)
9. Chen, T.H., Tsao, K.H.: Threshold visual secret sharing by random grids. *J. Syst. Softw.* **84**(7), 1197–1208 (2011)
10. Chen, Y.F., Chan, Y.K., Huang, C.C., Tsai, M.H., Chu, Y.P.: A multiple-level visual secret-sharing scheme without image size expansion. *Inf. Sci.* **177**(21), 4696–4710 (2007)
11. Cheng, Y., Fu, Z., Yu, B.: Improved visual secret sharing scheme for QR code applications. *IEEE Trans. Inf. Forensics Secur.* **13**(9), 2393–2403 (2018)
12. Chih-Ming, H., Wen-Guey, T.: Cheating prevention in visual cryptography. *IEEE Trans. Image Process* **16**(1), 36–45 (2007)
13. Chow, Y.W., Susilo, W., Wong, D.S.: Enhancing the perceived visual quality of a size invariant visual cryptography scheme. In: *Proceedings of the 14th International Conference on Information and Communications Security (ICICS 2012)*, Hong Kong, China, pp. 10–21 (2012)
14. Chow, Y.W., Susilo, W., Yang, G., Phillips, J.G., Pranata, I., Barmawi, A.M.: Exploiting the error correction mechanism in QR codes for secret sharing. In: Liu, J.K., Steinfeld, R. (eds.) *Information Security and Privacy*, pp. 409–425. Springer International Publishing, Cham (2016)
15. Cimate, S., Yang, S.N.: *Visual Cryptography and Secret Image Sharing*. CRC Press (2012)
16. Cimate, S., Santis, A.D., Ferrara, A.L., Masucci, B.: Ideal contrast visual cryptography schemes with reversing. *Inf. Process. Lett.* **93**(4), 199–206 (2005)
17. Fang, W.: Offline QR code authorization based on visual cryptography. In: 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 89–92 (2011)
18. Fang, W.P.: Friendly progressive visual secret sharing. *Pattern Recognit.* **41**(4), 1410–1414 (2008)
19. Horng, G., Chen, T., Tsai, D.S.: Cheating in visual cryptography. *Des. Codes Cryptogr.* **38**(2), 219–236 (2006)

20. Hou, Y.C., Tu, S.F.: A visual cryptographic technique for chromatic images using multi-pixel encoding method. *J. Res. Pract. Inf. Technol.* **37**(2), 179–191 (2005)
21. Hou, Y.C., Wei, S.C., Lin, C.Y.: Random-grid-based visual cryptography schemes. *IEEE Trans. Circuits Syst. Video Technol.* **24**(5), 733–744 (2014)
22. Hsu, C.S., Tu, S.F., Hou, Y.C.: An optimization model for visual cryptography schemes with unexpanded shares. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) *Foundations of Intelligent Systems*, pp. 58–67. Springer, Heidelberg (2006)
23. Ito, R., Kuwakado, H., Tanaka, H.: Image size invariant visual cryptography. *IEICE Trans. Fundam.* **E82-A**(10), 2172–2177 (1999)
24. Iwamoto, M.: A weak security notion for visual secret sharing schemes. *IEEE Trans. Inf. Forensics Secur.* **7**(2), 372–382 (2012)
25. Jin, D., Yan, W.Q., Kankanhalli, M.S.: Progressive color visual cryptography. *J. Electron. Imaging* **14**(3), 1–13 (2005)
26. Kafri, O., Keren, E.: Encryption of pictures and shapes by random grids. *Opt. Lett.* **12**(6), 377–379 (1987)
27. Lee, C.C., Chen, H.H., Liu, H.T., Chen, G.W., Tsai, C.S.: A new visual cryptography with multi-level encoding. *J. Vis. Lang. Comput.* **25**(3), 243–250 (2014)
28. Lin, C.H., Lee, Y.S., Chen, T.H.: Friendly progressive random-grid-based visual secret sharing with adaptive contrast. *J. Vis. Commun. Image Represent.* **33**(C), 31–41 (2015)
29. Liu, F., Yan, W.Q.: *Visual Cryptography for Image Processing and Security*. Springer (2014)
30. Liu, F., Wu, C., Lin, X.: Cheating immune visual cryptography scheme. *IET Inf. Secur.* **5**(1), 51–59 (2011)
31. Liu, F., Guo, T., Wu, C., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. *J. Vis. Commun. Image Represent.* **23**(2), 331–342 (2012)
32. Muecke, I.: Greyscale and colour visual cryptography. Master's thesis, Dalhousie University (1999)
33. Naidu, P.S., Kharat, R., Tekade, R., Mendhe, P., Magade, V.: E-voting system using visual cryptography and secure multi-party computation. In: *2016 International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–4 (2016)
34. Naor, M., Pinkas, B.: Visual authentication and identification. In: *Advances in Cryptology CRYPTO 1997. Lecture Notes in Computer Science*, Berlin, pp. 322–336 (1997)
35. Naor, M., Shamir, A.: Visual cryptography. In: *Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT 1994)*, Perugia, Italy, pp. 1–12 (1994)
36. Nisha, S., Madheswari, A.N.: Prevention of phishing attacks in voting system using visual cryptography. In: *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, pp. 1–4 (2016)
37. Pape, S.: *Authentication in Insecure Environments—Using Visual Cryptography and Non-Transferable Credentials in Practice*. Springer Vieweg, Wiesbaden, German (2014)
38. Prisco, R.D., Santis, A.D.: Cheating immune threshold visual secret sharing. *Comput. J.* **53**(9), 1485–1496 (2010)
39. Ross, A., Othman, A.: Visual cryptography for biometric privacy. *IEEE Trans. Inf. Forensics Secur.* **6**(1), 70–81 (2011)
40. Shyu, S.J.: Image encryption by random grids. *Pattern Recognit.* **40**(3), 1014–1031 (2007)
41. Shyu, S.J.: Image encryption by multiple random grids. *Pattern Recognit.* **42**(7), 1582–1596 (2009)
42. Tu, S.F., Hou, Y.C.: Design of visual cryptographic methods with smooth looking decoded images of invariant size for grey-level images. *Imaging Sci. J.* **55**(2), 90–101 (2007)
43. Tuyls, P., Hollmann, H.D.L., Lint, J.H.V., Tolhuizen, L.M.G.M.: XOR-based visual cryptography schemes. *Des. Codes Cryptogr.* **37**, 169–186 (2005)
44. Viet, D.Q., Kurosawa, K.: *Topics in Cryptology? CT-RSA 2004*, vol. 2964, pp. 21–37. Springer (2004). Chapter Almost Ideal Contrast Visual Cryptography with Reversing
45. Wang, D.S., Dong, L.: *Visual Cryptography and Secret Image Sharing*, pp. 155–183. CRC Press (2012). Chapter XOR-Based Visual Cryptography

46. Wang, D.S., Yi, F., Li, X.B.: Probabilistic visual secret sharing schemes for grey-scale images and color images. *Inf. Sci.* **181**(11), 2189–2208 (2011)
47. Wu, X.T., Sun, W.: Generalized random grid and its applications in visual cryptography. *IEEE Trans. Inf. Forensics Secur.* **8**(9), 1541–1553 (2013)
48. Wu, X.T., Sun, W.: Improving the visual quality of random grid-based visual secret sharing. *Signal Process.* **93**(5), 977–995 (2013)
49. Wu, X.T., Liu, T., Sun, W.: Improving the visual quality of random grid-based visual secret sharing via error diffusion. *J. Vis. Commun. Image Represent.* **24**(5), 552–566 (2013)
50. Wu, X., Sun, W.: Extended capabilities for xor-based visual cryptography. *IEEE Trans. Inf. Forensic Secur.* **9**(10), 1592–1605 (2014)
51. Yan, B., Chen, N., Yang, H.M., Hao, J.J.: Local blackness preserving visual cryptography for grayscale secret images. *J. Inf. Hiding Multimed. Signal Process.* **9**, 370–382 (2018)
52. Yan, B., Xiang, Y., Hua, G.: Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (AbS) approach. *IEEE Trans. Image Process.* **28**(2), 896–911 (2019)
53. Yan, X., Wang, S., El-Latif, A.A.A., Niu, X.: Random grids-based visual secret sharing with improved visual quality via error diffusion. *Multimed. Tools Appl.* **74**(21), 9279–9296 (2014)
54. Yang, C.N.: New visual secret sharing schemes using probabilistic method. *Pattern Recognit. Lett.* **25**(4), 481–494 (2004)
55. Yang, D., Doh, I., Chae, K.: Enhanced password processing scheme based on visual cryptography and OCR. In: 2017 International Conference on Information Networking (ICOIN), pp. 254–258 (2017)
56. Yang, D., Doh, I., Chae, K.: Secure medical image-sharing mechanism based on visual cryptography in EHR system. In: 2018 20th International Conference on Advanced Communication Technology (ICACT), pp. 463–467 (2018)
57. Yang, X.C., Li, J.P., Mou, J.P.: An anti-counterfeiting method based on VCS for mobile phone's identification. In: 2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP), pp. 169–172 (2012)

Chapter 2

Basic Visual Cryptography Algorithms



2.1 Framework of Visual Secret Sharing

In a (k, n) -threshold visual cryptography (visual secret sharing) problem [13], a *secret image* s is shared between n parties, called *participants*. Each participant obtains one *share image*. If less than k shares are obtained, it is not possible to infer s from them. But, if k or more than k shares are obtained, then the secret s can be revealed by stacking these shares, producing different average brightnesses for the white secret pixels and black secret pixels. For visual cryptography, the shares are usually printed on transparencies, and decoding corresponds to stacking the shares. The stacking result is called the *target image*.

The above threshold schemes can be extended to more general access structure [1]. There are two sets of participants, the forbidden set Γ_{Forb} , and the qualified set Γ_{Qual} . For a group of participants P , if $P \in \Gamma_{\text{Forb}}$, then it should be impossible to infer s from the shares owned by these participants. If $P \in \Gamma_{\text{Qual}}$, then by stacking all the shares owned by these participants, the content of the secret should be recovered with sufficient contrast.

For example, for a $(3, 3)$ -threshold scheme, we have

$$\Gamma_{\text{Qual}} = \{\{1, 2, 3\}\}, \quad (2.1)$$

$$\Gamma_{\text{Forb}} = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}. \quad (2.2)$$

2.1.1 A Note on Color Convention

For monochrome, there are two conventional ways of numerical representation of colors. For the first approach, the axis is lightness, where larger value corresponds to brighter color. Such a convention is widely used in digital image processing and digital display. For the second approach, the axis represents the amount of absorption

of light, where larger value corresponds to darker color. Such a convention is adopted in printing, visual cryptography and QR (Quick Response) code.

Visual cryptography, in its strict sense, needs to print its shares on transparencies. So, larger value should correspond to higher darkness. In this book, we adopt this convention. If an image is normalized to the range $[0, 1]$, then '0' corresponds to the brightest color (White) and '1' (for normalized image) or '255' (for 8-bit representation) corresponds to the darkest color (Black). In summary, we use the following numerical representation of pixels:

$$\blacksquare \leftrightarrow 1, \quad (2.3)$$

$$\square \leftrightarrow 0. \quad (2.4)$$

2.2 Deterministic Visual Cryptography

Deterministic VC is the type of VC proposed by Naor and Shamir [13]. The prefix 'deterministic' is used here to distinguish it from the probabilistic schemes. Please note that deterministic doesn't mean that every step of the algorithm is deterministic. For deterministic VC, each secret pixel is represented by a block on share images. Usually, the shape of the block is a square, in order to preserve the aspect ratio of objects in the secret image onto the target image.

2.2.1 Introduction

A simple example is shown in Fig. 2.1. When sharing a white pixel $s = 0$, one randomly chooses one row from the second and the third rows in Fig. 2.1, and distributes the second column to Share 1 and the third column to Share 2. The final stacking result is a 2×2 block with two black pixels, as shown in the last column. Similarly, while sharing a black pixel $s = 1$, one randomly chooses one row from the fourth and the fifth rows of the table, and distributes the second column to Share 1 and the third column to Share 2. The final stacking result is a 2×2 block with four black pixels, as shown in the last column. So, one secret pixel is expanded to a block having four pixels, or equivalently, one secret pixel is represented by four sub-pixels.

If we look at one share, no matter whether a white pixel is shared or a black pixel is shared, we always see roughly equal number of the following two types of blocks on each share:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (2.5)$$

Thus, by checking only one share, it is impossible to tell if a black pixel is shared or a white pixel is shared. This is the *security* consideration in visual cryptography. But

if we check the target image, the block corresponding to white secret pixel has two black pixels, while the block corresponding to the black secret pixel has four black pixels. Thus, one can distinguish between the black region and the white region of the secret image by checking the target image. This is the *contrast* consideration in visual cryptography.

When designing a deterministic VC algorithm, the blocks corresponding to a secret pixel are usually vectorized and organized into a matrix. For example, the two blocks in (2.5) are represented as a matrix

$$\mathbf{C} = \begin{bmatrix} \text{vec}(\mathbf{A}_1)^T \\ \text{vec}(\mathbf{A}_2)^T \end{bmatrix}, \quad (2.6)$$

where each block \mathbf{A}_i , $i = 1, 2$, is stretched to a column vector by function $\text{vec}(\mathbf{A}_i)$, transposed, and filled into one row of \mathbf{C} . So, each row of \mathbf{C} should be distributed to one share and then reshaped to appropriate shape. For the design of VC algorithm, the reshaping is not a central issue. So, it is usually omitted when defining the VC.

For the example in Fig. 2.1, we have the following matrices:

$$\begin{aligned} \mathbf{C}_0 &= \{\mathbf{C}_1^0, \mathbf{C}_2^0\} = \left\{ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \right\}, \\ \mathbf{C}_1 &= \{\mathbf{C}_1^1, \mathbf{C}_2^1\} = \left\{ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \right\}. \end{aligned} \quad (2.7)$$















Secret pixel	Share 1	Share 2	Stacked image (Target image)
			
			
			
			

Fig. 2.1 A demonstration of (2, 2)-threshold deterministic VC. The pixel expansion is 4

When sharing a secret pixel $s \in \{0, 1\}$, we randomly (with equal probabilities) choose one matrix from the set \mathbf{C}_s and distribute the first row of that matrix to Share 1, and distribute the second row to Share 2. How the shares reshape the vector to a block is not a central issue here. If we check only one row of the matrices, say first row, then we have

$$\hat{\mathbf{C}}_0 = \{[0 \ 1 \ 1 \ 0], [1 \ 0 \ 0 \ 1]\}, \quad (2.8)$$

$$\hat{\mathbf{C}}_1 = \{[0 \ 1 \ 1 \ 0], [1 \ 0 \ 0 \ 1]\}. \quad (2.9)$$

These two sets are equivalent in the sense that, (1) they have the same matrices, and (2) the frequencies of these matrices are equal, since we choose matrices in each set with equal probabilities. So by checking only one share, one cannot tell if $s = 0$ or $s = 1$. After stacking two rows of the matrices in set \mathbf{C}_0 , the number of black pixels, which is also the Hamming weight of the resulting vector, is 2. But for \mathbf{C}_1 , it is 4. So from the stacking result one can tell whether $s = 0$ or $s = 1$.

2.2.2 Definition

Let $\mathcal{H}(\mathbf{v})$ be the Hamming weight of a Boolean vector \mathbf{v} , i.e., the number of 1s in \mathbf{v} . Based on the above explanation, the (k, n) -threshold VC can be formally defined by the following definition [13]:

Definition 2.1 (k, n) -threshold VC: A (k, n) -threshold VC consists of two collections of $n \times m$ matrices \mathbf{C}_0 and \mathbf{C}_1 . To share a white (black resp.) pixel, one randomly chooses one matrix from \mathbf{C}_0 (\mathbf{C}_1 resp.). Each row of the chosen matrix is distributed to one share. The following two conditions should be satisfied:

1. **Contrast condition:** For any $\mathbf{C} \in \mathbf{C}_0$, let $\mathbf{C} = [\mathbf{c}_1^T, \dots, \mathbf{c}_n^T]^T$, where \mathbf{c}_i is the i -th row of matrix \mathbf{C} . Then, the stacking of any k rows or more satisfies $\mathcal{H}(\bigvee_{j=1}^k \mathbf{c}_{i_j}) \leq d - \alpha$, where i_1, \dots, i_k are k randomly chosen row indices. For any $\mathbf{C} \in \mathbf{C}_1$, $\mathcal{H}(\bigvee_{j=1}^k \mathbf{c}_{i_j}) \geq d$.
2. **Security condition:** For any $q < k$ randomly chosen shares $\{i_1, \dots, i_q\} \in \{1, 2, \dots, n\}$, the two collections of matrices $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$, obtained by restricting each matrix to the rows i_1, \dots, i_q , should be indistinguishable. Namely, $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ should contain the same matrices with the same frequencies.

The parameter m is pixel expansion rate and parameter α gives the minimum difference in Hamming weight between the stacking result for a black secret pixel and the stacking result for a white secret pixel.

2.2.3 Constructions

The construction of the two sets of matrices \mathbf{C}_0 and \mathbf{C}_1 can be based on two basis matrices \mathbf{B}_0 and \mathbf{B}_1 . After designing \mathbf{B}_0 (\mathbf{B}_1 resp.), \mathbf{C}_0 may include all column permutations of \mathbf{B}_0 (\mathbf{B}_1 resp.). For a matrix $\mathbf{B}_0 \in \mathbb{Z}_2^{n \times m}$, there are $m!$ matrices in corresponding \mathbf{C}_0 .

One optimal construction for (n, n) -threshold scheme was proposed by Naor and Shamir [13]. It starts with a set having n components, which is called a ground set. Let the set be $W = \{e_1, \dots, e_n\}$, which has 2^n subsets. Among these subsets, let $U_1, \dots, U_{2^{n-1}}$ be the 2^{n-1} subsets with even cardinality, and let $V_1, \dots, V_{2^{n-1}}$ be the 2^{n-1} subsets with odd cardinality. Then we obtain two basis matrices \mathbf{B}_0 and \mathbf{B}_1 as follows:

$$\begin{aligned} B_0[i, j] &= \begin{cases} 1, & \text{if } e_i \in U_j, \\ 0, & \text{otherwise.} \end{cases} \\ B_1[i, j] &= \begin{cases} 1, & \text{if } e_i \in V_j, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (2.10)$$

Then the set of matrices \mathbf{C}_0 (\mathbf{C}_1 resp.) is obtained by all possible column permutations of \mathbf{B}_0 (\mathbf{B}_1 resp.).

For example, Let $n = 3$ and $m = 4$, then we use the ground set $W = \{e_1, e_2, e_3\}$. So, we have the following subsets:

$$\begin{aligned} U_1 &= \{e_1, e_2\}, & U_2 &= \{e_1, e_3\}, & U_3 &= \{e_2, e_3\}, & U_4 &= \emptyset, \\ V_1 &= \{e_1\}, & V_2 &= \{e_2\}, & V_3 &= \{e_3\}, & V_4 &= \{e_1, e_2, e_3\}. \end{aligned}$$

From (2.10), we get two basis matrices:

$$\mathbf{B}_0 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{B}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

The set of matrices \mathbf{C}_0 (\mathbf{C}_1 resp.) contains $2^{n-1}! = 4! = 24$ matrices that are all possible column permutations of the matrix \mathbf{B}_0 (\mathbf{B}_1 resp.).

Naor and Shamir proved that this scheme is optimal, in that it has the minimum pixel expansion m and maximum contrast α for a given n :

Theorem 2.1 ([13]) *For any (n, n) -threshold deterministic VC, the contrast is upper bounded by $1/2^{n-1}$, and the pixel expansion m is lower bounded by 2^{n-1} .*

Proof for security and contrast condition can be found in [13], which is thus omitted here. We also direct the reader to [13] for more general constructions such as (k, n) -threshold scheme.

An experimental result using the scheme in Fig. 2.1 is shown in Fig. 2.2.

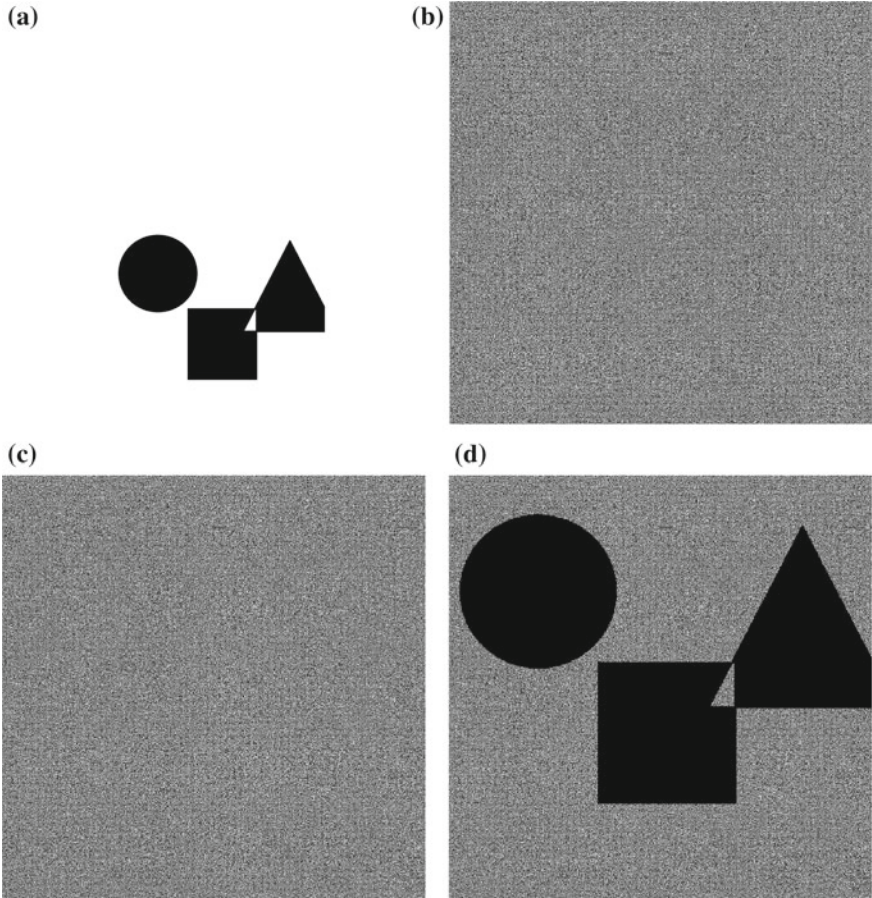


Fig. 2.2 Experimental result for (2, 2)-threshold VC constructed by Naor and Shamir. **a** Secret image. **b** Share 1. **c** Share 2. **d** Target image. Please note that the sizes of the shares and target images are four times of the size of the secret image

2.3 Probabilistic Visual Cryptography

The probabilistic VC is build upon the deterministic VC and its original purpose is to produce size-invariant shares [9, 18], i.e., shares having the same size as the secret image.

Ito et al. transformed the deterministic VC into probabilistic counterpart. The sets of matrices \mathbf{C}_0 and \mathbf{C}_1 are the same as those constructed for deterministic VC. But Ito et al. used it in a probabilistic way. This algorithm is summarized in Algorithm 1.

If the two sets of matrices \mathbf{C}_0 and \mathbf{C}_1 are constructed by permuting columns of two basis matrices \mathbf{B}_0 and \mathbf{B}_1 , then one can safely choose $\mathbf{C}_0 = \mathbf{B}_0$ and $\mathbf{C}_1 = \mathbf{B}_1$ in Algorithm 1.

Algorithm 1 (k, n) -threshold probabilistic VC constructed by Ito et al. [9].

Require:

Binary secret image: $x[n] \in \{0, 1\}$.
 Two sets of matrices C_0 and C_1 .

Ensure:

n binary share images: $s_1[n], \dots, s_n[n]$.

- 1: Randomly choose two matrices $C_0 \in C_0$ and $C_1 \in C_1$.
 - 2: **for** each pixel n **do**
 - 3: **if** $x[n] = 0$ **then**
 - 4: $c \leftarrow$ Randomly select one column of C_0 .
 - 5: **else**
 - 6: $c \leftarrow$ Randomly select one column of C_1 .
 - 7: **end if**
 - 8: $s_i[n] \leftarrow c_i$, where $i = 1, \dots, n$.
 - 9: **end for**
-

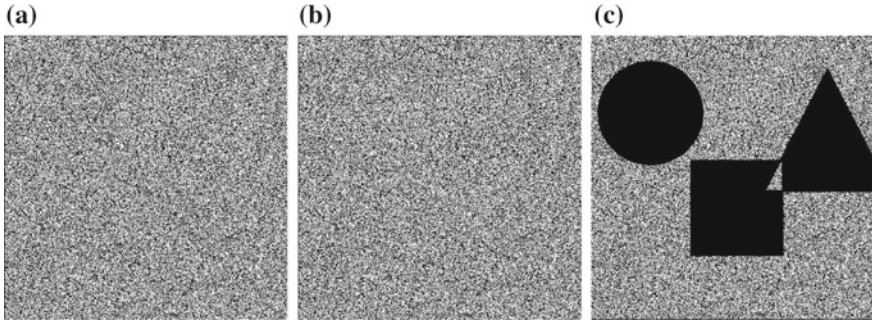


Fig. 2.3 Experimental result for $(2, 2)$ -threshold probabilistic VC constructed by Ito et al. **a** Share 1. **b** Share 2. **c** Target image

Using the basis matrices

$$B_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

the experimental result is shown in Fig. 2.3. Compared with the deterministic result, we see that the average contrasts on target images are indistinguishable. But, for deterministic VC, the black and white pixels are more evenly distributed in region corresponding to white secret pixels. In contrast, we see many black clusters for probabilistic VC. This is due to the probabilistic use of the basis matrices.

Yang formally defined the (k, n) -threshold probabilistic VC scheme and constructed various solutions. We briefly review this definition and a (n, n) -threshold scheme.

The definition of (k, n) -threshold probabilistic VC can be explained as follows. The core to the probabilistic VC is two collection of vectors, C_0 and C_1 . The elements of each set are n -dimensional vectors. Suppose that the set C_0 contains n_0 vectors

and the set \mathbf{C}_1 contains n_1 vectors, where n_0 may not be equal to n_1 . For example, if we take $n = 3$, we may have

$$\mathbf{C}_0 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\}, \quad \mathbf{C}_1 = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}. \quad (2.11)$$

When sharing a secret pixel $s \in \{0, 1\}$, a vector is randomly chosen from \mathbf{C}_s , and the rows of this vector are randomly permuted. Then the i -th element is assigned to share i . For each vector $\mathbf{c} = [c_1, \dots, c_n]^T \in \mathbf{C}_i$, we can calculate the stacking result $\mathcal{L}(\mathbf{c}) = c_1 \vee \dots, \vee c_n$. So, we have two sets \mathbf{L}_0 and \mathbf{L}_1 derived from \mathbf{C}_0 and \mathbf{C}_1 , respectively, where

$$\mathbf{L}_i = \{\mathcal{L}(\mathbf{c}), \forall \mathbf{c} \in \mathbf{C}_i\}, i = 0, 1. \quad (2.12)$$

Each set \mathbf{L}_i contains stacking results for all vectors in set \mathbf{C}_i . For the example in (2.11), we have $\mathbf{L}_0 = \{0, 1, 1, 1\}$ and $\mathbf{L}_1 = \{1, 1, 1, 1\}$.¹

If the following two conditions are satisfied, then this scheme is called a (n, n) -threshold probabilistic VC.

1. **Contrast condition:** Let p_b be the proportion of white pixels in the set \mathbf{L}_b . Then the two sets \mathbf{C}_0 and \mathbf{C}_1 must satisfy $p_0 \geq p_{\text{TH}}$ and $p_1 \leq p_{\text{TH}} - \alpha$, for some threshold $p_{\text{TH}} > 0$ and contrast $\alpha > 0$.
2. **Security condition:** For $q < k$ and any subset $\{i_1, \dots, i_q\} \subset \{1, 2, \dots, n\}$, the probabilities p_0 and p_1 should be the same. More specifically, for each vector $\mathbf{c} \in \mathbf{C}_i$, if we retain only the components whose indices are in $\{i_1, \dots, i_q\}$, we get two new sets $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$, and two corresponding sets $\hat{\mathbf{L}}_0$ and $\hat{\mathbf{L}}_1$. The p_0 and p_1 obtained from $\hat{\mathbf{L}}_0$ and $\hat{\mathbf{L}}_1$ should be equal.

For the example in (2.11), one can verify that for $k = 3$, $p_0 = 1/4$ and $p_1 = 0$. Thus we can set $p_{\text{TH}} = 1/4$ and $\alpha = 1/4$.

Yang constructed $(2, n)$, (n, n) and (k, n) -threshold schemes for probabilistic VC. Here we outline his (n, n) -threshold scheme since it is relevant to our AbS scheme in later chapters.

Let $\mu_{i,j}$ be the set of vectors in \mathbf{C}_j whose Hamming weight is i , i.e.,

$$\mu_{i,j} = \{\mathbf{c} \in \mathbf{C}_j, \text{ such that } \mathcal{H}(\mathbf{c}) = i\}, \quad (2.13)$$

where $\mathcal{H}(\mathbf{c})$ returns Hamming weight of vector \mathbf{c} . Then, the two sets \mathbf{C}_0 and \mathbf{C}_1 can be constructed as:

¹Please note that calling \mathbf{L}_i a set is not mathematically vigorous, since the elements in \mathbf{L}_i are allowed to be the same. We stick to this notation since it is widely used in the literature of probabilistic VC.

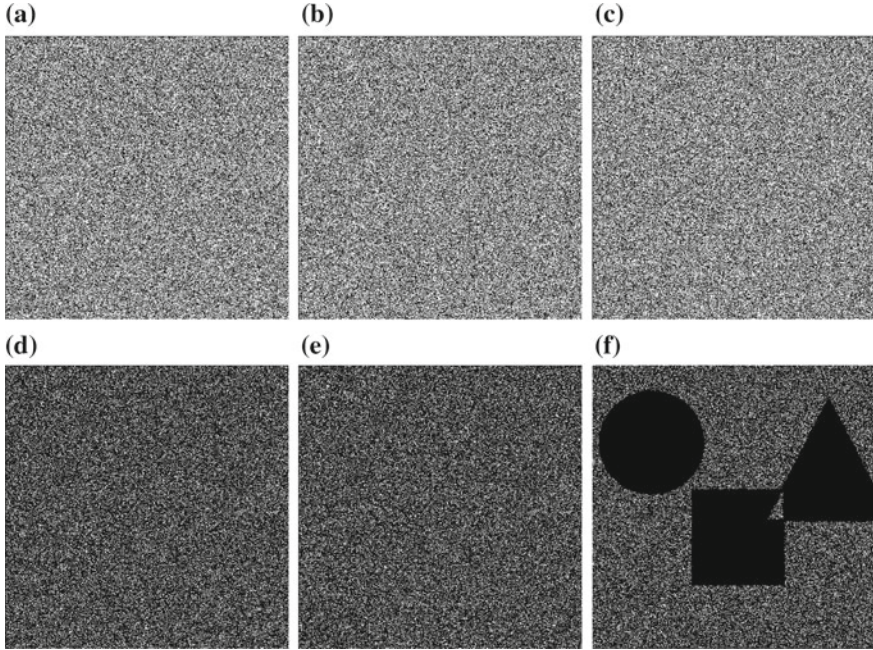


Fig. 2.4 Experimental result for $(3, 3)$ -threshold probabilistic VC constructed by Yang [18]. **a** Share 1. **b** Share 2. **c** Share 3. **d** Stacking result of share 1 and share 2. **e** Stacking result of share 1 and share 3. **f** Stacking result of three shares

$$C_0 = \{\mu_{i0}, i = \text{even, and } i \in \{0, \dots, n\}\}, \quad (2.14)$$

$$C_1 = \{\mu_{i1}, i = \text{odd, and } i \in \{0, \dots, n\}\}. \quad (2.15)$$

This construction is shown to have threshold $p_{TH} = 1/2^{n-1}$ and contrast $\alpha = 1/2^{n-1}$. For $n = 3$, the design result is shown in (2.11). Using this construction, the experimental result is shown in Fig. 2.4.

2.4 Random Grid Visual Cryptography

The random grid (RG) based VC doesn't rely on the basis matrices. A random grid is a collection of black and white pixels located at regularly spaced points, usually with equal probability. Using RG VC, the secret image is encoded as relationships between two or more random grids.

The three $(2, 2)$ -threshold RG VC algorithms proposed by Kafri and Keren [11] were extended to gray/color image and (n, n) -threshold by Shyu [14, 15]. Later, the more general $(2, n)$, (n, n) and (k, n) -threshold algorithms were constructed by Chen and Tsao [2, 3]. Wu and Sun generalized the classical random grid VC, where the

probability of white pixels on the first share can be larger than the default value $1/2$ that is assumed by previous algorithms [17]. This modification can help to increase the contrast for $(2, n)$ schemes. Recent effort by Hou et al. further push the quality to the theoretical maximum [5].

The basic idea of (n, n) -threshold RG VC is to repeatedly split the secret image and then the shares using $(2, 2)$ -threshold RG VC [2]. After each iteration, a new share is generated. So after $n - 1$ steps, n shares are generated. Here we summarize the algorithm proposed by Chen and Tsao [2], using the notation in this book. The basic $(2, 2)$ -threshold RG VC is shown in Algorithm 2. The first share is a randomly generated binary image, i.e., a random grid. Pixels are independent and identically distributed (IID) with a uniform distribution over the set $\{0, 1\}$. The second share encodes the secret image \mathbf{x} as relationship between itself and the first share. If a black pixel needs to be shared, then the pixel $s_2[\mathbf{n}]$ on the second share is complementary to the corresponding pixel $s_1[\mathbf{n}]$ on the first share, i.e., $s_2[\mathbf{n}] = 1 - s_1[\mathbf{n}]$. Otherwise, the pixel $s_2[\mathbf{n}]$ on the second share is the same as the corresponding pixel $s_1[\mathbf{n}]$ on the first share, i.e., $s_2[\mathbf{n}] = s_1[\mathbf{n}]$. A simple demo is shown in Fig. 2.5.

Algorithm 2 ($s_1[\mathbf{n}], s_2[\mathbf{n}] \leftarrow \text{RandGrid22}(x[\mathbf{n}])$): Basic $(2, 2)$ -threshold random grid VC (Algorithm 1 in [11]).

Require:

Binary secret image: $x[\mathbf{n}] \in \{0, 1\}$.

Ensure:

Two binary share images: $s_1[\mathbf{n}], s_2[\mathbf{n}]$.

```

1: for each pixel  $\mathbf{n}$  do
2:    $s_1[\mathbf{n}] \stackrel{\text{IID}}{\sim} \mathcal{U}\{0, 1\}$ .
3:   if  $x[\mathbf{n}] = 0$  then
4:      $s_2[\mathbf{n}] \leftarrow s_1[\mathbf{n}]$ .
5:   else
6:      $s_2[\mathbf{n}] \leftarrow 1 - s_1[\mathbf{n}]$ .
7:   end if
8: end for
```

Based on the $(2, 2)$ -threshold random grid VC, the (n, n) -threshold random grid VC can be constructed, as shown in Algorithm 3. In the first step, two shares are generated using a $(2, 2)$ -threshold random grid VC. Then, the second share is treated as a new secret image and is used to generate another two shares using $(2, 2)$ -threshold VC. So, this algorithm iteratively split one share into more shares, till the required number of shares are obtained. An experimental result for $(3, 3)$ -threshold random grid is shown in Fig. 2.6.

The $(2, n)$ -threshold random grid constructed by Chen and Tsao can be summarized in Algorithm 4 [2]. In brief, to share a white pixel, the shares copy the random grid (the first share). To share a black pixel, each share randomly generates its own random grid. Thus when stacking, the average blackness for the region Ω_0 is $1/2$, and the average blackness for the region Ω_1 will increase with the number of shares, thus producing contrast between region Ω_0 and Ω_1 .

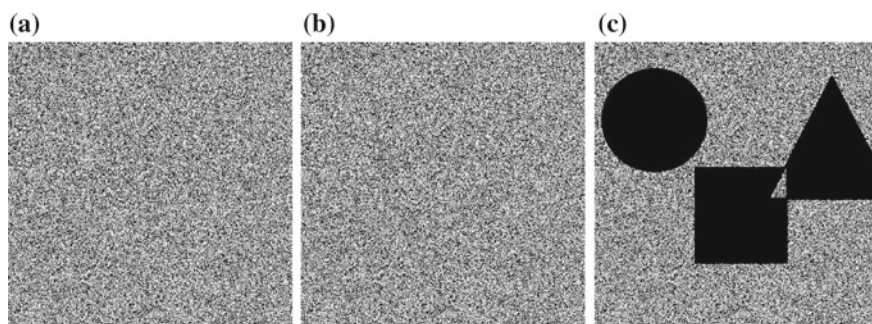


Fig. 2.5 Experimental result for $(2, 2)$ -threshold random grid. **a** Share 1. **b** Share 2. **c** Target image

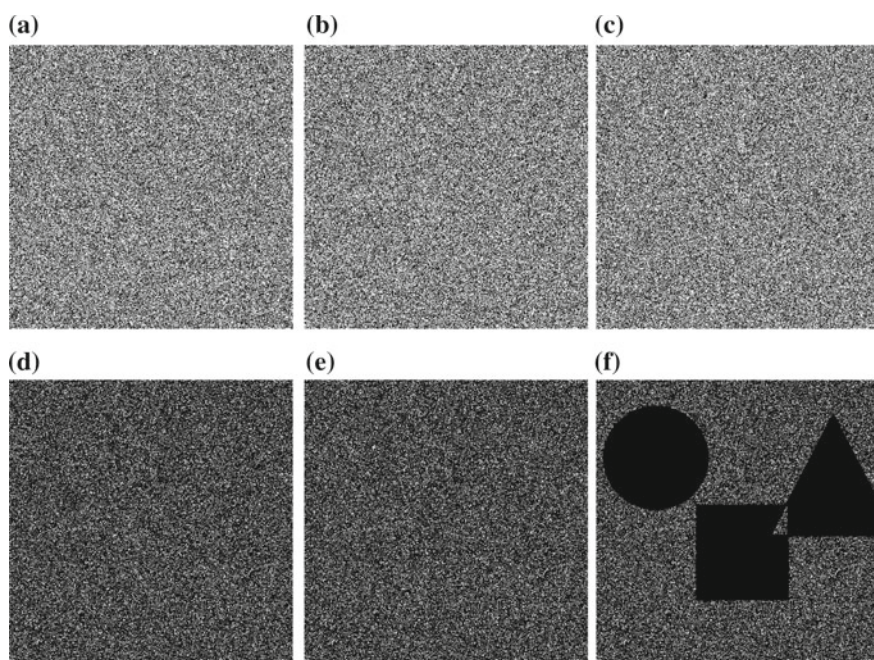


Fig. 2.6 Experimental result for $(3, 3)$ -threshold random grid. **a** Share 1. **b** Share 2. **c** Share 3. **d** Stacking of share 1 and share 2. **e** Stacking of share 1 and share 3. **f** Stacking of three shares

Algorithm 3 (n, n) -threshold random grid VC [2].**Require:**

Binary secret image: $x[\mathbf{n}] \in \{0, 1\}$.

Ensure:

n binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

- 1: $(s_1[\mathbf{n}], R_1[\mathbf{n}]) \leftarrow \text{RandGrid22}(x[\mathbf{n}])$. // See Algorithm 2
- 2: **for** $k \leftarrow 2$ to $n - 1$ **do**
- 3: $(s_k[\mathbf{n}], R_k[\mathbf{n}]) \leftarrow \text{RandGrid22}(R_{k-1}[\mathbf{n}])$.
- 4: **end for**
- 5: $s_n[\mathbf{n}] \leftarrow R_{n-1}[\mathbf{n}]$.

Algorithm 4 $(2, n)$ -threshold random grid VC [2].**Require:**

Binary secret image: $x[\mathbf{n}] \in \{0, 1\}$.

Ensure:

n binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

- 1: Define $\Omega_0 = \{\mathbf{n} : x[\mathbf{n}] = 0\}$, $\Omega_1 = \{\mathbf{n} : x[\mathbf{n}] = 1\}$.
- 2: **for** every pixel location \mathbf{n} **do**
- 3: $s_1[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$.
- 4: **for** $k \leftarrow 2$ to n **do**
- 5: **if** $\mathbf{n} \in \Omega_0$ **then**
- 6: $s_k[\mathbf{n}] \leftarrow s_{k-1}[\mathbf{n}]$.
- 7: **else**
- 8: $s_k[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$.
- 9: **end if**
- 10: **end for**
- 11: **end for**
- 12: Output share images $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

The (k, n) -threshold random grid VC is also sample based. For a position \mathbf{n} , a secret pixel is repeatedly split into two sub-pixels, till we get k sub-pixels. Then, k share images are randomly chosen and the k sub-pixels from last step are filled to position \mathbf{n} on these k shares. For the other $n - k$ share images, the current position \mathbf{n} is filled with random bits. Repeat the above process, till all pixels are filled on each share. This algorithm is summarized in Algorithm 5. To understand that the contrast condition and security condition are satisfied, one can set $k = n$, then this algorithm degenerates to Algorithm 3. For $k < n$, the other $n - k$ shares acts as random noise. So they won't affect the security condition. When stacking, these noises decrease the quality of the target image, but the contrast between white region and black region is still nonzero.

Algorithm 5 (k, n) -threshold random grid VC [3].**Require:**Binary secret image: $x[\mathbf{n}] \in \{0, 1\}$.**Ensure:** n binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

```

1: for every pixel location  $\mathbf{n}$  do
2:    $(r_1, r_2) \leftarrow \text{RandGrid22}(x[\mathbf{n}])$ .
3:   for  $i \leftarrow 2$  to  $k - 1$  do
4:      $(r_i, r_i + 1) \leftarrow \text{RandGrid22}(r_i)$ .
5:   end for
6:    $\Omega \leftarrow k$  random integers from  $\{1, 2, \dots, n\}$ .
7:   for every  $i \in \Omega$  do
8:      $s_i[\mathbf{n}] \leftarrow r_i$ .
9:   end for
10:  for every  $i \notin \Omega$  do
11:     $s_i[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$ .
12:  end for
13: end for
14: Output share images  $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$ .

```

2.4.1 Generalized Random Grid

Recall that, Wu and Sun generalized the classical random grid VC, where the probability of white pixels on the first share can be larger than the default value 1/2 that is assumed by previous algorithms [17]. This modification can help to increase the contrast for $(2, n)$ schemes.

Our purpose here is to help the reader to understand the basic idea and implementation. A similar construction for probabilistic VC is also reviewed. Three algorithms are introduced in Wu's work, including $(2, 2)$ -threshold, $(2, n)$ -threshold and (n, n) -threshold algorithms. The $(2, n)$ case can be summarized in Algorithm 6.

Algorithm 6 $(2, n)$ -threshold generalized random grid VC [17].**Require:**Binary secret image: $x[\mathbf{n}] \in \{0, 1\}$.Probability of white pixels: p .**Ensure:** n binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

```

1: for each position  $\mathbf{n}$  do
2:   Generate share 1:  $s_1[\mathbf{n}]$  from  $\{0, 1\}$  with probabilities  $\{p, 1 - p\}$ .
3:   if  $x[\mathbf{n}] = 0$  then
4:      $s_k[\mathbf{n}] \leftarrow s_1[\mathbf{n}]$ , for  $k = 2, \dots, n$ .
5:   else
6:     Generate share  $k$ :  $s_k[\mathbf{n}]$  from  $\{0, 1\}$  with probabilities  $\{p, 1 - p\}$ ., where  $k = 2, \dots, n$ .
7:   end if
8: end for
9: Output share images  $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$ .

```

Algorithm 6 says that, when sharing a white pixel $x[\mathbf{n}] = 0$, one randomly generates a white pixel with probability $p > 1/2$ on share 1, then copies this pixel on share 1 to other shares. So the stacking result is white with probability p , no matter how many shares are stacked as long as $k > 2$. When sharing a black pixel $x[\mathbf{n}] = 1$, all the shares randomly and independently generate a white pixel with probability p . So, when stacking the shares, if more shares are involved, then the higher the chance that the result is black. Thus, by stacking more shares, the contrast between white secret bits and black secret bits becomes larger. For $(2, 20)$ -threshold, and when 20 shares are stacked, the contrast can be as high as 0.8 [17].

A similar construction of $(2, n)$ -threshold probabilistic VC may also be helpful to understand why such a construction can help to improve contrast. For a $(2, n)$ -threshold probabilistic VC constructed in [9], the two basis matrices are

$$\mathbf{B}_0 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \mathbf{B}_1 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \quad (2.16)$$

If we look at each share, the proportion of white pixel is $p = \frac{n-1}{n}$ and the proportion of black pixel is $\frac{1}{n}$. If $d > 1$ shares are stacked, for the region on the target image that corresponds to the white secret pixels, the proportion of white pixels is $\frac{n-1}{n}$, no matter how many shares are stacked. But for the region on the target image that corresponds to the black secret pixels, the proportion of white pixel is $\frac{n-d}{n}$, which decreases with d . So, if more shares are stacked, the contrast between black region and white region on the target image will increase.

2.5 Security Issue in Visual Cryptography

2.5.1 Strong Security and Weak Security

As a cryptography scheme, the security requirement is usually mandatory. So, when designing a VC algorithm, one usually tries to maximize the contrast, under the constraint of security. Some researchers also suggest that there are tradeoffs between security, contrast and pixel expansion. If we relax the requirement on one aspect, then maybe it is possible to improve the performance of the other two. For example, if we relax requirement on security, then it is possible to improve contrast and reduce pixel expansion [6, 10, 12].

For a VC scheme in its strict sense, the shares are usually printed on transparencies and decoding is realized by stacking, and no computation is required for decoding. In the definition of VC (Definition 2.1), for $q < k$ shares, the two sets of sub-matrices $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ are equivalent. So, an attacker, after obtaining $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$, cannot tell if

$s = 0$ or $s = 1$, no matter the computational abilities he may have at his disposal. This is called *strict sense security* or *unconditional security* [6, 10]. For this security, we assume that the attacker has infinite computational abilities.

However, considering the media of the shares and the decoding mechanism, it is reasonable to assume that if the attacker only uses his vision system to find clue of secret from the $q < k$ shares, then the ‘computation’ devices are stacking operation and HVS. Then if from the stacking result of the $q < k$ rows of the matrices, one cannot infer s , our algorithm is safe under these assumptions. We call this the *weak security*.

Weak security was proposed by Liu [12] and Iwamoto [10] independently for different types of VC systems. Liu’s work focuses on block encoding approach to size-invariant VC, while Iwamoto’s works focus on size-expanded VC (deterministic VC) for color image. A key conclusion from their work is that, by relaxing the security level to weak security, it is possible to improve the quality of the target image and reduce the pixel expansion.

In what follows, we introduce three security issues:

1. Iwamoto’s weak security.
2. Liu’s weak security.
3. Replacement attack.

In order to introduce the concept of weak security, we need to formalize some operations on the basis matrices: *row restriction* and *row stacking*, and introduce the concept of *equivalence* between two sets of matrices [10].

For (k, n) -threshold scheme, a set of participants can be represented by $\mathbf{P} = \{i_1, \dots, i_q\} \subset \{1, \dots, n\}$. Given a basis matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$, one can make another matrix by restricting the rows of \mathbf{B} to the rows specified in set \mathbf{P} . This operation is denoted by

$$\hat{\mathbf{B}} = \mathbf{B}[\mathbf{P}], \quad (2.17)$$

with

$$\hat{B}[\ell, j] = B[i_\ell, j], \quad (2.18)$$

where $\ell \in \{1, \dots, q\}$, $i_\ell \in \mathbf{P}$, and $j \in \{1, \dots, n\}$.

Another formal operation introduced by Iwamoto is stacking of rows of a matrix. Let a matrix \mathbf{B} be partitioned as

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \in \mathbb{Z}_2^{n \times m}. \quad (2.19)$$

Then, the stacking of rows of \mathbf{B} is denoted as

$$\eta(\mathbf{B}) = \mathbf{b}_1 \vee \dots \vee \mathbf{b}_n, \quad (2.20)$$

and the result is a $1 \times m$ vector. Two matrices \mathbf{A} and \mathbf{B} are said to be equivalent if one can be obtained by column permutation of the other, and is denoted as $\mathbf{A} \sim \mathbf{B}$. A set of matrices \mathbf{C} that contains all possible column permutations of a given matrix \mathbf{C} is denoted as $\mathbf{C} = \langle \mathbf{C} \rangle$.

Iwamoto's weak security is defined for color image. We tailor it here for binary image. For binary (black and white) secret image, a deterministic VC scheme with weak security can be defined as:

Definition 2.2 (k, n) -threshold VC with weak security is defined by two collections of matrices $\mathbf{C}_0 = \langle \mathbf{B}_0 \rangle$ and $\mathbf{C}_1 = \langle \mathbf{B}_1 \rangle$, such that the following two conditions should be satisfied:

1. **Contrast condition:** For any $\mathbf{C} \in \mathbf{C}_0$, and a set of participants $\mathbf{P} = \{i_1, \dots, i_q\}$, where $q \geq k$, they must satisfy: $\mathcal{H}(\eta(\mathbf{C}[\mathbf{P}])) \leq d - \alpha$. For any $\mathbf{C} \in \mathbf{C}_1$, $\mathcal{H}(\eta(\mathbf{C}[\mathbf{P}])) \geq d$.
2. **Security condition:** For any $q < k$ and any set of q participants $\mathbf{P} = \{i_1, \dots, i_q\}$, they must satisfy $\eta(\mathbf{B}_0[\mathbf{P}]) \sim \eta(\mathbf{B}_1[\mathbf{P}])$.

Comparing Definitions 2.1 with 2.2, weak security only imposes restriction on stacking result of $q < k$ shares, while strong security requires that the two sets of sub-matrices are distinguishable. From sub-matrix to stacking result (a vector) is a many-to-one mapping. So, imposing restrictions only on mapping result leaves more space and freedom in designing the basis matrices. From this consideration, it is possible to improve the visual quality of the target image, by imposing only weak security on the design.

Iwamoto also proved that, for binary secret image, weak security is equivalent to strong security. Namely, for binary secret image, a VC algorithm can be designed to meet the weak security requirement without sacrificing strong security. For color secret image, however, an algorithm designed to meet the weak security may not be strongly secure.

While Iwamoto's work focuses on deterministic VC with pixel expansion, Liu's work focuses on block (multi-pixel) encoding, where all secret pixels in a block are encoded together. The block encoding is shown in Fig. 2.7.

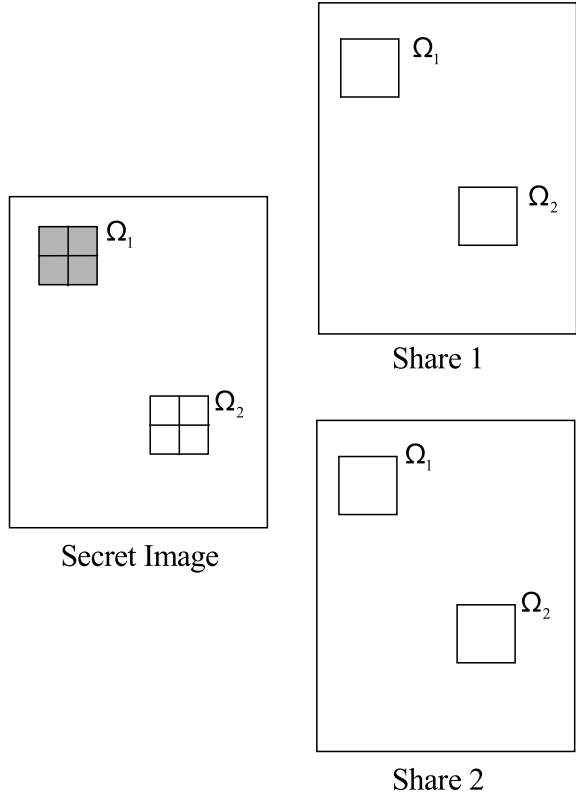
For block encoding, each block is treated as an encoding unit. Let Ω be a set of indices for pixels in a block. Since the scheme is size-invariant, so the size of the shares are the same as the size of the secret image. A block at Ω_1 on secret image corresponds to blocks at Ω_1 on share images. Liu's weak security for block encoding can be summarized in Definition 2.3.

Definition 2.3 (k, n) -threshold block encoding with weak security: Let the size of each block Ω be m . Suppose that we have two blocks Ω_1 and Ω_2 on the secret image, where $\mathcal{H}(x[\Omega_1]) > \mathcal{H}(x[\Omega_2])$. Each secret block $x[\Omega_b]$, $b = 0, 1$ is divided into n share blocks $s_1[\Omega_b], \dots, s_n[\Omega_b]$. The following two conditions should be satisfied.

1. **Contrast condition:** For $q \geq k$ and any q shares i_1, \dots, i_q , the stacking result should satisfy:

$$\mathbb{E} \{ \mathcal{H}(s_{i_1}[\Omega_1] \vee \dots, s_{i_q}[\Omega_1]) \} > \mathbb{E} \{ \mathcal{H}(s_{i_1}[\Omega_2] \vee \dots, s_{i_q}[\Omega_2]) \}, \quad (2.21)$$

Fig. 2.7 A demonstration of block encoding approach to size-invariant VC ($n = 2$) [12]



where the expectation \mathbb{E} is with respect to all possible stacking results at their respective locations Ω_b , $b = 1, 2$.

2. **Security condition:** for $q < k$ and any q shares i_1, \dots, i_q , the stacking result should satisfy:

$$\begin{aligned} \mathbb{E} \{ \mathcal{H} (s_{i_1} [\Omega_1] \vee \dots, s_{i_q} [\Omega_1]) \} &= \mathbb{E} \{ \mathcal{H} (s_{i_1} [\Omega_2] \vee \dots, s_{i_q} [\Omega_2]) \} , \\ \mathbb{V} \{ \mathcal{H} (s_{i_1} [\Omega_1] \vee \dots, s_{i_q} [\Omega_1]) \} &= \mathbb{V} \{ \mathcal{H} (s_{i_1} [\Omega_2] \vee \dots, s_{i_q} [\Omega_2]) \} , \end{aligned}$$

where \mathbb{V} denotes variance.

Liu's definition of weak security also imposes requirement on stacking results, instead of the collection of shares.

As is illustrated in Fig. 2.8, in replacement attack [4], an attacker is able to intercept one copy of the shares, say share 1, and hence can produce a fake share 2 according to his fake secret. Then he can send the fake share to the receiver. If the receiver stack the share 1 with the fake share 2, then the fake secret will be revealed.

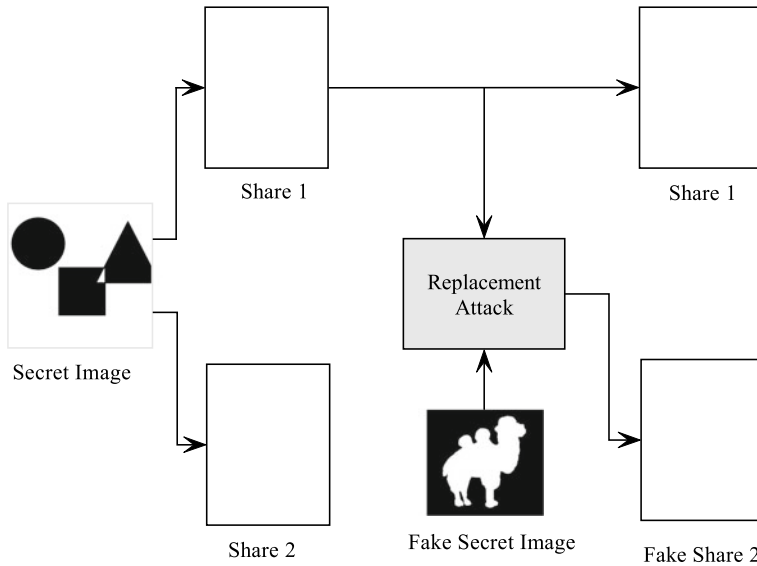


Fig. 2.8 A demonstration of replacement attack

As a countermeasure to this attacking scenario, one can embed another secret image, called a watermark, to serve as an authentication mechanism [7, 8, 16, 19]. This solution will be discussed in Chap. 4.

2.6 Summary

This chapter reviewed three common VC frameworks, including deterministic VC, probabilistic VC and random grid VC. Both the probabilistic VC and the random grid VC are size-invariant. The concept of block encoding is also touched when introducing Liu's weak security, which will be elaborated in later chapters. We focus on intuitions behind the frameworks and examples, instead of mathematical rigor. The notion of weak security is also briefly reviewed. Imposing weak security requirement provides one possible way to improve visual quality of target image.

References

1. Ateniese, G., Blundo, C., Santis, A.D., Stinson, D.R.: Visual cryptography for general access structures. *Inf. Comput.* **129**(2), 86–106 (1996)
2. Chen, T.H., Tsao, K.H.: Visual secret sharing by random grids revisited. *Pattern Recognit.* **42**(9), 2203–2217 (2009)

3. Chen, T.H., Tsao, K.H.: Threshold visual secret sharing by random grids. *J. Syst. Softw.* **84**(7), 1197–1208 (2011)
4. Fang, W.P., Lin, J.C.: Visual cryptography with extra ability of hiding confidential data. *J. Electron. Imaging* **15**(2), 1–7 (2006)
5. Hou, Y.C., Wei, S.C., Lin, C.Y.: Random-grid-based visual cryptography schemes. *IEEE Trans. Circuits Syst. Video Technol.* **24**(5), 733–744 (2014)
6. Hsu, C.S., Tu, S.F., Hou, Y.C.: An optimization model for visual cryptography schemes with unexpanded shares. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) *Foundations of Intelligent Systems*, pp. 58–67. Springer, Heidelberg (2006)
7. Huang, H.C., Fang, W.C.: Authenticity preservation with histogram-based reversible data hiding and quadtree concepts. *Sensors* **11**(10), 9717–9731 (2011)
8. Huang, H.C., Chang, F.C., Fang, W.C.: Reversible data hiding with histogram-based difference expansion for QR code applications. *IEEE Trans. Consum. Electron.* **57**(2), 779–787 (2011)
9. Ito, R., Kuwakado, H., Tanaka, H.: Image size invariant visual cryptography. *IEICE Trans. Fundam.* **E82-A**(10), 2172–2177 (1999)
10. Iwamoto, M.: A weak security notion for visual secret sharing schemes. *IEEE Trans. Inf. Forensics Secur.* **7**(2), 372–382 (2012)
11. Kafri, O., Keren, E.: Encryption of pictures and shapes by random grids. *Opt. Lett.* **12**(6), 377–379 (1987)
12. Liu, F., Guo, T., Wu, C., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. *J. Vis. Commun. Image Represent.* **23**(2), 331–342 (2012)
13. Naor, M., Shamir, A.: Visual cryptography. In: *Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT 1994)*, Perugia, Italy, pp. 1–12 (1994)
14. Shyu, S.J.: Image encryption by random grids. *Pattern Recognit.* **40**(3), 1014–1031 (2007)
15. Shyu, S.J.: Image encryption by multiple random grids. *Pattern Recognit.* **42**(7), 1582–1596 (2009)
16. Thajeel, S.A.N., Sulong, G.: A novel approach for detection of copy move forgery using completed robust local binary pattern. *J. Inf. Hiding Multimed. Signal Process.* **6**(2), 351–364 (2015)
17. Wu, X.T., Sun, W.: Generalized random grid and its applications in visual cryptography. *IEEE Trans. Inf. Forensics Secur.* **8**(9), 1541–1553 (2013)
18. Yang, C.N.: New visual secret sharing schemes using probabilistic method. *Pattern Recognit. Lett.* **25**(4), 481–494 (2004)
19. Yang, C.Y., Wang, W.F., Wang, Y.F., Wang, R.Z.: A simple watermarking scheme with high perceptual quality for still color images based on RWM and centroid. *J. Inf. Hiding Multimed. Signal Process.* **6**(3), 577–590 (2015)

Chapter 3

Digital Halftoning



3.1 Introduction to Digital Halftoning

Most printing devices and some display devices can only render limited number of colors [9]. For example, a typical laser printer can only output a black dot or ‘no dot’ at a time. In order to print grayscale images, we must quantize the input color to coarser scales, such that the perceived color is still similar to the original color when observed by human eyes.

In this chapter, we assume that the input image is a grayscale image and each pixel is normalized to the range $[0, 1]$. If the pixel is quantized, 8-bit quantization is assumed. The printer is assumed to be able to produce only two colors: black (black dot) and white (no dot).

The image quantization problem then can be stated as follows: Given an input grayscale image $x[i, j]$, the quantizer produces a binary image $y[i, j] \in \{0, 1\}$ such that it is visually similar to the input image $x[i, j]$ when viewing from sufficient distance. Let \mathcal{H} be a system representing the human visual perception, then the digital halftoning problem can be formulated as an optimization problem:

$$\min_{\mathbf{y} \in \mathbb{Z}_2^{M \times N}} \mathcal{H} \{ \mathbf{x} - \mathbf{y} \}. \quad (3.1)$$

where the images are of size $M \times N$. Solving this optimization problem directly is usually impractical due to the high dimensional searching space that has $2^{M \times N}$ feasible solutions.

Many heuristic approaches are proposed for solving the halftoning problem in (3.1), including constant threshold bi-level quantization, ordered dithering, error diffusion and direct binary search (DBS).

3.2 Bi-level Quantization

The simplistic approach is to quantize each sample independently using a bi-level quantizer [6]:

$$y[i, j] = \begin{cases} 1, & \text{if } x[i, j] > T, \\ 0, & \text{otherwise,} \end{cases} \quad (3.2)$$

where T is a constant threshold. For this quantizer, the two reconstruction levels are $(y_0, y_1) = (0, 1)$ and the decision boundaries are $(d_0, d_1, d_2) = (0, T, 1)$. The threshold T can be designed by minimizing the mean-squared error:

$$\text{MSE}(T) = \sum_{i=0}^1 \int_{d_i}^{d_{i+1}} (x - y_i)^2 p_X(x) dx, \quad (3.3)$$

where $p_X(x)$ is the PDF of the samples of the input image x . It can be modeled as uniform distribution over the interval $[0, 1]$, considering the wide varieties of the histogram of natural images. By solving $\frac{\partial \text{MSE}(T)}{\partial T} = 0$, one can easily find that $T = 1/2$.

A halftoning result using bi-level quantizer on Lena image is shown in Fig. 3.1. As can be seen, only some high contrast edges and textures are preserved, a lot of low contrast details are lost. The halftone image is not visually similar to the original grayscale image.

Several important properties of the grayscale images and halftone images are not utilized in simple bi-level quantization halftoning. First, a grayscale image usually consists of smooth regions separated by edges. Second, the HVS can only perceive a local average of the halftone image in a small region, or equivalently, the HVS is a



Fig. 3.1 Halftoning by bi-level quantizer. **a** Original Lena image. **b** Halftone image

low-pass system. One way to utilize these properties is to use a distributed multilevel quantizer, where the quantization levels are spatially distributed in a small region. This idea leads to ordered dithering.

3.3 Ordered Dithering

Considering the fact that the HVS is a low-pass system that only perceives the result of local average, one can design a distributed multilevel quantizer in a small region. For example, using a 2×2 block, we may design a five-level quantizer with thresholds $(1/8, 3/8, 5/8, 7/8)$ and reconstruction points $(0, 1/4, 2/4, 3/4, 1)$. Then the thresholds are placed in a 2×2 block as:

$$\mathbf{T} = \begin{bmatrix} 3/8 & 5/8 \\ 7/8 & 1/8 \end{bmatrix} \quad (3.4)$$

Suppose that the input value is a constant 2×2 block \mathbf{x} :

$$\mathbf{x} = \begin{bmatrix} c & c \\ c & c \end{bmatrix} \quad (3.5)$$

then we perform sample-wise quantization as in (3.2) as:

$$y[i, j] = \begin{cases} 1, & \text{if } c > T[i, j], \\ 0, & \text{otherwise.} \end{cases} \quad (3.6)$$

So, the number of black pixels in y will be approximately proportional to the value of c in \mathbf{x} . Or, one can think of the proportion of black pixels in y as representing the reconstruction levels $(0, 1/4, 2/4, 3/4, 1)$. Thus, by arranging the quantization levels in a small block of thresholds, we get a distributed multi-level quantizer. The reconstruction levels are the proportion of black pixels in the output block. This is usually referred to as *Ordered Dithering* since we can consider the thresholds in \mathbf{T} as dithering of the constant threshold $1/2$.

In general, for ordered dithering, the thresholds are specified by an *index matrix*. For a $L \times L$ block, the index matrix contains a list of all the L^2 integer numbers $\{0, \dots, L^2 - 1\}$. For example, for $L = 2$, the index matrix can be

$$\mathbf{I} = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}, \quad (3.7)$$

which specifies the sequence of turning on the corresponding pixels to black. From it, the threshold matrix can be determined:

$$T[i, j] = \frac{I[i, j] + 1/2}{L^2} \quad (3.8)$$

which gives the threshold matrix:

$$\mathbf{T} = \begin{bmatrix} 7/8 & 1/8 \\ 5/8 & 3/8 \end{bmatrix} \quad (3.9)$$

In general, the size of the image is larger than the size of the threshold matrix. So, the image is segmented into blocks having the same size of the threshold matrix, and then we use the threshold matrix to quantize each block.

$$y[i, j] = \begin{cases} 1, & \text{if } x[i, j] > T[i \bmod L, j \bmod L], \\ 0, & \text{otherwise.} \end{cases} \quad (3.10)$$

The size of the threshold matrix is usually much larger than 2×2 , such as 8×8 , 16×16 or even the size of the image to be processed. For such a large size, the arrangement of thresholds is not a trivial issue. Different arrangements of the quantization levels lead to different halftoning effects. If nearby quantization levels are spatially close to each other, then we have *clustered dot dithering*. If nearby quantization levels are located far away from each other, then we have *dispersed dot dithering*.

3.3.1 Clustered Dot Dithering

The term ‘clustered dot’ comes from the fact that when using this dithering, the black dots are clustered together when the image pixels are smoothly varying. For this reason, changing the input gray level is equivalent to changing the size of the whole clustered dot, and is usually called *Amplitude Modulation* (AM) [9].

To construct the index matrix, we start from the center of the matrix, and put the integers from 0 to $L^2 - 1$ into it, by following a spiral curve with increasing radius. This is shown in Fig. 3.2.

Similarly, we can get index matrix for $L = 8$:

$$I = \begin{bmatrix} 62 & 57 & 48 & 36 & 37 & 49 & 58 & 63 \\ 56 & 47 & 35 & 21 & 22 & 38 & 50 & 59 \\ 46 & 34 & 20 & 10 & 11 & 23 & 39 & 51 \\ 33 & 19 & 9 & 3 & 0 & 4 & 12 & 24 \\ 32 & 18 & 8 & 2 & 1 & 5 & 13 & 25 \\ 45 & 31 & 17 & 7 & 6 & 14 & 26 & 40 \\ 55 & 44 & 30 & 16 & 15 & 27 & 41 & 52 \\ 61 & 54 & 43 & 29 & 28 & 42 & 53 & 60 \end{bmatrix}$$

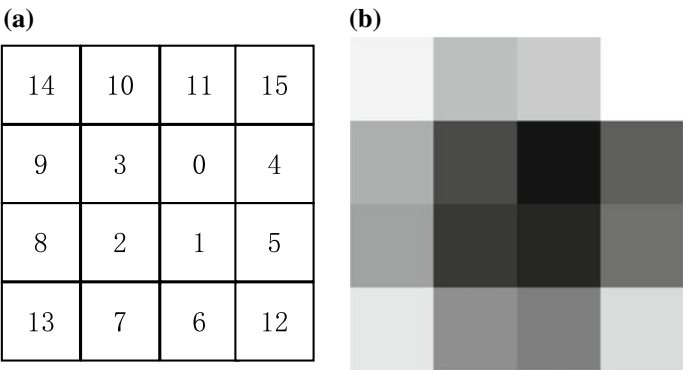
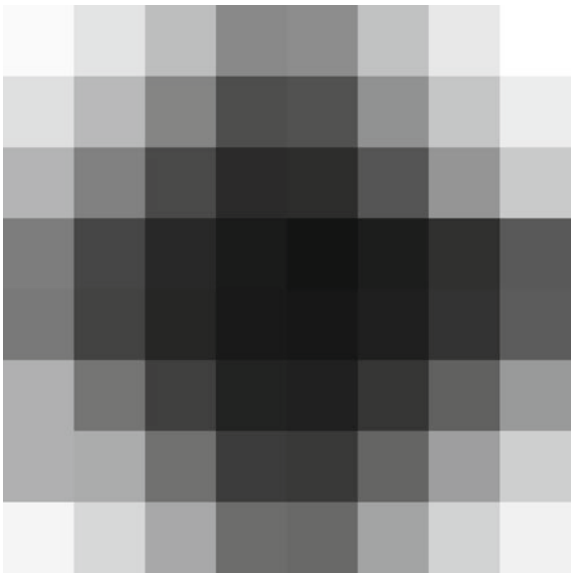


Fig. 3.2 Index matrix for 4×4 clustered dot dithering. **a** Index matrix. **b** Index matrix shown as an image

Fig. 3.3 A 8×8 index matrix for clustered dot dithering, shown as a grayscale image

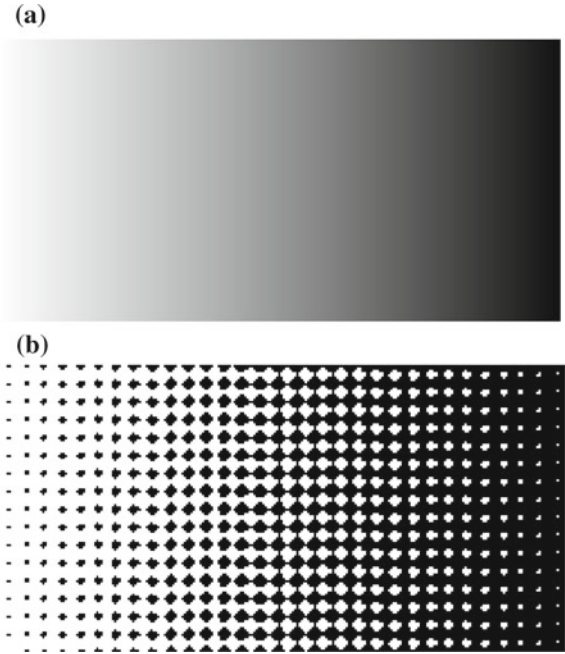


which is shown as an image in Fig. 3.3

Using this index matrix, we halftone a test image having continuous varying gray scale, as shown in Fig. 3.4. Visually, the size of the marco-dot is changing with increasing blackness.

One drawback of clustered dot dithering, due to the dot-size modulation, is that the halftone result has reduced resolution. The larger the size of the block, the lower the resolution. This drawback can be remedied by dispersed dot dithering.

Fig. 3.4 Clustered dot dithering as amplitude modulation. **a** Grayscale image. **b** Halftone image



3.3.2 Dispersed Dot Dithering

For dispersed dot dithering, such as Bayer's dithering, adjacent thresholds are separated as far as possible from each other. Bayer's index matrix is defined recursively [2], starting from trivial 2×2 index matrix

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

and

$$\mathbf{I}_{2n} = \begin{bmatrix} 4\mathbf{I}_n + 1 & 4\mathbf{I}_n + 2 \\ 4\mathbf{I}_n + 3 & 4\mathbf{I}_n \end{bmatrix}$$

for $n = 2, 4, \dots$. The 8×8 threshold matrix for dispersed dot dithering is shown in Fig. 3.5a, and the halftone result for the grayscale image in Fig. 3.4a is shown in Fig. 3.5b. Since the thresholds are separated from each other, different blackness corresponds to different frequency of black dots. The resolution is significantly improved.

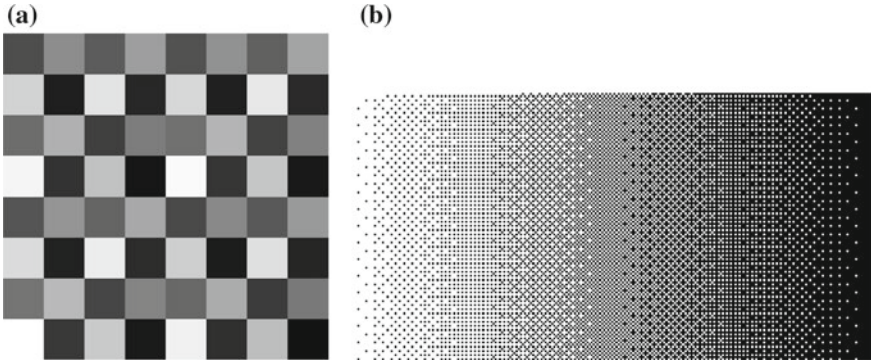


Fig. 3.5 Dispersed dithering as frequency modulation (FM). **a** The threshold matrix. **b** Halftone result

3.4 Error Diffusion and Its Mathematical Model

Instead of using a distributed quantizer with constant thresholds, error diffusion uses adaptive threshold and it compensates quantization error locally. In this section, we briefly review the error diffusion algorithm. To better understand the tone reproducibility of error diffusion, we will also review a linear model for error diffusion. This model will be used to show the contrast condition in AbS-based VC, in Chap. 5.

3.4.1 Error Diffusion

A block diagram for error diffusion halftoning is shown in Fig. 3.6. The modified input signal $\hat{x}[i, j]$ is quantized by a simple bi-level and constant threshold quantizer \mathcal{Q} , and produces binary output image $y[i, j]$. The quantization error $e[i, j] = y[i, j] - \hat{x}[i, j]$ is fed back to the input and distributed to neighboring un-quantized samples using a diffusion filter $h[i, j]$. Thus, the quantization error can be compensated by neighboring pixels. The received error for pixel $[i, j]$ is thus

$$\hat{e}[i, j] = \sum_{(k,l) \in \mathcal{N}(i,j)} h[k, l]e[i - k, j - l], \quad (3.11)$$

where $\mathcal{N}(i, j)$ is a small neighborhood around current pixel $[i, j]$.

The diffusion filter, also called the diffusion kernel, plays an important role in the quality of the halftone image. First, to ensure that all errors are distributed, the filter coefficients must sum to 1, i.e.,

Fig. 3.6 Block diagram for error diffusion halftoning

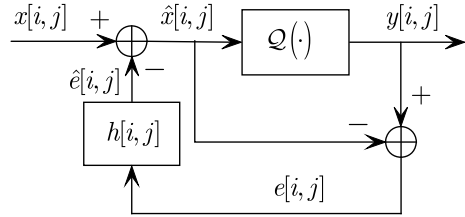
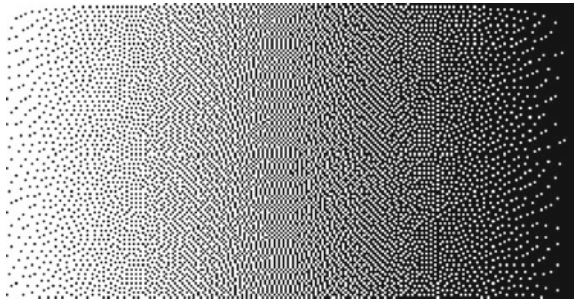


Fig. 3.7 The Floyd and Steinberg kernel

	$[i, j]$	$\frac{7}{16}$
$\frac{3}{16}$	$\frac{5}{16}$	$\frac{1}{16}$

Fig. 3.8 Halftoning of test image using error diffusion and Floyd and Steinberg kernel



$$\sum_i \sum_j h[i, j] = 1$$

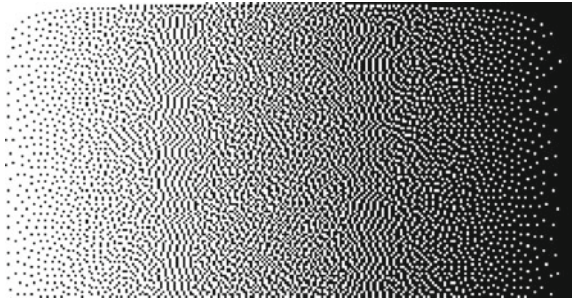
The popular kernel proposed by Floyd and Steinberg is shown in Fig. 3.7 [3]. The error $e[i, j]$ at $[i, j]$ will distribute a fraction of $\frac{7}{16}$ to position $[i, j + 1]$, a fraction of $\frac{3}{16}$ to position $[i + 1, j - 1]$, etc.

Using the Floyd and Steinberg kernel, we halftone the continuous tone image shown in Fig. 3.4a, the result is shown in Fig. 3.8. Compared to the result of dispersed dithering in Fig. 3.5b, we see that error diffusion can render the continuous tone better. However, one may notice that, for dark and bright region, there are worm-like structures. These are usually called worm artifact.

Using larger mask can remove the artifact produced by the Floyd and Steinberg mask, but requires higher computation load [5]. The Jarvis kernel is shown in Fig. 3.9 and the halftoning result using Jarvis kernel on the continuous tone image (Fig. 3.4a)

Fig. 3.9 The Jarvis kernel

		$[i, j]$	$\frac{7}{48}$	$\frac{5}{48}$
$\frac{3}{48}$	$\frac{5}{48}$	$\frac{7}{48}$	$\frac{5}{48}$	$\frac{3}{48}$
$\frac{1}{48}$	$\frac{3}{48}$	$\frac{5}{48}$	$\frac{3}{48}$	$\frac{1}{48}$

Fig. 3.10 Halftoning of test image using error diffusion and Jarvis kernel

is shown in Fig. 3.10. Comparing Figs. 3.10 and 3.8, we see that the ‘worm’ effect is largely reduced.

The approach described by (3.11) is an error collection approach, or filtering approach. In practice, the error diffusion algorithm is usually implemented as an ‘error pushing’ approach, which is described in Algorithm 7. The support Ω of the diffusion kernel is defined in a coordinate system with the current pixel $[i, j]$ at the origin. For example, for Floyd and Steinberg kernel, the support is:

$$\Omega = \{[0, 1], [1, -1], [1, 0], [1, 1]\}. \quad (3.12)$$

3.4.2 Mathematical Model

The error diffusion loop, as a nonlinear feedback loop, is in general difficult to analyze and design. To better understand the behavior of this loop, it is necessary to simplify it into a linear one that still captures the essential features of its nonlinear counterpart.

Knox found that the error image $e[i, j]$ is correlated with the input image, as shown in Fig. 3.11. A linear model consisting of a linear term and an additive noise term can better model the sharpening effect from error diffusion halftoning [8].

Algorithm 7 Implementation of error diffusion halftoning by pushing error forward.

Input:

grayscale image $x[i, j]$ with size $M \times N$.

Diffusion kernel $h[k, l]$, where $[k, l] \in \Omega$ and Ω is support of the Kernel.

Output:

Halftone image $y[i, j]$ with size $M \times N$.

- 1: **for** each pixel $[i, j]$ **do**
 - 2: Quantize modified input: $y[i, j] = \mathcal{Q}(\hat{x}[i, j])$.
 - 3: Error calculation: $e[i, j] = y[i, j] - \hat{x}[i, j]$.
 - 4: Error diffusion:
 - 5: **for** every $[k, l] \in \Omega$ **do**
 - 6: $\hat{x}[i + k, j + l] = \hat{x}[i + k, j + l] - h[k, l]e[i, j]$.
 - 7: **end for**
 - 8: **end for**
-

Fig. 3.11 Error image from error diffusion halftoning



Kite et al. confirmed this linear model and estimated parameters for the linear term and the noise term [7]. This model will be used in our theoretic analysis in Chap. 5.

Referring to Fig. 3.6, the nonlinear quantizer $y[i, j] = \mathcal{Q}(\hat{x}[i, j])$ can be approximated by a linear model:

$$y[i, j] = \gamma \hat{x}[i, j] + q[i, j], \quad (3.13)$$

where the coefficient γ models correlation between the error image $e[i, j]$ and the input image $x[i, j]$. The term $q[i, j]$ is used to account for the component that is uncorrelated to the input.

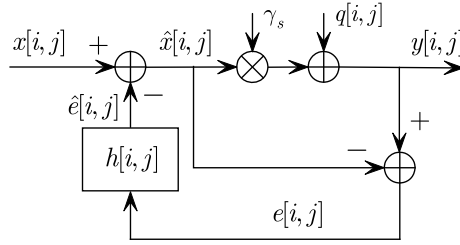


Fig. 3.12 Linear model for error diffusion halftoning

Next, we derive the transfer function for this system by following a similar approach in [7]. The equivalent linear system in Fig. 3.12 has two inputs $x[i, j]$ and $q[i, j]$, and one output $y[i, j]$. Recall that, for a linear system \mathcal{L} and two input signals $x_1[i, j]$ and $x_2[i, j]$, the output is

$$\mathcal{L}(\alpha x_1[i, j] + \beta x_2[i, j]) = \alpha \mathcal{L}(x_1[i, j]) + \beta \mathcal{L}(x_2[i, j]), \quad (3.14)$$

where α and β are two scaling factors. So, the transfer function can be derived for each of the two input signals.

By setting $q[i, j] = 0$, we get the system for *signal path*, as shown in Fig. 3.13a, where γ_s is the scaling factor for signal path. Using \mathcal{Z} -transform, this system can be expressed by the following set of equations:

$$E(\mathbf{z}) = (\gamma_s - 1) X(\mathbf{z}) \quad (3.15)$$

$$\hat{X}(\mathbf{z}) = X(\mathbf{z}) - H(\mathbf{z})E(\mathbf{z}) \quad (3.16)$$

$$Y_s(\mathbf{z}) = \gamma_s \hat{X}(\mathbf{z}), \quad (3.17)$$

where $\mathbf{z} = (z_1, z_2)$ is the z variable for two-dimensional \mathcal{Z} -transform, and Y_s is the system output for signal path. After some simple algebra, we get the output for signal path:

$$Y_s(\mathbf{z}) = \frac{\gamma_s}{1 + (\gamma_s - 1)H(\mathbf{z})} X(\mathbf{z}). \quad (3.18)$$

For *noise path*, we mute the signal input: $x[i, j] = 0$. The equivalent system is shown in Fig. 3.13b, which is described by the following set of equations:

$$Y_n(\mathbf{z}) = \gamma_n \hat{Q}(\mathbf{z}) + Q(\mathbf{z}), \quad (3.19)$$

$$\hat{Q}(\mathbf{z}) = -H(\mathbf{z})E(\mathbf{z}), \quad (3.20)$$

$$E(\mathbf{z}) = Y_n(\mathbf{z}) - \hat{Q}(\mathbf{z}), \quad (3.21)$$

where γ_n is the scaling factor for the noise path, and Y_n is the output for noise path. Thus, the output signal for noise path is:

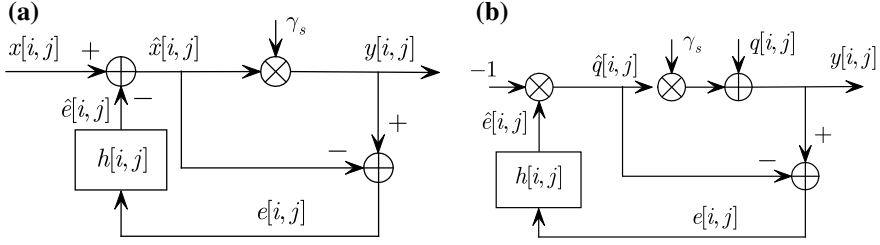


Fig. 3.13 Signal path and noise path of the linear model for error diffusion loop. **a** Signal path. **b** Noise path

$$Y_n(\mathbf{z}) = \frac{1 - H(\mathbf{z})}{1 + (\gamma_n - 1) H(\mathbf{z})} Q(\mathbf{z}). \quad (3.22)$$

From (3.18) and (3.22), we get the overall output signal [7]:

$$Y(\mathbf{z}) = \underbrace{\frac{\gamma_s}{1 + (\gamma_s - 1) H(\mathbf{z})} X(\mathbf{z})}_{\text{Signal Transfer Function (STF)}} + \underbrace{\frac{1 - H(\mathbf{z})}{1 + (\gamma_n - 1) H(\mathbf{z})} Q(\mathbf{z})}_{\text{Noise Transfer Function (NTF)}}. \quad (3.23)$$

The coefficient γ_s can be found by minimizing the error energy between $y[i, j]$ and $\gamma_s \hat{x}[i, j]$:

$$\min_{\gamma_s} \left(\sum_{i=1}^M \sum_{j=1}^N (\gamma_s \hat{x}[i, j] - y[i, j])^2 \right), \quad (3.24)$$

where the image is of size $M \times N$. The estimated γ_s varies from image to image. For Floyd-Steinberg filter, γ_s is close to 2. The coefficient γ_n for noise path was found experimentally to be 1. Please refer to [7] for more results for different images and diffusion filters.

By setting $\gamma_s = 2$ and $\gamma_n = 1$, and noting that $H(\mathbf{z})$ is a low-pass filter with unity DC gain, we observe the following from (3.23). First, the STF is a high-pass filter with unity DC gain, which means that the loop can reproduce slow-varying signals and sharpen fast-varying signal such as edges; Second, the NTF is also a high-pass filter with zero DC gain, which means that DC component of the noise $q[i, j]$ is suppressed by the loop. High-frequency component of the noise is left in halftone image. But, it can be filtered out by the equivalent low-pass filter of the HVS.

3.5 Direct Binary Search

While the ordered dithering uses a distributed multi-level quantizer, and error diffusion uses an adaptive bi-level quantizer, the DBS algorithm attempts to solve the optimization problem in (3.1) directly [1]. DBS can be viewed as an iterative greedy algorithm for solving (3.1).

To formulate the optimization problem, we must have a mathematical model modeling how the HVS perceives a halftone image. The perception of black dot is related to how a black dot is rendered, and how the HVS perceives dots.

A black pixel is rendered as a dot on paper. This dot has different sizes and shapes depending the type of printers used, which can be described by a profile function $f(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2)$ is the spatial coordinate on paper. When the HVS sees a dot, this dot stimulate a region on retinal. So, a point is spread as a region (bell-shaped) on retinal. The function modeling this is called a *point spread function* (PSF). The net effect is that, the HVS only perceive a smoothed version of the dots. Let $g(\mathbf{x})$ be the PSF. Then, a black point goes through the composited system $f(\mathbf{x}) \otimes g(\mathbf{x})$. Here, we only use the sampled version of this composite filter. The sampled version of the point profile and PSF are denoted as $p[\mathbf{n}]$ and $h_{\text{HVS}}[\mathbf{n}]$, respectively.

Having obtained the sampled version of the PSF $h_{\text{HVS}}[\mathbf{n}]$ and the point profile $p[\mathbf{n}]$, we can formulate the optimization problem as follows. Let

$$h[\mathbf{n}] = p[\mathbf{n}] \otimes h_{\text{HVS}}[\mathbf{n}]$$

be the combining effect of these two filters. Let $\hat{x}[\mathbf{n}] = x[\mathbf{n}] \otimes h[\mathbf{n}]$ and $\hat{y}[\mathbf{n}] = y[\mathbf{n}] \otimes h[\mathbf{n}]$ be the sampled version of perceived grayscale image and halftone image, respectively. Then, one chooses the halftone image $y[\mathbf{n}]$ to minimize

$$E = \|\hat{x}[\mathbf{n}] - \hat{y}[\mathbf{n}]\|^2 \quad (3.25)$$

$$= \|x[\mathbf{n}] \otimes h[\mathbf{n}] - y[\mathbf{n}] \otimes h[\mathbf{n}]\|^2 \quad (3.26)$$

Direct solution by brute force approach is prohibitive since the searching space is of order $2^{M \times N}$ for an image with size $M \times N$.

The DBS approach is an iterative greedy algorithm. It starts with an initial $y[\mathbf{n}]$, using one of the ordered dithering halftoning or error diffusion halftoning. It can even be a randomly generated binary image. Then DBS iteratively improves the initial image.

In each iteration, all image pixels are visited sequentially, by following a certain scanning order. One pixel of output halftone image $y[\mathbf{n}]$ is determined in each step. At each pixel location, DBS uses a small 3×3 neighborhood and tries 10 different solutions to see which one can reduce the error in (3.25) to the lowest. These 10 solutions include toggling the center pixel (2 solutions) and exchanging the center pixel with one of its eight neighbors (8 solutions). The trial solution resulting in the lowest error E is retained for the current pixel (at pixel position \mathbf{n}).

The algorithm iterates till no change occur on $y[\mathbf{n}]$. Upon termination, the final $y[\mathbf{n}]$ is output as the final halftone image.

3.5.1 Direct Implementation

Based on the basic idea from last section, a direction implementation can be designed. Let the matrix Δ denotes the set of coordinates for the eight neighbors:

$$\Delta = \begin{bmatrix} -1 & -1 & -1 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & -1 & 1 & -1 & 0 & 1 \end{bmatrix} = [\Delta_1, \dots, \Delta_8], \quad (3.27)$$

where each column of Δ corresponds to displacement of one of the eight neighboring pixels. Then one iteration of the DBS algorithm can be described by Algorithm 8.

Algorithm 8 $y \leftarrow \text{DbsIteration}(\mathbf{x}, \mathbf{y})$: One iteration of direct implementation of DBS algorithm.

Input:

grayscale image \mathbf{x} of size $M \times N$.

Filter $\mathbf{h} \in \mathbb{R}^{L \times L}$.

Halftone image $\mathbf{y} \in \mathbb{Z}_2^{M \times N}$.

Output:

Halftone image $\mathbf{y} \in \mathbb{Z}_2^{M \times N}$.

```

1: for each pixel  $\mathbf{n}$  do
2:    $E_0 = \|(\mathbf{x} - \mathbf{y}) \otimes \mathbf{h}\|_2^2$ .
3:   for  $k = 1$  to 8 do
4:      $\mathbf{y}^k = \text{Exchange } y[\mathbf{n}] \text{ and } y[\mathbf{n} + \Delta_k] \text{ in } \mathbf{y}$ .
5:      $E^k = \|(\mathbf{x} - \mathbf{y}^k) \otimes \mathbf{h}\|_2^2$ .
6:   end for
7:    $\mathbf{y}^9 = \text{Flip } y[\mathbf{n}] \text{ in } \mathbf{y}$ .
8:    $E^9 = \|(\mathbf{x} - \mathbf{y}^9) \otimes \mathbf{h}\|_2^2$ .
9:    $k^* = \text{argmin}_{k=0, \dots, 9} E_k$ .
10:   $\mathbf{y} = \mathbf{y}^{k^*}$ .
11: end for
```

Using Algorithm 8 as a building block, we build the following iterative algorithm, as described in Algorithm 9. Starting with an initial random binary image \mathbf{y}^0 , the algorithm processes the output halftone image several passes, until the number of changes between two iterations falls below some threshold τ .

Algorithm 9 Direct implementation of DBS algorithm.**Input:**

Grayscale image \mathbf{x} with size $M \times N$.
 Filter $\mathbf{h} \in \mathbb{R}^{L \times L}$.
 Threshold τ .

Output:

Halftone image $\mathbf{y} \in \mathbb{Z}_2^{M \times N}$.

- 1: Initialize halftone output \mathbf{y}^0 as a random binary image.
- 2: $\mathbf{y}^1 = \text{DbsIteration}(\mathbf{x}, \mathbf{y}^0)$.
- 3: $\delta = \|\mathbf{y}^0 - \mathbf{y}^1\|_2^2$.
- 4: **while** $\delta \geq \tau$ **do**
- 5: $k = k + 1$.
- 6: $\mathbf{y}^k = \text{DbsIteration}(\mathbf{x}, \mathbf{y}^{k-1})$.
- 7: $\delta = \|\mathbf{y}^k - \mathbf{y}^{k-1}\|_2^2$.
- 8: **end while**
- 9: $\mathbf{y} = \mathbf{y}^k$.

3.5.2 Fast Implementation

One drawback of the direct implementation is its high computational load. Within one iteration and at each pixel location, every trial changes to \mathbf{y} involves filtering the whole image \mathbf{y} with the filter \mathbf{h} , in order to calculate related error E^k . Filtering the whole image is actually not necessary, considering the fact that the support of the filter \mathbf{h} is much smaller than the size of the image. One typical value for the size of support of \mathbf{h} is $L = 7$, while the size of the image is typically $M = N = 512$. So a change of a pixel on \mathbf{y} at location \mathbf{n}_0 won't affect the perceptual result at pixel location \mathbf{n}_1 if they are separated more than $\lfloor \frac{L}{2} \rfloor$ away from each other.

Fast implementation of the DBS algorithms involves expressing the error energy E as a function of correlations between the error and the filter kernel, and updating it locally. For a comprehensive review of efficient implementation of DBS, the reader is directed to [9].

3.6 Quality Measures for Halftone Image

A halftone image is intended for human observer, so the perceptual quality should be evaluated by human. For this purpose, one may use one of the many subjective quality evaluation approaches, such as five-scale scoring, etc. However, such subjective approaches are time consuming and are not appropriate to use in a halftone generation algorithm. A model for the subjective evaluation is a better choice. There are two main types of quality measures for halftone image. The first type is fidelity measure, which measures the difference between the observed original image and observed halftone image. For example, the MSE as used in DBS, and the HPSNR (Human Peak Signal-to-Noise Ratio) measure to be introduced in this section. These measures are

directly related to the human perception. The second type calculates the spatial and spectral properties of a halftone random process, such as RAPSD. In the following sections, we briefly review these two types of measures.

3.6.1 Fidelity Measures

The fidelity between the reconstructed image and the original grayscale image is characterized by tone similarity and structural similarity. The tone similarity is measured by *HPSNR* [4], which is the PSNR between two smoothed images using the lowpass filter model of the HVS. The structure similarity is measured by *MSSIM* (Mean Structure Similarity).

To calculate the HPSNR, both the original image and the reconstructed image are smoothed by a Gaussian filter. Then PSNR is calculated between these two smoothed images:

$$\text{HPSNR} = 10 \log_{10} \frac{M \times N}{\sum_{\mathbf{n}} \sum_{\mathbf{m} \in \mathcal{N}(\mathbf{n})} w[\mathbf{m}] (x[\mathbf{m}] - y[\mathbf{m}])^2}, \quad (3.28)$$

where $M \times N$ is the size of the image and $\mathcal{N}(\mathbf{n})$ is a neighborhood around the pixel \mathbf{n} . $w[\mathbf{m}]$ is the Gaussian filter model for the HVS. As a typical value, the standard deviation of the Gaussian filter is $\sigma = 2$, so a 13×13 square neighborhood should be chosen in order to cover the $\pm 3\sigma$ range [4].

The MSSIM measure combines the local luminance comparison, local contrast comparison and local correlation between two images into one metric [11].

When calculating the HPSNR and MSSIM, the dynamic range of the reference image and the reconstructed image must be adjusted to be the same. HPSNR may also be influenced by histogram equalization which is adopted by some algorithms to enhance the perceptual quality.

3.6.2 Blue Noise and Spectral Characterization

A dither pattern from halftoning can be modeled as a spatial random process. The spectral feature of this process correlates with its perceptual quality. To produce visually pleasing dither patterns, the minority pixels must be scattered as far as possible from each other. This requirement is reflected as blue noise (or high frequency noise) on the spectrum [9].

Without loss of generality, let $\phi[\mathbf{n}]$ be a halftone process, and that the pixel \mathbf{n} is a minority pixel if $\phi[\mathbf{n}] = 1$. In order to estimate the power spectrum density (PSD), $\phi[\mathbf{n}]$ is divided into K overlapping segments, each having length N . We get K sample vectors, $\{\phi_1, \dots, \phi_K\}$, then the PSD can be estimated by average

periodograms method:

$$\hat{P}(f) = \frac{1}{K} \sum_{i=1}^K \frac{|\mathcal{F}\{\phi_i\}|^2}{N},$$

where \mathcal{F} denotes 2D Fourier transform. The PSD is a two-dimensional statistics. By partitioning the 2D frequency domain into a series of annular rings, we can get two convenient 1D statistics: the radially averaged power spectrum density (RAPSD) $P(f_\rho)$ and anisotropy $A(f_\rho)$. Let the radius and the width of the annular ring $R(f_\rho)$ be f_ρ and Δ_ρ respectively, then

$$P(f_\rho) = \frac{1}{|R(f_\rho)|} \sum_{\forall f \in R(f_\rho)} \hat{P}(f),$$

where $|R(f_\rho)|$ denotes the number of elements in the set $R(f_\rho)$. For blue noise pattern, the RAPSD has zero low-frequency component, and a peak at the principal frequency, followed by flat high frequency region [10].

Strong anisotropy on halftone image results in annoying patterns, such as the worm effect [9]. Thus, to produce visually pleasing halftone image, the dots should be scattered apart and the scattering should be isotopic. *Directional distribution function* (DDF) can be used to characterize the spatial anisotropy in a halftone image. In brief, at each minority pixel, DDF partitions the directions into several sectors and counts the number of minority pixels in each sector. For an isotropic halftone image, the number of minority pixels in these sectors are roughly equal. This DDF function can be plotted as histogram on polar coordinates.

3.6.3 Residual Variance

For most sample based VC, such as probabilistic VC and random grid VC, the target image is not a blue noise pattern. Some algorithms may also produce near white noise pattern, with clusters of white pixels and black pixels. After processing by the HVS model, which is essentially a low-pass system, a significant amount of noise is left within the frequency band to which the HVS is sensitive. For blue noise pattern, only a small amount of noise is left within this frequency band.

So, if the testing image is a constant image, then one may use the residual noise power after applying the HVS as a measure of perceptual quality. This is introduced in [12] and is referred to as *residual variance*.

Let the target image be $\hat{y}[\mathbf{n}]$, and the HVS be simply modeled as a Gaussian lowpass filter $G_\sigma[\mathbf{n}]$ with standard deviation σ . Then the noise that is ‘visible’ to the HVS is

$$\bar{y}[\mathbf{n}] = \hat{y}[\mathbf{n}] \otimes G_\sigma[\mathbf{n}]. \quad (3.29)$$

The residual variance is calculated as variance of $\bar{y}[\mathbf{n}]$:

$$V_R = \mathbb{E} \{ (\bar{y}[\mathbf{n}] - \mathbb{E}(\bar{y}[\mathbf{n}]))^2 \}. \quad (3.30)$$

Obviously, if $\hat{y}[\mathbf{n}]$ is a blue noise pattern, then $\bar{y}[\mathbf{n}]$ is close to zero. So the smaller the V_R , the better perceptual quality.

Note that the residual variance is not the same as the MSE measure used by DBS algorithm. MSE is calculated between smoothed grayscale image and halftone image. But residual variance is calculated between smoothed halftone image and its mean. Thus residual variance is not a fidelity measure, and is suitable only when one uses constant test images.

Compared to RAPSD, the residual variance is a summary of the quality for each gray level. So a 2D plot of the residual variance can help us to explore the qualities for all possible gray levels, while RAPSD needs 3D plotting. But one should note that residual variance only gives a summary of the amount of noise in low-frequency band, so it cannot tell the distribution of the noise over the whole frequency band.

3.7 Summary

This chapter briefly review some important and classical halftoning techniques that are widely used in VC for grayscale images, including simple bi-level quantization, ordered dithering, error diffusion and DBS. All these types of halftoning techniques are used by authors of VC research papers. Among them, error diffusion keeps a good balance between computational efficiency and image quality and is the mostly adopted approach. A linear model for error diffusion is also reviewed and is shown to be useful to understand the sharpening effect in halftone image and tone reproduction property. This later property will be used to prove the contrast condition in AbS-based VC. Several quality measures, including HPSNR, RAPSD, DDF are reviewed for later chapters.

References

1. Analoui, M., Allebach, J.P.: Model-based halftoning using direct binary search. In: Proceeding of SPIE 1666, vol. 1666 (1992)
2. Bayer, B.E.: An optimum method for two-level rendition of continuous-tone pictures. In: IEEE International Conference on Communications (1973)
3. Floyd, R.W., Steinberg, L.: An adaptive algorithm for spatial gray-scale. Proc. Soc. Inf. Disp. **17**(2), 75–78 (1976)
4. Guo, J.M., Chang, J.Y., Liu, Y.F., Lai, G.H., Lee, J.D.: Tone-replacement error diffusion for multitoning. IEEE Trans. Image Process. **24**(11), 4312–4321 (2015)
5. Jarvis, J.F., Judice, C.N., Ninke, W.H.: A survey of techniques for the display of continuous tone pictures on bilevel displays. Comput. Graph. Image Process. **5**(1), 13–40 (1976)

6. Jayant, N., Noll, P.: Digital Coding of Waveforms: Principles and Applications to Speech and Video. Prentice-Hall signal processing series, Prentice-Hall (1984)
7. Kite, T.D., Evans, B.L., Bovik, A.C.: Modeling and quality assessment of halftoning by error diffusion. *IEEE Trans. Image Process.* **9**(5), 909–922 (2000)
8. Knox, K.T.: Error image in error diffusion. *Proc. SPIE* **1657**, 268–279 (1992)
9. Lau, D., Arce, G.: Modern Digital Halftoning, 2nd edn. Taylor & Francis, Boca Raton, FL (2001)
10. Ulichney, R.A.: Dithering with blue noise. *Proc. IEEE* **76**(1), 56–79 (1988)
11. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**(4), 600–612 (2004)
12. Yan, B., Xiang, Y., Hua, G.: Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (AbS) approach. *IEEE Trans. Image Process.* **28**(2), 896–911 (2019)

Chapter 4

Improving Visual Quality for Share Images



4.1 Binary Visual Cryptography with Meaningful Shares: Extended VC

Ordinary VC produces meaningless and noise-like shares, which makes it difficult to manage the shares if more than one shares need to be stored. It is difficult for the share manager to determine which share belongs to which secret image. Furthermore, during transmission, these meaningless shares may arouse the suspicions from potential attackers.

An extended visual cryptography (extended VC), as proposed in [8], produces meaningful shares (also called shadows). By using a tag image as the cover image, the meaningful shares are easier to store or manage. Furthermore, the meaningful shares may also act as a steganographic mechanism in visual cryptography and try to hide the fact that the transmitted image is a share from visual cryptography. As a result, the shares in extended VC are less likely to arouse suspicion from attackers if their quality is high enough. For this purpose, the share should be perceptually as close to the cover image as possible.

4.1.1 Basic Extended VC

A (2,2)-threshold extended VC was first introduced in Naor and Shamir's 1994 seminal paper [8]. Let S be a secret image and C_1 and C_2 be two cover images. The secret S is split into the two shares B_1 and B_2 , where B_i has the appearance of corresponding cover image C_i , respectively. In Naor and Shamir's scheme, each secret pixel is expanded to a 2×2 block. Let's focus on one secret pixel $S[i, j]$. Let (c_1, c_2) be the two corresponding pixels on the cover images. To encode a white secret pixel $S[i, j] = 0$, the encoder uses one of the following four basis matrices:

$$\begin{aligned}
\mathbf{M}_{00}^0 &= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, & \mathbf{M}_{01}^0 &= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \\
\mathbf{M}_{10}^0 &= \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, & \mathbf{M}_{11}^0 &= \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix},
\end{aligned} \tag{4.1}$$

where the basis matrix $\mathbf{M}_{c_1 c_2}^s$ is for the case when the secret pixel equals to s and the two corresponding cover pixels equal to c_1 and c_2 , respectively. Similarly, to encode a black secret pixel $S[i, j] = 1$, the encoder uses one of the following four basis matrices:

$$\begin{aligned}
\mathbf{M}_{00}^1 &= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, & \mathbf{M}_{01}^1 &= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \\
\mathbf{M}_{10}^1 &= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, & \mathbf{M}_{11}^1 &= \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix},
\end{aligned} \tag{4.2}$$

After a basis matrix is chosen by the combination of $S[i, j] = s$, $C_1[i, j] = c_1$ and $C_2[i, j] = c_2$, the columns of this matrix are permuted. Then, the first row is re-organized into a 2×2 block and assigned to share \mathbf{B}_1 , while the second row is re-organized into a 2×2 block and assigned to share \mathbf{B}_2 . By stacking two rows of matrices \mathbf{M}_{c_1, c_2}^0 , the blackness is 3, while stacking two rows of matrices \mathbf{M}_{c_1, c_2}^1 produces blackness 4. So, the target image will reveal the secret image. Furthermore, for $c_i = 1$, the i -th row of corresponding \mathbf{M}^s contains three black pixels, while for $c_i = 0$, the i -th row of corresponding \mathbf{M}^s contains two black pixels. So, the shares will resemble corresponding cover images.

Using the images in Fig. 4.1 as secret and cover images, we get the experimental result in Fig. 4.2. While this algorithm shows acceptable result for binary cover images, it is not directly applicable to halftone image.

The binary secret image and two halftone cover images are shown in Fig. 4.3. The two share images and recovered target image are shown in Fig. 4.4. The share images show reduced contrast. Some low contrast details are also lost.



Fig. 4.1 Binary test images for extended VC. **a** Secret image. **b** Cover image 1. **c** Cover image 2

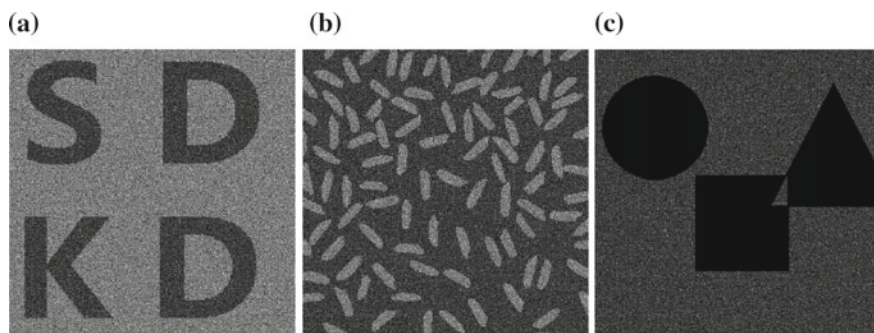


Fig. 4.2 Experimental result for extended VC using Naor and Shamir's scheme. **a** Share 1. **b** Share 2. **c** Target image



Fig. 4.3 Binary and halftone test images for extended VC. **a** Secret image. **b** Cover image 1. **c** Cover image 2

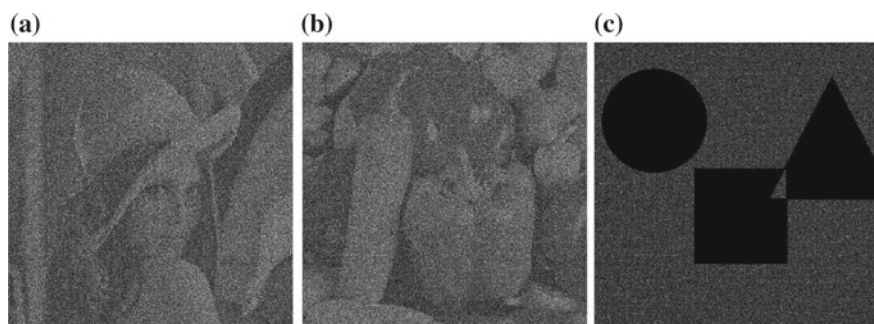


Fig. 4.4 Experimental result for extended VC using Naor and Shamir's scheme and halftone cover images. **a** Share 1. **b** Share 2. **c** Target image

4.1.2 User-Friendly Random Grid

Naor and Shamir's scheme is not size-invariant, because the size of the shares is four times of the size of the secret and cover images. Chen proposed a random grid based scheme, friendly random grid visual secret sharing (FRGVSS), which is size-invariant [1].

Note that in an extended VC, each share has to carry two type of pixels: the secret pixels and the cover pixels. The FRGVSS algorithm uses a probabilistic approach to determine which pixel on a share is to carry the secret pixel and which pixel to carry the corresponding cover pixel. With probability β , the share pixel carries a secret pixel. When a secret pixel is chosen, an ordinary random grid algorithm is used to generate two share pixels on two share images. With probabilities $\frac{1-\beta}{2}$, a cover pixel is chosen (assuming (2,2)-threshold VC), then the corresponding share pixel will reflect this cover pixel, and the other share will generate a black pixel to ensure that the stacking result is black.

The parameter $0 < \beta < 1$ controls the tradeoff between the quality of the share images and the quality of the target image. Using a small β , only a small number of pixels on a share are used to carrier the secret pixel, and a large number of pixels are used to carry the corresponding cover pixels. So, we may expect that the share images have a good quality (higher fidelity with the cover image), while the target image has a worse quality.

Chen's algorithm is summarized in Algorithm 10.

The experimental results for share images and target images are shown in Fig. 4.5 and Fig. 4.6, respectively, for different β . Obviously, we observe improved quality for the share images for a smaller β . But this comes with the price of a lower contrast in target image.

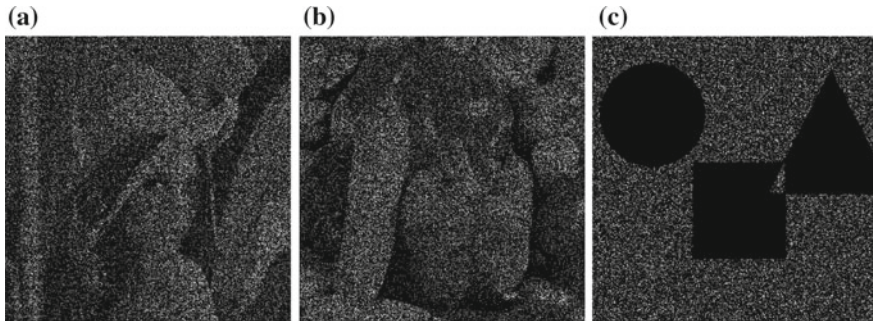


Fig. 4.5 Experimental result for extended VC using FRGVSS and halftone cover images ($\beta = 1/3$). **a** Share 1. **b** Share 2. **c** Target image

Algorithm 10 FRGVSS: Friendly Random-Grid Based Visual Secret Sharing**Input:**

Halftone cover images: $C_1 \in \mathbb{Z}_2^{M_c \times N_c}$, $C_2 \in \mathbb{Z}_2^{M_c \times N_c}$; {The set $\mathbb{Z}_2 \triangleq \{0, 1\}$ }

Binary secret image: $S \in \mathbb{Z}_2^{M_c \times N_c}$;

Parameter β ;

Output:

Share images: $B_1 \in \mathbb{Z}_2^{M_c \times N_c}$, $B_2 \in \mathbb{Z}_2^{M_c \times N_c}$;

```

1: for  $i \leftarrow 1$  to  $M_c$  do
2:   for  $j \leftarrow 1$  to  $N_c$  do
3:      $k \leftarrow \text{RandomSelect} \left( \{0, 1, 2\}, \left\{ \beta, \frac{1-\beta}{2}, \frac{1-\beta}{2} \right\} \right)$ 
4:     if  $k = 1$  then
5:        $B_1[i, j] \leftarrow C_1[i, j]$ 
6:        $B_2[i, j] \leftarrow 1$ 
7:     else if  $k = 2$  then
8:        $B_1[i, j] \leftarrow 1$ 
9:        $B_2[i, j] \leftarrow C_2[i, j]$ 
10:    else
11:       $B_1[i, j] \leftarrow \text{RandomSelect} \left( \{0, 1\}, \left\{ \frac{1}{2}, \frac{1}{2} \right\} \right)$ 
12:      if  $S[i, j] = 1$  then
13:         $B_2[i, j] \leftarrow 1 - B_1[i, j]$ 
14:      else
15:         $B_2[i, j] \leftarrow B_1[i, j]$ 
16:      end if
17:    end if
18:  end for
19: end for

```

{RandomSelect(A, B): randomly select an element from the set A with the corresponding probabilities listed in set B .}

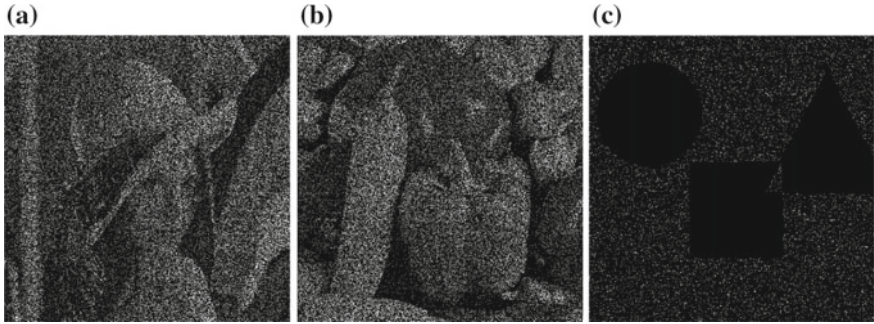


Fig. 4.6 Experimental result for extended VC using FRGVSS and halftone cover images ($\beta = 1/8$). **a** Share 1. **b** Share 2. **c** Target image

4.1.3 Pixel Swapping Algorithm

Another effort in improving the quality of share image is Lou's pixel swapping algorithm [7]. This algorithm is block-based and tries to re-arrange the locations

of black pixels within a small block to make the target block as close to the secret block as possible. If the secret block is a black block (i.e., containing only black pixels), then among the two corresponding share blocks, we choose the one having more black pixels to modify. The black pixels in this block is re-arranged, so that the stacking result contains as many black pixels as possible. If the secret block is a white block (i.e., containing only white pixels), then among the two corresponding share blocks, we choose the one having more black pixels to modify. The black pixels in this block is re-arranged, so that the stacking result contains as many white pixels as possible. Obviously, this re-arrangement only guarantees best-effort approximation, so the target image is only partially reconstructed.

The pixels are only moved around in a small block and the proportion of black pixels is not changed, so the share image has a good visual quality. However, only the secret block with all black pixels and the secret block with all white pixels are approximated with best effort. From the stacking result, we may see interferences from the cover image. Lou's algorithm is summarized in Algorithm 11.

Algorithm 11 Visual cryptography based on pixel swapping [7]

Input:

Halftone cover images: $\mathbf{C}_1 \in \mathbb{Z}_2^{M_c \times N_c}$, $\mathbf{C}_2 \in \mathbb{Z}_2^{M_c \times N_c}$; {The set $\mathbb{Z}_2 \triangleq \{0, 1\}$ }

Binary secret image: $\mathbf{S} \in \mathbb{Z}_2^{M_c \times N_c}$;

Output:

Share images: $\mathbf{B}_1 \in \mathbb{Z}_2^{M_c \times N_c}$, $\mathbf{B}_2 \in \mathbb{Z}_2^{M_c \times N_c}$;

```

1: for  $i \leftarrow 1$  to  $\lfloor \frac{M_c}{2} \rfloor$  do
2:   for  $j \leftarrow 1$  to  $\lfloor \frac{N_c}{2} \rfloor$  do
3:      $\mathbf{C}_{i,j}^1 \leftarrow$  The  $(i, j)$ -th block of  $\mathbf{C}_1$ ;
4:      $\mathbf{C}_{i,j}^2 \leftarrow$  The  $(i, j)$ -th block of  $\mathbf{C}_2$ ;
5:     if  $\sum_{k=1}^2 \sum_{\ell=1}^2 \mathbf{C}_{i,j}^1[k, \ell] > \sum_{k=1}^2 \sum_{\ell=1}^2 \mathbf{C}_{i,j}^2[k, \ell]$  then
6:        $\mathbf{m} \leftarrow \mathbf{C}_{i,j}^1$ ;
7:        $\mathbf{f} \leftarrow \mathbf{C}_{i,j}^2$ ;
8:     else
9:        $\mathbf{m} \leftarrow \mathbf{C}_{i,j}^2$ ;
10:       $\mathbf{f} \leftarrow \mathbf{C}_{i,j}^1$ ;
11:     end if
12:     if  $\sum_{k=1}^2 \sum_{\ell=1}^2 S[k, \ell] = 4$  then
13:       Swap pixels in  $\mathbf{m}$  to maximize  $\sum_{k=1}^2 \sum_{\ell=1}^2 \bar{m}[k, \ell] \vee \bar{f}[k, \ell]$ ;
14:     else if  $\sum_{k=1}^2 \sum_{\ell=1}^2 S[k, \ell] = 0$  then
15:       Swap pixels in  $\mathbf{m}$  to minimize  $\sum_{k=1}^2 \sum_{\ell=1}^2 \bar{m}[k, \ell] \vee \bar{f}[k, \ell]$ ;
16:     end if
17:     Fill  $\mathbf{m}$  and  $\mathbf{f}$  into the corresponding positions in share images  $\mathbf{B}_1$  and  $\mathbf{B}_2$ .
18:   end for
19: end for
```

The testing result using Lou's algorithm and halftone images is shown in Fig. 4.7. As can be observed, the share images are visually very close to the corresponding halftone images in Fig. 4.3, thanks to the local pixel swapping operation on halftone

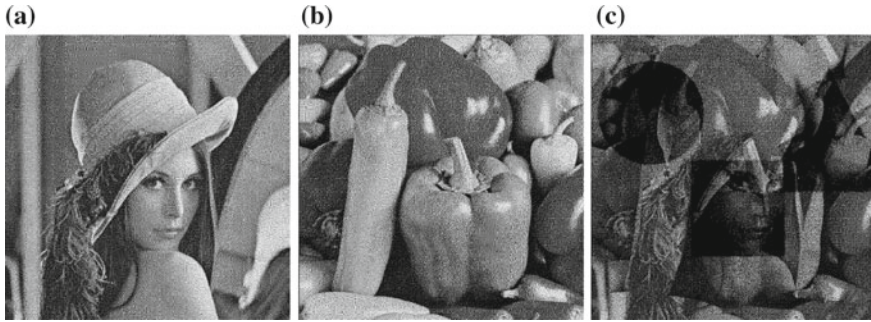


Fig. 4.7 Experimental result for extended VC using Lou's pixel swapping algorithm and halftone cover images. **a** Share 1. **b** Share 2. **c** Target image

shares. However, the target image is only partially reproduced. We see severe interferences from the cover images. So, Lou's algorithm should be used when the user concerns more about the quality of the share images than the target image.

Naor and Shamir's algorithm, Chen's algorithm and Lou's pixel swapping use binary or halftone image as cover image. The loss of details of the cover image due to VC is not compensated by neighboring pixels. By integrating an error diffusion halftoning process into the VC encoding, it is possible to compensate for the loss of image quality from VC encoding.

4.2 Error-Diffusion Based Scheme

In this section, we introduce a constrained error diffusion approach to improve the quality of share images [12, 15]. This algorithm uses small blocks as processing units. In order to produce a meaningful share and to convey secret on shares, the pixels on a share block should be classified into three types:

- **Secret Information Pixels (SIPs)**. These pixels are used to carry the information of the secret image, and are encoded according to ordinary binary VC encoding.
- **Cover Information Pixels (CIPs)**. These pixels are used to carry the information of the cover image.
- **Auxiliary Black Pixels (ABPs)**. These pixels are set as black, so that after stacking with another share, the resultant target pixel is black. The CIPs and ABPs are called non-SIPs, since they don't convey the secret information.

The reason for the above classification is that, for an extended VC, two contradicting goals must be met. First, each share image should retain reasonably high visual quality, and should be perceptually similar to the corresponding cover image. Second, the recovered secret image and watermark image should be free from interference from the shares.

The first goal implies that only a small number of pixels on a share can be used to carry the secret, while other pixels should be used to carry the information of cover image. So, those pixels that are used to carry secret/watermark information are SIPs, and all others are non-SIPs [12]. To meet the second goal, non-SIPs should be stacked to be black. We must guarantee that for the two corresponding pixels on the two shares, at least one is black. This can be done by using the ABPs. The positions of the ABPs on the two shares are complementary such that when stacking, the resultant pixels are black. So if a pixel is ABP on one share, then the corresponding pixel on the other share should be CIP. For this reason, determining the positions of ABPs on one share is equivalent to determining the positions of CIPs on the other share. So, we don't need to determine the positions of CIPs.

In the VC encoder, one may first determine the ABPs to meet the second goal, and then encode the secret into the SIPs. Finally, a constrained error diffusion can be adopted to approach the first goal.

4.2.1 SIPs and ABPs

First, we need to determine the positions of SIPs and ABPs. Let the size of the cover images be $M_c \times N_c$. We encode 1 bit of secret image (i.e. 1 pixel) into a $Q \times Q$ block in share images. So if the size of the share image is $M_s \times N_s$, then it must satisfy: $M_s = \lfloor \frac{M_c}{Q} \rfloor$ and $N_s = \lfloor \frac{N_c}{Q} \rfloor$, where $\lfloor x \rfloor$ returns the nearest integer towards $-\infty$.

In [12], the positions of the ABPs and SIPs are determined using a multi-tone error diffusion. Here, we adopt a random approach. Using this random approach, we can guarantee that there are exactly the required number of SIPs and ABPs in each block, while the approach in [12] needs post-processing to guarantee this.

Let the number of SIPs in each block be γ , and the number of ABPs in each block for each share be β . Then we collect all the SIPs positions into a set \mathbf{S} , i.e., $\mathbf{S} \triangleq \{(i, j); (i, j) \text{ is a SIP position}\}$. Similarly we can collect all the ABPs in share 1 into a set \mathbf{A}_1 and all ABPs in share 2 into the set \mathbf{A}_2 .

In each block, we must guarantee that the stacking of non-SIPs are black pixels, then the ABPs of each share must have at least more than half of the non-SIPs. Since there are totally $Q^2 - \gamma$ non-SIPs in each block, the minimum number of ABPs should be $\beta = \lceil \frac{Q^2 - \gamma}{2} \rceil$.

First, we arrange the pixels in a block using linear indexing. For example see the indexing in Fig. 4.8a. Then let the vector \mathbf{p} contain a random permutation of integer numbers from 1 to Q^2 . For example, $\mathbf{p} = (1, 6, 2, 8, 4, 7, 9, 3, 5)$ for $Q = 3$. This permutation should vary from block to block, so that the relative positions of SIPs and ABPs in different blocks are different. Then as shown in Fig. 4.8a, take the first γ indices as the indices for SIPs. The ABPs for share 1 is taken as $(p_{\gamma+1}, \dots, p_{\gamma+\beta})$, while the ABPs for share 2 is taken as $(p_{\gamma+\beta+1}, \dots, p_{Q^2})$ if $Q^2 - \gamma$ is even, and as $(p_{\gamma+\beta+1}, \dots, p_{Q^2}, p_{\gamma+1})$ if $Q^2 - \gamma$ is odd. Since the ABPs will be filled with

black pixels, the stacking of share 1 and share 2 is guaranteed to be black for non-SIPs. After allocating the SIPs and ABPs for both shares, we map the position to two-dimensional index within the $Q \times Q$ block. This is shown in Fig. 4.8b.

Due to the random permutation, the above allocation of SIPs and ABPs is random in nature. Even though this is not optimal in the sense that the SIPs and the ABPs are not maximally separated from each other, but this algorithm still can provide satisfying result, as will be shown in experimental results. Besides, using this algorithm, exactly γ SIPs and β ABPs are contained in each block. The method in [12] utilized a multitone process to help to allocate the SIPs and ABPs. But the multitone algorithm only guarantee that globally a fraction of γ/Q^2 pixels are SIPs. So the number of SIPs within a $Q \times Q$ block may not be exactly γ . Thus, post-processing is needed to guarantee that exactly γ SIPs and β ABPs are contained in each block. More importantly, from the security point of view, deterministic allocation of the position for SIPs and ABPs is not secure. If an attacker has the knowledge of the positions of SIPs and ABPs, then he can use it to prevent the recovery of the secret image. On the contrary, using random allocation, the locations of the SIPs and ABPs are unknown to the attacker if he doesn't know the seed or key.

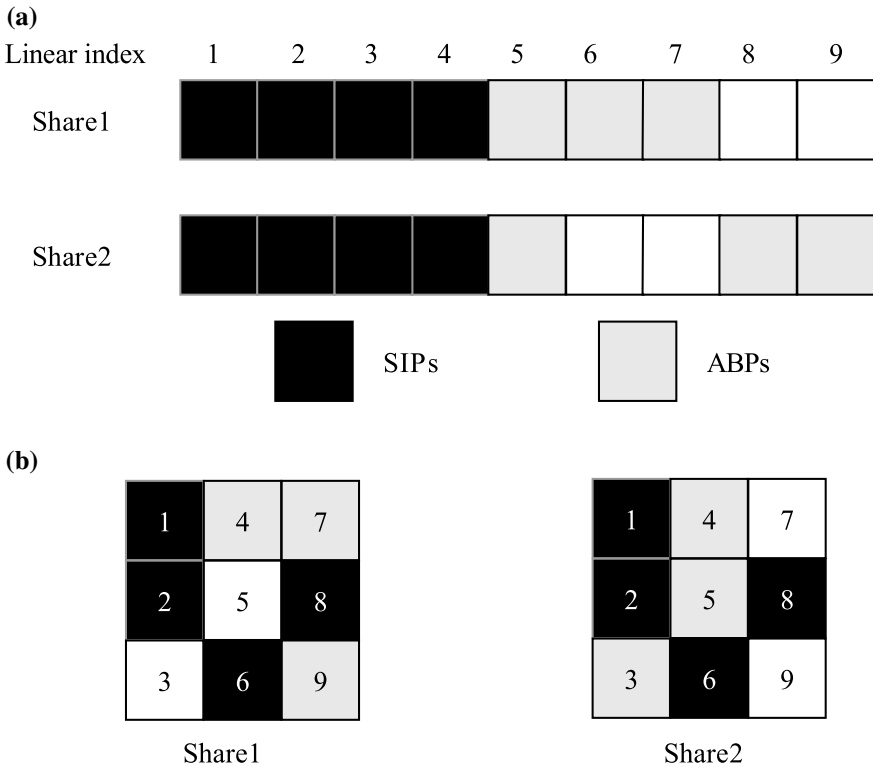


Fig. 4.8 Determining the SIPs and ABPs positions ($Q = 3, \gamma = 4, \beta = 3$)

4.2.2 Constrained Error Diffusion

After filling in the SIPs and the ABPs, the content of the secret image can be recovered correctly. But the shares are still meaningless. To encode the cover images into the shares, a constrained error diffusion halftoning method as described in [12] can be used. Since the SIPs and ABPs are fixed now, the error diffusion is a constrained one: some pixels are not allowed to be changed by the diffusion process. This is illustrated in Fig. 4.9, where the error is not allowed to diffuse to the SIPs and the ABPs.

As with ordinary error diffusion, the pixels of the gray image are visited using raster scanning order. Let the function $F_q[i, j]$, $q = 1, 2$ be a indicator function, indicating whether the current pixel $[i, j]$ is allowed to change or not:

$$F_q[i, j] = \begin{cases} 0, & \text{if } [i, j] \in \mathbf{S} \cup \mathbf{A}_q; \\ 1, & \text{otherwise.} \end{cases}$$

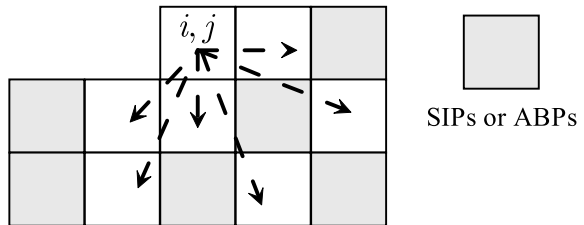
If $h_{i,j}[k, \ell]$ is a diffusion coefficient for the diffusion mask located at pixel $[i, j]$, then in constrained diffusion, the modified diffusion coefficient is $h_{i,j}[k, \ell] = h_{i,j}[k, \ell] \cdot F_q[k, \ell]$. Namely, the diffusion coefficients to the SIPs and the ABPs are set to be zero. Since the SIPs and ABPs are fixed now, the quantization process in the original halftoning is modified to be

$$s_q[i, j] = (1 - F_q[i, j]) \cdot s_q[i, j] + F_q[i, j] \cdot \mathcal{Q}(c_q[i, j]),$$

where $\mathcal{Q}(x)$ quantizes x using a two level quantizer whose threshold equals to 127, for a 8-bits quantized image. This quantizer skips the SIPs and ABPs.

The testing results for a (2, 2)-threshold scheme are shown in Fig. 4.10, using parameters $Q = 4$, $\gamma = 4$, and $\beta = 6$. The target image in Fig. 4.10c doesn't have interference from the cover images. Furthermore, the share images have much better appearance than the result of FRGVSS, as shown in Figs. 4.5 and 4.6.

Fig. 4.9 Constrained error diffusion. The error at position $[i, j]$ is allowed only to diffused to CIPs, and is not allowed to diffuse to SIPs or ABPs



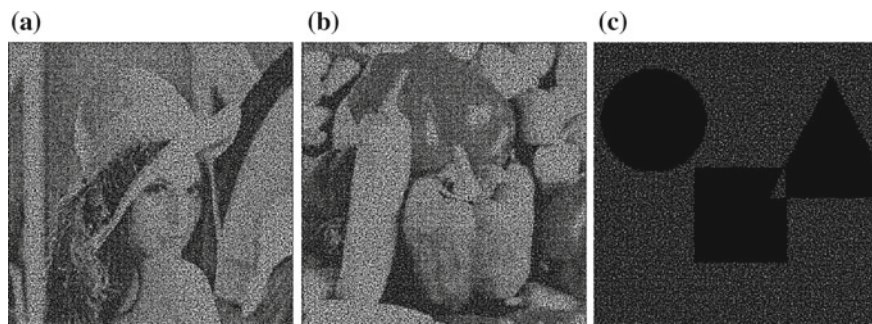


Fig. 4.10 Experimental result for extended VC using error diffusion and halftone cover images. **a** Share 1. **b** Share 2. **c** Target image

4.3 Extended VC with a Hidden Watermark

Authentication¹ of the shares is necessary when facing the replacement attack [2]. If an attacker is able to intercept one copy of the shares, say share 1, and hence can produce a fake share 2 according to his fake secret. Then he can send the fake share to the receiver. If the receiver stack the share 1 with the fake share 2, then the fake secret will be revealed. As a countermeasure to this attacking scenario, one can embed another secret image, called a watermark, to serve as an authentication mechanism [3, 4, 10, 14].

In [2], Fang et al. constructed a watermark embedding algorithm to VC. But Fang's algorithm produces meaningless shares and the pixel expansion is 4 [2]. Recently, Huang improved Fang's algorithm such that there is no pixel expansion in VC [5]. But Huang's algorithm still generates meaningless shares. Liu et al. proposed the embedded extended visual cryptography, which is combined with the halftoning process [6]. But the pixel expansion is not 1 and there is no watermark embedded. Simultaneous embedding of both the secret and watermark into two meaningful shares imposes too many constrains on the VC algorithm. Hence its design is more complicated. In [7], Lou et al. proposes such a system. But unfortunately, on the recovered secret and watermark images, there are severe interferences from the share images. One need to carefully identify the secret/watermark image, especially when the grayscale of the share is close to the grayscale of the secret or watermark images.

For an extended VC with a hidden watermark, we must satisfy the following requirements: (1) The input cover image is a grayscale image; (2) The shares are meaningful; (3) There is no pixel expansion. The size of the share images is the same as the cover image; (4) No computation is needed during decoding; (5) Both the secret and the watermark are encoded; (6) There is no interference from shares on the stacking results.

¹Adapted by permission from Springer Nature: Springer, Multimedia Tools and Applications, Vol. 75, No. 18, Size-invariant extended visual cryptography with embedded watermark based on error diffusion, Bin Yan et al. ©, 2015.

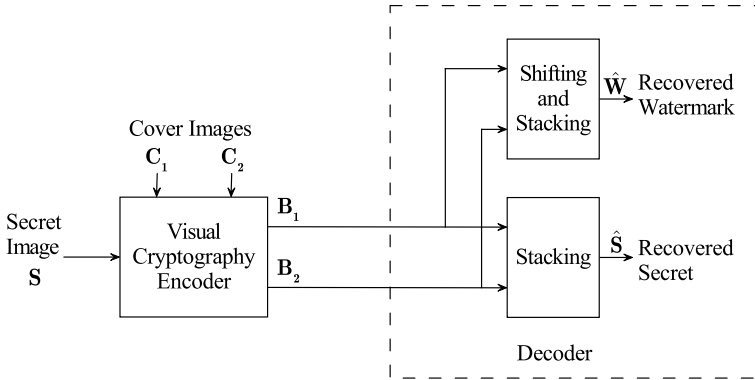


Fig. 4.11 Overall structure of the system

This section introduces one of our recent works on this topic [13]. The structure of the overall system is shown in Fig. 4.11. A secret image \mathbf{S} is encoded into two shares \mathbf{B}_1 and \mathbf{B}_2 such that the appearance of the shares looks similar to the two cover images \mathbf{C}_1 and \mathbf{C}_2 respectively. Each share is transmitted independently through a different channel. After receiving both shares, the receiver can stack them to recover an image $\hat{\mathbf{S}}$ that has the same information content as \mathbf{S} . In addition, after stacking \mathbf{B}_2 on a shifted \mathbf{B}_1 , the recovered image $\hat{\mathbf{W}}$ contains the same information content as \mathbf{W} . The cover images are grayscale images, while both the secret image and the watermark image are binary.

4.3.1 Simultaneous Encoding of Secret and Watermark

The positions of SIPs and ABPs can be determined as in Sect. 4.2.1. After that, we get the sets \mathbf{S} , \mathbf{A}_1 and \mathbf{A}_2 . In this section, we describe the encoding of both the secret image and watermark into the SIPs positions. This process is based on the algorithm proposed in [2]. But the algorithm in [2] is constructed for meaningless shares. For meaningful shares, special care must be taken to the SIPs and the ABPs.

In order to embed the secret image and the watermark simultaneously, the positions of SIPs and ABPs must be modified. Remember that the SIPs and ABPs for different block are different. So, when stacking the shares, the SIPs for the corresponding pixels must be the same, and the ABPs for the corresponding pixels must be complementary. To help the description of the proposed algorithm, we introduce the following indicator matrices.

$$\begin{aligned}
 \mathbf{S} &= [s[i, j] \in \{0, 1\}]_{M_c \times N_c}, \quad s[i, j] = 1 \text{ if } (i, j) \in \mathbf{S} \\
 \mathbf{A}_1 &= [a_1[i, j] \in \{0, 1\}]_{M_c \times N_c}, \quad a_1[i, j] = 1 \text{ if } (i, j) \in \mathbf{A}_1 \\
 \mathbf{A}_2 &= [a_2[i, j] \in \{0, 1\}]_{M_c \times N_c}, \quad a_2[i, j] = 1 \text{ if } (i, j) \in \mathbf{A}_2
 \end{aligned}$$

Each element in \mathbf{S} indicates whether the current position is a SIP or not. Similarly, each element in \mathbf{A}_1 indicates whether the current position in share 1 is an ABP or not.

The following regions are also defined on the two shares, as shown in Fig. 4.12. Let d be the amount of shift of share 1 when stacking with share 2 to recover the watermark. Here we consider only horizontal shift and restrict its amount to be $d \geq \frac{N_c}{2}$. Define

$$\begin{aligned} \mathbf{L}_1 &= \{(i, j); 1 \leq j \leq N_c - d\}, \quad \mathbf{R}_1 = \{(i, j); N_c - d < j \leq N_c\}, \\ \mathbf{L}_2 &= \{(i, j); 1 \leq j \leq d\}, \quad \mathbf{R}_2 = \{(i, j); d < j \leq N_c\}. \end{aligned}$$

First, the positions for the SIPs and ABPs for the two shares are generated according to Sect. 4.2.1. Then we copy the position of SIPs in area \mathbf{R}_2 of share 2 to the position of SIPs in area \mathbf{L}_1 of share 1, i.e.,

$$s[i, j] \leftarrow s[i, j - d + 1], \quad d \leq j \leq N_c. \quad (4.3)$$

So the SIPs positions in region \mathbf{R}_2 and \mathbf{L}_1 are the same. This will ensure that after stacking the \mathbf{R}_2 and \mathbf{L}_1 , the watermark $\hat{\mathbf{W}}$ is revealed. To ensure that the ABPs on the two shares are complementary, we have to copy the ABPs in \mathbf{L}_1 to the ABPs in \mathbf{R}_2 on each share:

$$a_2[i, j] \leftarrow a_2[i, j - d + 1], \quad d \leq j \leq N_c; \quad (4.4)$$

$$a_1[i, j] \leftarrow a_1[i, j - d + 1], \quad d \leq j \leq N_c. \quad (4.5)$$

The case when $d < \frac{N_c}{2}$ and non-zero vertical shift can also be constructed similarly, but is more involved. So we omit it here. After the above preprocessing, when

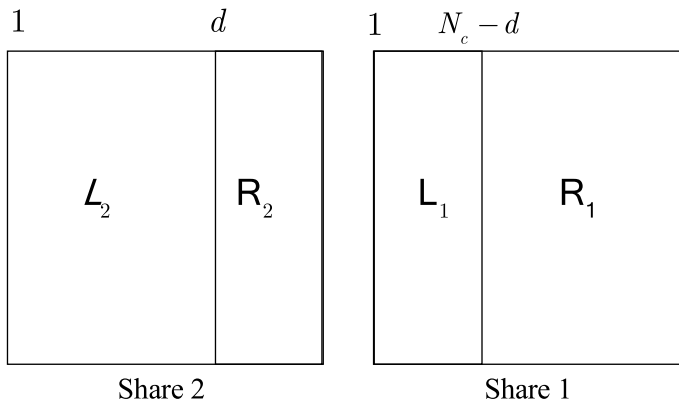


Fig. 4.12 Four regions defined on the two shares. In the figure, two regions are plotted on share/cover 1 and two on share/cover 2, but the four regions are defined for each share/cover image

stacking the two shares, the SIPs coincide and the ABPs are complementary for the same position. So now we are ready to embed the secret image and the watermark.

Suppose that the basis matrices for bit ‘0’ and ‘1’ are \mathbf{M}_0 and \mathbf{M}_1 respectively [8]. For example, for a (2, 2) scheme, when $\gamma = 4$, we can use

$$\mathbf{M}_0 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}. \quad (4.6)$$

If a bit ‘0’ is to be encoded, then the matrix \mathbf{M}_0 is chosen. After a random column permutation $\Pi(\cdot)$, the first row is filled into the γ SIPs positions in the current block of share 1, and the second row is filled into the γ SIPs positions in the current block of share 2.

Since now we need to encode both the secret image and watermark into the shares, we have to treat each region differently. This is described by the following steps.

1. First, for region \mathbf{L}_2 on both share 1 and share 2, the SIPs are determined by the corresponding pixels in the secret image. Let $[k, \ell]$ be the index of the blocks, which is also the index of the pixel in the secret image and the watermark image. From the index of the pixels in the shares we can find the block index as $k = \lceil \frac{i}{Q} \rceil$, and $\ell = \lceil \frac{j}{Q} \rceil$. If $s[k, \ell]$ is ‘0’, then we fill in the SIPs with the rows of $\Pi(\mathbf{M}_0)$ in the two shares, where $\Pi(\mathbf{M})$ denotes a random permutation of the columns of the matrix \mathbf{M} . Otherwise, the SIPs are filled with the rows of $\Pi(\mathbf{M}_1)$.
2. Then, the SIPs in region \mathbf{R}_2 of share 2 are determined by the watermark image \mathbf{W} and the SIPs in region \mathbf{L}_1 of share 1. If the watermark bit $w[k, \ell]$ is ‘0’, then we fill the complement of the SIPs in \mathbf{L}_1 to region \mathbf{R}_2 . Otherwise, the value of the SIPs of share 1 are copied to share 2. This step encodes the watermark image into the shares, so that stacking \mathbf{L}_1 and \mathbf{R}_2 reveals the watermark.
3. Finally, we determine the SIPs in region \mathbf{R}_2 in share 1 from the SIPs from the same region in share 2 and the secret image. If the secret image bit $s[k, \ell]$ is ‘0’, then we fill the complement of the SIPs in region \mathbf{R}_2 in share 2 to region \mathbf{R}_2 in share 1. Otherwise, the SIPs of share 2 is copied to share 1.

After the above steps, the SIPs and the ABPs of the two shares are determined.

4.3.2 Extension to $N > 2$ Shares

This algorithm can be extended to more than two shares. The baseline algorithm is designed to generate $N \geq 2$ shares [12], and our modification of the original algorithm, i.e., the random allocation of SIPs and ABPs, doesn’t change the process of encoding the secret, so our algorithm can also be extended to $N > 2$ shares. Let’s use $N = 3$ as an instance and assume that the horizontal shift is $d = \frac{M_c}{2}$. First,

following the 3-out-of-3 scheme as described in [12], we can embed the left half of the secret image, i.e., L_S , into L_1 , L_2 and L_3 . Then we stack share 2 and share 3 to obtain a composite share. The left half of this composite share is already determined, so we can use the watermark image as a ‘secret’ image and modify R_1 , so that the stacking of R_1 and L reveals the watermark image. Finally, we can embed the right half of the secret image, i.e., R_S , to the right half of composite share by modifying R_2 and R_3 . As the result of the above embedding processing, when stacking the three shares, the secret image is revealed. When stacking share 1 with shifted share 2 and share 3, the watermark image is revealed.

4.3.3 Constrained Error Diffusion

After encoding the secret image and watermark, the constrained error diffusion as presented in Sect. 4.2.2 should be used to encode the cover pixels to CIPs.

4.3.4 Experimental Test

Two types of quality measures should be considered in the experiments. The quality of the share image should be evaluated by measures for halftone images, such as PSNR (Peak Signal to Noise Ratio) and MSSIM (Mean Structure Similarity) measures. The quality of the target image should be evaluated by measures for binary images, such as global contrast.

To measure the quality of the share images, we use the PSNR measure [9] and the MSSIM measure [11]. Each share image B_k is compared with the corresponding halftone image H_k obtained using ordinary error diffusion. Before calculating the PSNR, each image is smoothed by a Gaussian filter with variance 4 and kernel size 11×11 . Then the PSNR value is calculated from the two filtered images. While the PSNR value measures the tone consistency between the two images, the MSSIM value measures the structure similarity between two images, which includes luminance, contrast and structure correlation. When calculating MSSIM, the default setting as recommended in [11] is adopted.

The following parameters are used in the experiment: $Q = 4$, $\gamma = 2$, $\beta = 7$, $M_c = N_c = 768$ and $d = M_c/2$. The diffusion mask is the Floyd and Steinberg mask. For the fundamental VC scheme, we choose the (2, 2) scheme with pixel expansion 2 [8]. The two cover images along with the secret image and the watermark image are shown in Fig. 4.13.

The two share images are shown in Fig. 4.14. The PSNR values for the two shares are 16.38 and 16.17 respectively. The MSSIM values are 0.2138 and 0.2142 respectively. We also calculated the MSSIM values between the share and halftone image after Gaussian smoothing, they are 0.6723 and 0.6744 respectively.

The stacking result of the two shares is shown in Fig. 4.15a, which reveals the content of the secret image. If we shift the share image in Fig. 4.14a by d and stack it with the share image in Fig. 4.14b, then the content of the watermark image is revealed, as shown in Fig. 4.15b. The results shown in Figs. 4.14 and 4.15 verify the effectiveness of this algorithm.

The perceptual quality of the binary target image and extracted watermark can be evaluated by the contrast. For the $(2, 2)$ scheme, we can find the expression for the contrast as follows. The contrast of the recovered VC secret, as defined in [15], is the relative difference of 1's between the representation of the black and the white pixels. In our system, if the pixel expansion of the basic VC scheme is γ , and the size of the block is $Q \times Q$, then the contrast can be found to be

$$\alpha = \frac{\gamma}{2Q^2}. \quad (4.7)$$

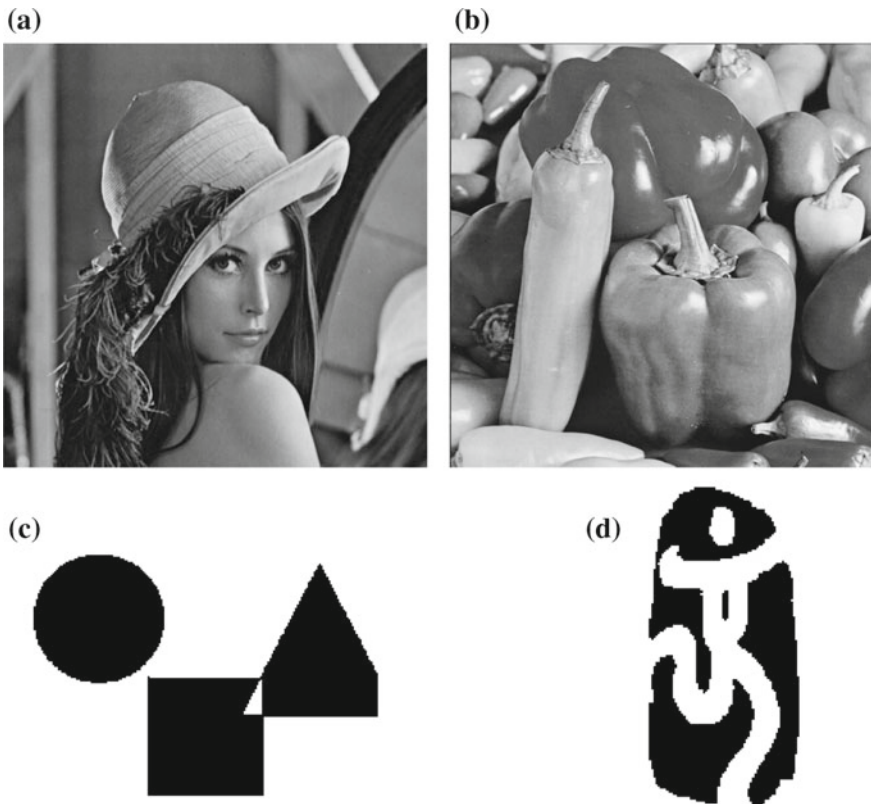


Fig. 4.13 The images used in the experiments. **a** Cover image 1. **b** Cover image 2. **c** Secret image. **d** Watermark Image. The secret image and watermark are resized to meet the size of the cover images

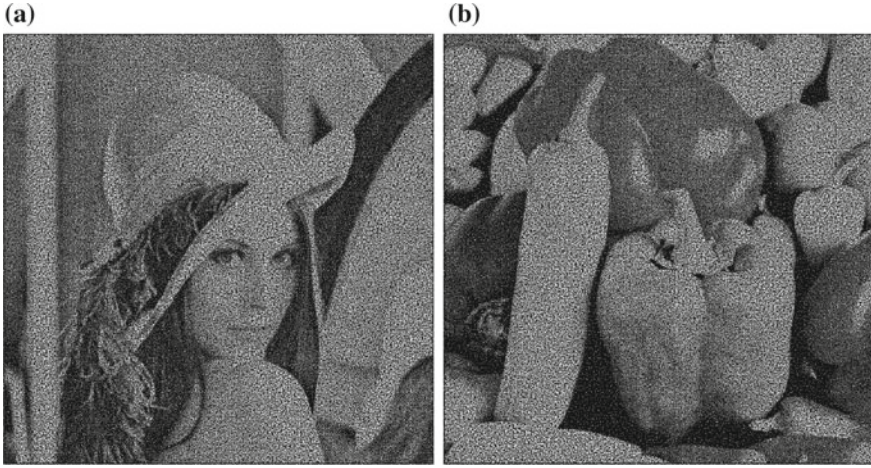


Fig. 4.14 The generated meaningful shares ($\gamma = 2$). **a** Share 1. **b** Share 2

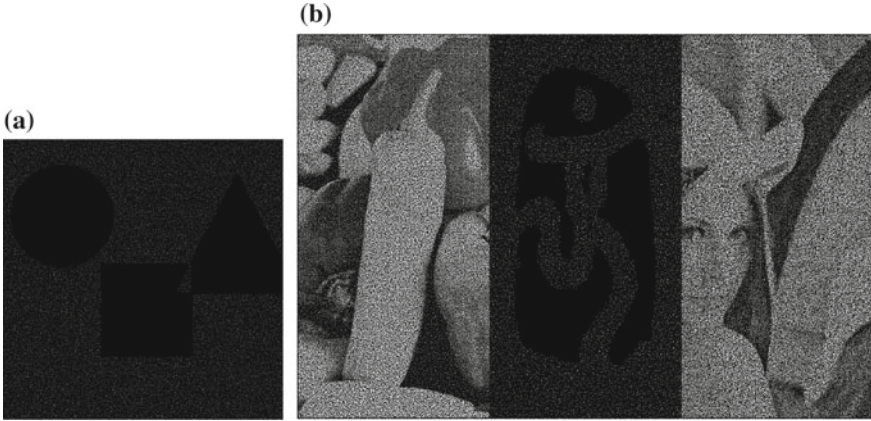


Fig. 4.15 The recovered target image and watermark image. **a** Target image. **b** Watermark image

So if $\gamma = 2$ and $Q = 4$, then the contrast of the recovered secret and watermark is $1/16$. If a contrast of $1/8$ is needed, then for the same $Q = 4$, we should use $\gamma = 4$.

However, the improvement of the contrast of the recovered secret and watermark usually result in decrease of the quality of the cover image. There is tradeoff between quality of the share image and quality of target/watermark image. The quality of the share is inherently limited by the number of non-SIPs and non-ABPs that are reserved to carry the information of the cover image. So, as defined in [12], the quality index is

$$s = \frac{Q^2 - \gamma - \beta}{Q^2}.$$

Then, for our (2, 2) scheme, it is not difficult to show the tradeoff between the contrast α and this quality index s :

$$s = \begin{cases} \frac{1}{2} - \alpha, & \text{if } Q^2 - \gamma \text{ is even;} \\ \frac{1}{2} - \alpha - \frac{1}{Q^2}, & \text{if } Q^2 - \gamma \text{ is odd.} \end{cases} \quad (4.8)$$

The Eq.(4.8) can also be used to find the appropriate design parameters Q , γ and β from the required quality index s and the requirement on the contrast α of the recovered secret and watermark.

4.3.5 Attacks on SIPs and ABPs

If an attacker has access to one of the two shares, say share 1 (i.e., \mathbf{B}_1), then the attacker can modify this share. In addition, if the allocation of SIPs and ABPs are deterministic, then given the cover image, an attacker is able to find the exact locations of the SIPs and the ABPs. The attacker can simply set all the SIPs to '0' (black) and leave the ABPs intact. Let's call this modified share as $\hat{\mathbf{B}}_1$. Then after stacking \mathbf{B}_2 with $\hat{\mathbf{B}}_1$, all the pixels are black. Both the secret and the watermark will be destroyed.

Using the random allocation as in this work, the attacker cannot reproduce the position of the SIPs and the ABPs without knowing the seed of the random permutation. Then the attacker's knowledge about the SIPs is the relative ratio of SIPs within a block, i.e., γ/Q^2 . His best attack is randomly choosing γ pixels in each block and setting them to '0'. But the watermark can still be partially recovered.

4.4 Summary

This chapter reviews algorithms for improving the perceptual quality of share images in extended VC. The classical Naor and Shamir's scheme produces good result for binary cover images, but is not directly applicable to grayscale or halftone cover images. Size expansion of the share images with respect to the cover images is another issue. Chen's FRGVSS solved the issue of size expansion but still produce relatively low quality share images. Lou's pixel swapping algorithm produces good perceptual quality on share images, but the target image has interferences from cover images. Wang's error diffusion algorithm works directly on grayscale image and can compensate for the loss of visual quality due to VC encryption. We also present our improvement by incorporating a hidden watermark for share authentication.

References

1. Chen, T.H., Tsao, K.H.: User-friendly random-grid-based visual secret sharing. *IEEE Trans. Circuits Syst. Video Technol.* **21**, 1693–1703 (2011)
2. Fang, W.P., Lin, J.C.: Visual cryptography with extra ability of hiding confidential data. *J. Electron. Imaging* **15**(2), 1–7 (2006)
3. Huang, H.C., Fang, W.C.: Authenticity preservation with histogram-based reversible data hiding and quadtree concepts. *Sensors* **11**(10), 9717–9731 (2011)
4. Huang, H.C., Chang, F.C., Fang, W.C.: Reversible data hiding with histogram-based difference expansion for QR code applications. *IEEE Trans. Consum. Electron.* **57**(2), 779–787 (2011)
5. Huang, Y.J., Chang, J.D.: Non-expanded visual cryptography scheme with authentication. In: *IEEE 2nd International Symposium on Next-Generation Electronics (ISNE)*, pp. 165–168. IEEE, Taiwan (2013)
6. Liu, F., Wu, C.K.: Embedded extended visual cryptography schemes. *IEEE Trans. Inf. Forensics Secur.* **6**(2), 307–322 (2011)
7. Lou, D.C., Chen, H.H., Wu, H.C., Tsai, C.S.: A novel authenticatable color visual secret sharing scheme using non-expanded meaningful shares. *Displays* **32**, 118–134 (2011)
8. Naor, M., Shamir, A.: Visual cryptography. In: *Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'94)*, pp. 1–12. Perugia, Italy (1994)
9. Pang, W.M., Qu, Y., Wong, T.T., Cohen-Or, D., Heng, P.A.: Structure-aware halftoning. *ACM Trans. Graph. (SIGGRAPH 2008 issue)* **27**(3), 89:1–89:8 (2008)
10. Thajeel, S.A.N., Sulong, G.: A novel approach for detection of copy move forgery using completed robust local binary pattern. *J. Inf. Hiding Multimed. Signal Process.* **6**(2), 351–364 (2015)
11. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**(4), 600–612 (2004)
12. Wang, Z.M., Arce, G.R., Crescenzo, G.D.: Halftone visual cryptography via error diffusion. *IEEE Trans. Inf. Forensics Secur.* **4**(3), 383–396 (2009)
13. Yan, B., Wang, Y.F., Song, L.Y., Yang, H.M.: Size-invariant extended visual cryptography with embedded watermark based on error diffusion. *Multimed. Tools Appl.* **75**(18), 11157–11180 (2015)
14. Yang, C.Y., Wang, W.F., Wang, Y.F., Wang, R.Z.: A simple watermarking scheme with high perceptual quality for still color images based on RWM and centroid. *J. Inf. Hiding Multimed. Signal Process.* **6**(3), 577–590 (2015)
15. Zhou, Z., Arce, G.R., Crescenzo, G.D.: Halftone visual cryptography. *IEEE Trans. Image Process.* **15**(8), 2441–2453 (2006)

Chapter 5

Improving Visual Quality for Probabilistic and Random Grid Schemes



5.1 (k, n) -Threshold VC for Grayscale Image

For grayscale secret image, there are two traditional VC encoding schemes: (1) Separate halftoning and VC encoding. (2) Direct block quantization and mapping. As shown in Fig. 5.1.

For separate halftoning and VC encoding [5, 9, 13, 14, 19, 22], a digital halftoning algorithm is applied to input grayscale image and convert it to binary or halftone image. The halftone secret image then can be encoded by the three types of fundamental VC schemes, as reviewed in Chap. 1. This structure is shown in Fig. 5.1a. The advantage of this approach is that one can use the many well-developed and analyzed VC algorithms for binary secret image. For deterministic VC with pixel expansion, this scheme indeed generates good visual quality on target image. But for probabilistic VC and random grid VC, the visual quality of target image is usually far from acceptable.

For direct block quantization and mapping scheme [2, 4], the halftoning step is excluded. It intends to produce size-invariant shares. The grayscale image is segmented into small blocks of square shape. The average intensity of the grayscale image in a block is quantized and mapped to target block with number of black pixels approximately proportional to the average intensity. This idea is shown in Fig. 5.1b. In essence, it is equivalent to sub-sampling, quantization, followed by encryption using size-expanded VC. This sub-sampling inevitably reduces the resolution of the secret image, and the reduced resolution cannot be compensated by other adjacent blocks.

One common feature of the above schemes is the halftone share images and target image. This is necessary since one needs to stack the shares in order to recover the secret image. So the general structure of a grayscale VC can be summarized in Fig. 5.2. This diagram is featured with grayscale secret image, halftone share images and halftone target image.

Following the general requirement on a typical VC scheme, we can define the VC for grayscale image as follows:

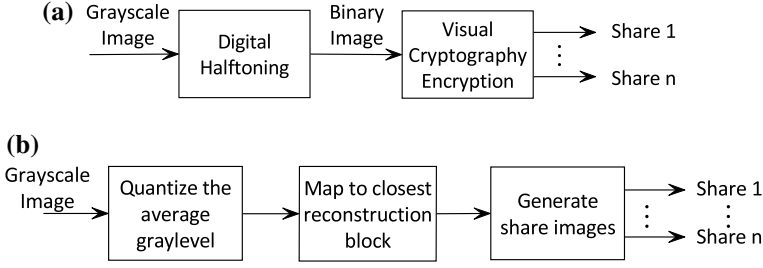
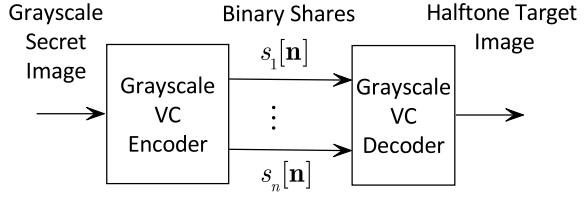


Fig. 5.1 Classification of VC encoding for grayscale secret image. **a** Separate half-toning and VC encoding. **b** Direct block quantization and mapping

Fig. 5.2 Block diagram for a general VC with grayscale secret image



Definition 5.1 ($((k, n)$ -SIGHVC) Considering 8-bit quantized digital image \mathbf{x} , let the set of grayscales of the secret image be \mathbb{Z}_{256} , and those used by the VC encoder be $\{0, 1\}$. Given two grayscale images, each having constant grayscales: $x_1[\mathbf{n}] = g_1$ and $x_2[\mathbf{n}] = g_2$, and $0 \leq g_1 < g_2 \leq 255$. Each image is divided into n shares: $\{s_1^1[\mathbf{n}], \dots, s_n^1[\mathbf{n}]\}$, $\{s_1^2[\mathbf{n}], \dots, s_n^2[\mathbf{n}]\}$, and $s_i^\ell[\mathbf{n}] \in \{0, 1\}$. If the following two conditions are satisfied, then the secret sharing algorithm is called a *grayscale halftone VC*.

1. **Contrast condition:** Let $q \geq k$, and i_1, \dots, i_q be q participants, then the stacking of the q shares

$$z_\ell[\mathbf{n}] \triangleq \bigvee_{i=i_1}^{i_q} s_i^\ell[\mathbf{n}], \ell = 1, 2, \quad (5.1)$$

satisfies $\mathbb{E}(z_1[\mathbf{n}]) < \mathbb{E}(z_2[\mathbf{n}])$.

2. **Security condition:** Let i_1, \dots, i_d be d participants and $d < k$, then, $\forall \mathbf{n}$, the following two random vectors are equivalent:

$$\mathbf{f}_1 = [s_{i_1}^1[\mathbf{n}], \dots, s_{i_d}^1[\mathbf{n}]]^T, \mathbf{f}_2 = [s_{i_1}^2[\mathbf{n}], \dots, s_{i_d}^2[\mathbf{n}]]^T, \quad (5.2)$$

i.e., they have the same sample space and same probability distribution, and are independent of \mathbf{n} . Equivalently, given the random vector

$$\mathbf{f}_\ell \triangleq [s_{i_1}^\ell[\mathbf{n}], \dots, s_{i_d}^\ell[\mathbf{n}]],$$

where $\ell = 1, 2$, no additional information about $x_\ell[\mathbf{n}]$ can be derived, i.e.,

$$\Pr \{x_\ell[\mathbf{n}] = g\} = \Pr \{x_\ell[\mathbf{n}] = g \mid \mathbf{f}_\ell\}, \quad \forall g \in \mathbb{Z}_{256}.$$

As an example, we show that the random grid based VC for grayscale image in [17] is a valid construction of (k, n) - *SIGHVC*. This algorithm is summarized in Algorithm 12, which belongs to the type of separate halftoning and VC encoding, as depicted in Fig. 5.1a. Without loss of generality, we focus on the Algorithm 1 in [17].

Algorithm 12 Random grid VC for grayscale images [17].

Input:

Grayscale image $x[\mathbf{n}]$, with size $M \times N$.

Output:

Binary share images: $s_1[\mathbf{n}], s_2[\mathbf{n}]$

1: Halftone $x[\mathbf{n}]$ to a binary image $y[\mathbf{n}]$.

2: **for** each \mathbf{n} **do**

3: $s_1[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$: draw IID samples from the uniform distribution on set $\{0, 1\}$.

4: $s_2[\mathbf{n}] \leftarrow \begin{cases} 1 - s_1[\mathbf{n}], & \text{if } y[\mathbf{n}] = 1; \\ s_1[\mathbf{n}], & \text{if } y[\mathbf{n}] = 0. \end{cases}$

5: **end for**

Theorem 5.1 *The Algorithm 12 is a valid construction of $(2, 2)$ -SIGHVC.*

Proof We show that the contrast condition and the security condition in Definition 5.1 are satisfied.

(*Contrast*) If we use Algorithm 12 to encrypt two grayscale images with constant levels g_1 and g_2 , where $g_1 < g_2$, then we get shares $\{s_1^1[\mathbf{n}], s_2^1[\mathbf{n}]\}$ and $\{s_1^2[\mathbf{n}], s_2^2[\mathbf{n}]\}$ respectively. The stacking of the two shares are $z_1[\mathbf{n}] = s_1^1[\mathbf{n}] \vee s_2^1[\mathbf{n}]$ and $z_2[\mathbf{n}] = s_1^2[\mathbf{n}] \vee s_2^2[\mathbf{n}]$. Then,

$$\begin{aligned} \mathbb{E}(z_1[\mathbf{n}]) &= \Pr(z_1[\mathbf{n}] = 1) \\ &= 1 - \Pr(s_1^1[\mathbf{n}] = 0, s_2^1[\mathbf{n}] = 0) \\ &= 1 - \frac{1}{2} \left\{ \Pr(y[\mathbf{n}] = 0) \Pr(s_2^1[\mathbf{n}] = 0 \mid s_1^1[\mathbf{n}] = 0, x[\mathbf{n}] = 0) \right. \\ &\quad \left. - \frac{1}{2} \left\{ \Pr(y[\mathbf{n}] = 1) \Pr(s_2^1[\mathbf{n}] = 0 \mid s_1^1[\mathbf{n}] = 0, y[\mathbf{n}] = 1) \right\} \right\} \\ &= 1 - \frac{1}{2} [g_1 \times 1 + (1 - g_1) \times 0] = \frac{1}{2} g_1. \end{aligned}$$

Similarly, $\mathbb{E}(z_2[\mathbf{n}]) = \frac{1}{2} g_2$. Since $g_1 < g_2$, so we have

$$\mathbb{E}(z_1[\mathbf{n}]) < \mathbb{E}(z_2[\mathbf{n}]).$$

The contrast condition is satisfied. Here we utilized the fact that, for a halftone image $y[\mathbf{n}]$ representing a gray level g , we have $\mathbb{E}\{y[\mathbf{n}]\} = g$ [12].

(Security) (1) If we take the first share, then from the generation of random grid, we know that $s_1^1[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$ and $s_1^2[\mathbf{n}] \stackrel{IID}{\sim} \mathcal{U}\{0, 1\}$, so it is obvious that the two random variables $s_1^1[\mathbf{n}]$ and $s_1^2[\mathbf{n}]$ are equivalent. (2) If we take the second share, it is sufficient to prove that $\Pr(s_2^1[\mathbf{n}] = 0) = \Pr(s_2^2[\mathbf{n}] = 0)$.

$$\begin{aligned} \Pr(s_2^1[\mathbf{n}] = 0) &= \Pr(s_1^1[\mathbf{n}] = 0) \Pr(s_2^1[\mathbf{n}] = 0 \mid s_1^1[\mathbf{n}] = 0) + \\ &\quad \Pr(s_1^1[\mathbf{n}] = 1) \Pr(s_2^1[\mathbf{n}] = 0 \mid s_1^1[\mathbf{n}] = 1) \\ &= \frac{1}{2}g_1 + \frac{1}{2}(1 - g_1) = \frac{1}{2}. \end{aligned}$$

Similarly, $\Pr(s_2^2[\mathbf{n}] = 1) = \frac{1}{2}$. So $\Pr(s_2^1[\mathbf{n}] = 1) = \Pr(s_2^2[\mathbf{n}] = 1)$. Thus we conclude that the two random variables $s_2^1[\mathbf{n}]$ and $s_2^2[\mathbf{n}]$ are equivalent.

Since we have shown that Algorithm 12 satisfies both the contrast condition and the security condition in Definition 5.1, Algorithm 12 is a valid construction of (2, 2)-SIGHVC. \square

5.2 Probabilistic VC for Grayscale Secret Image

In addition to the VC schemes depicted in Fig. 5.1, there are several recent efforts in developing VC for grayscale images directly. This section reviews Wang's algorithm and our recent AbS-based probabilistic VC.

5.2.1 Wang's Algorithm

Wang's algorithm operates directly on grayscale image and halftoning is not needed [20]. It also supports flexible pixel expansion for the probabilistic version. For size-invariant scheme, we set the pixel expansion to be 1.

As with many probabilistic schemes, Wang's algorithm is also based on basis matrices of the corresponding deterministic VC for grayscale image. So the deterministic scheme will be presented and then will be extended to probabilistic scheme, as is done in [20].

5.2.1.1 Deterministic VC for Grayscale Secret Image

The deterministic VC for grayscale secret image was developed by Muecke in [15] and Blundo et al. in [1]. Following a similar idea from binary VC, the VC for grayscale secret image with g gray levels can be defined in Definition 5.2.

Definition 5.2 ((k, n, m_g, g) -threshold VC for grayscale secret image [1, 15, 20]: Let the number of gray levels of the secret image be $g \geq 2$. The VC for grayscale secret image consists of g basis matrices $\mathbf{G}_0, \dots, \mathbf{G}_{g-1}$. If a secret pixel with gray level $b \in \{0, \dots, g-1\}$ is to be shared, then one randomly permutes the basis matrix \mathbf{G}_b , and distributes the i -th row to share i . The following two conditions should be satisfied (using notation from Sect. 2.5).

1. **Contrast condition:** For any $q \geq k$, any q participants $\mathbf{P} = \{i_1, \dots, i_q\}$, and each $b \in \{0, \dots, g-2\}$, we have $\mathcal{H}(\eta(\mathbf{G}_b[\mathbf{P}])) \leq d_b - m_g \alpha_{b,b+1}$, and $\mathcal{H}(\eta(\mathbf{G}_{b+1}[\mathbf{P}])) \geq d_b$, where d_b is the threshold parameter, and $\alpha_{b,b+1}$ is the contrast parameter.
2. **Security condition:** For any $q < k$, any q participants $\mathbf{P} = \{i_1, \dots, i_q\}$, $\mathbf{G}_s[\mathbf{P}] \sim \mathbf{G}_t[\mathbf{P}]$, for $s \neq t$.

Recall that $\mathcal{H}(\mathbf{x})$ returns the Hamming weight of a binary vector \mathbf{x} , $\eta(\mathbf{A})$ performs row stacking operation, and $\mathbf{G}[\mathbf{P}]$ is a sub-matrix of \mathbf{G} by restricting to the rows specified in set \mathbf{P} .

The contrast condition ensures that, when enough shares are gathered, then one can distinguish adjacent gray levels s and $s+1$. The security condition ensures that, with insufficient shares, one cannot infer the gray level s from the shares.

The construction of the VC scheme satisfying Definition 5.2 relies on matrix concatenation of basis matrices from binary VC. Let \mathbf{B}_0 and \mathbf{B}_1 be two basis matrices for binary VC, where $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times m}$, then the basis matrix \mathbf{G}_b , $b \in \{0, \dots, g-1\}$ can be constructed as

$$\mathbf{G}_b = \left[\underbrace{\mathbf{B}_0 \cdots \mathbf{B}_0}_{g-1-b} \underbrace{\mathbf{B}_1 \cdots \mathbf{B}_1}_b \right]. \quad (5.3)$$

Obviously, the larger the darkness b , the more usage of \mathbf{B}_1 in \mathbf{G}_b . One problem with this construction is the large pixel expansion. The pixel expansion is given by $m_g = m \times (g-1)$, which increases with the number of gray levels. For typical $g = 256$ and $m = 4$, we get $m_g = 1020$. The share image is more than 1000 times larger than the secret image! Each secret pixel is expanded to roughly a 10×10 block on shares. Furthermore, since more gray levels are to be reproduced, the contrast between adjacent levels are also reduced. It is shown that $\alpha_{b,b+1} = \frac{\alpha}{g-1}$, where α is the contrast for binary VC. Typical value of α is $\alpha = 1/2$, then for $g = 256$, the contrasts are only roughly $1/500$. Note that this small contrast is under the condition of a very large pixel expansion (1000 times).

As an example, consider a $(2, 2)$ case. Let's use the $(2, 2)$ -threshold binary deterministic VC with the following basis matrices:

$$\mathbf{B}_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

We can construct the following basis matrices for $g = 3$ gray levels.

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

5.2.1.2 Probabilistic VC for Grayscale Image

To resolve the pixel expansion in deterministic VC for grayscale image, Wang et al. transformed the deterministic schemes in last section to a probabilistic scheme [20]. The basic idea is to use only m_p of the m_g columns of the basis matrices \mathbf{G}_b , thus producing a probabilistic VC with a smaller pixel expansion $m_p \leq m_g$.

Let the set of basis matrices for a (k, n) -threshold VC with grayscale secret image be \mathbf{G}_b , $b \in \{0, \dots, g-1\}$. Then, for each of the matrix, we get a set of matrices \mathbf{G}_b , where each matrix in it is obtained by randomly sampling m_p columns from \mathbf{G}_b . The encoding process for probabilistic VC is as follows. To share a pixel with gray level b , one randomly chooses one matrix from the set \mathbf{G}_b , and then distributes each row to its corresponding share. This algorithm is summarized in Algorithm 13.

Algorithm 13 (k, n) -threshold probabilistic VC for grayscale image [20].

Input:

Grayscale secret image: $x[\mathbf{n}] \in \{0, \dots, g-1\}$.

Basis matrices: \mathbf{G}_b , $b \in \{0, \dots, g-1\}$.

m_p : pixel expansion.

Output:

n binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

```

1: for  $b \leftarrow 0$  to  $g-1$  do
2:    $\mathbf{G}_b \leftarrow$  all matrices by sampling  $m_p$  columns of  $\mathbf{G}_b$ .
3: end for
4: for every location  $\mathbf{n}$  do
5:   if  $x[\mathbf{n}] = b$  then
6:     Randomly choose one matrix from  $\mathbf{G}_b$ .
7:     Distribute  $i$ -th row to  $\mathbf{n}$ -th block on Share  $s_i$ .
8:   end if
9: end for
10: Output share images  $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$ .
```

Wang's algorithm is shown to preserve the average contrast from deterministic VC, no matter the value of the pixel expansion m_p . However, one may notice that the inherent tradeoff between pixel expansion and block variance is not resolved. The block variance measures the variation of gray levels across different blocks on target image for a given gray level on secret image [14]. For $m_p = m_g$, i.e., the deterministic VC, the block variance is zero. But when m_p deviates from m_g , the block variance

will increase. This is investigated by the measure *recognition area* in Wang's work. In essence, the recognition area is the smallest area required to recognize a given gray level from the surroundings. Higher block variance calls for larger recognition area, hence the target image has lower ability in rendering details in the original secret image.

5.2.2 AbS-Based Probabilistic VC

Unlike¹ the previous effectors in designing the VC for grayscale images, the AbS (Analysis-by-Synthesis) approach intends to approximate the secret image using the target image, by feeding the error back to the VC encoder [24].

The block diagram is shown in Fig. 5.3, which can be described by Algorithm 14.

The *gamut mapping* step is used to ensure that the input gamut is contained within the output gamut. For the diffusion loop to be stable, it is shown that the output gamut must be able to cover the input gamut [6, 21]. To determine the gamut mapping, we must find the input gamut and output gamut. The input gamut is the whole range of a grayscale image. For normalized image, this range is $[0, 1]$, while for 8-bit quantized image, this range is $[0, 255]$. In general, let this range be $[J_{\min}, J_{\max}]$. The output gamut should be the average gamut since the output has average contrast of $1/2^{n-1}$ for the probabilistic algorithm build upon the basis matrices in (2.10). For this (n, n) -threshold case, we can choose the output gamut $[\beta_{\min}, \beta_{\max}] = [0, 1/2^{n-1}]$. The corresponding gamut mapping is an affine function:

$$x[n] = \frac{\beta_{\max} - \beta_{\min}}{J_{\max} - J_{\min}} J[n] + \frac{\beta_{\min} J_{\max} - \beta_{\max} J_{\min}}{J_{\max} - J_{\min}}. \quad (5.4)$$

Gamut mapping functions other than the affine function can also be adopted here. For example, the mapping $\mathcal{G} : [J_{\min}, J_{\max}] \rightarrow [\beta_{\min}, \beta_{\max}]$ can be a nonlinear or an

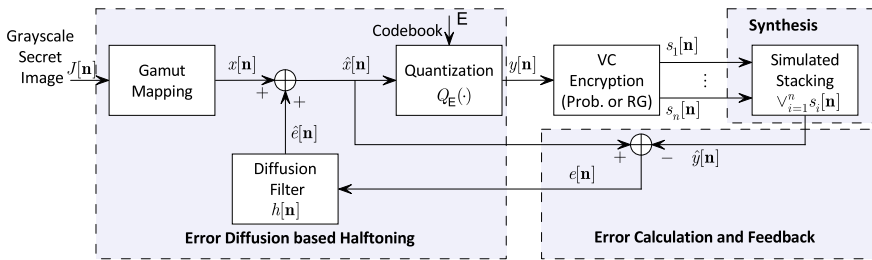


Fig. 5.3 Block diagram for AbS-based probabilistic VC for grayscale image

¹© 2019 IEEE. Reprinted, with permission, from IEEE Transactions on Image Processing, Vol. 28, No. 2, Improving the Visual Quality of Size-Invariant Visual Cryptography for Grayscale Images: An Analysis-by-Synthesis (AbS) Approach, Bin Yan et al.

Algorithm 14 (n, n) -AbSProb: AbS-based Probabilistic VC for Grayscale Images.**Input:**

Grayscale image $J[\mathbf{n}] \in \mathbb{Z}_{256}$, with size $M \times N$.

Two basis matrices for deterministic (n, n) -threshold binary VC: \mathbf{B}_0 and \mathbf{B}_1 .

Output:

Binary share images: $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$

1: Gamut mapping of $J[\mathbf{n}]$ to ensure stability.

2: **for** each pixel \mathbf{n} **do**

3: Quantize $\hat{x}[\mathbf{n}]$ with codebook $\mathbf{E} = \{0, 1\}$: $y[\mathbf{n}] \leftarrow \mathcal{Q}_{\mathbf{E}}(\hat{x}[\mathbf{n}])$.

4: Encode $y[\mathbf{n}]$ into $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$ using the (n, n) -threshold probabilistic VC: If $y[\mathbf{n}] = 1$ (0 resp.), randomly select one column from \mathbf{B}_1 (\mathbf{B}_0 resp.), permute the rows and assign them to $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$, respectively.

5: Simulated stacking: $\hat{y}[\mathbf{n}] \leftarrow \bigvee_{i=1}^n s_i[\mathbf{n}]$.

6: Calculate the reconstruction error: $e[\mathbf{n}] \leftarrow \hat{x}[\mathbf{n}] - \hat{y}[\mathbf{n}]$.

7: Error diffusion: Push the error to all pixels in the neighborhood of \mathbf{n} , i.e., $\forall \mathbf{m} \in \mathbf{N}(\mathbf{n}), \hat{x}[\mathbf{m}] \leftarrow \hat{x}[\mathbf{m}] + h[\mathbf{m}]e[\mathbf{n}]$, where $h[\mathbf{m}]$ is the coefficient of the diffusion filter.

8: **end for**

9: Output the shares: s_1, \dots, s_n .

adaptive function, such as the affine mapping followed by histogram equalization within the range $[\beta_{\min}, \beta_{\max}]$.

The diffusion loop is similar to the loop in and error diffusion-based halftoning. For example, one can use the Floyd and Steinberg's kernel [7], the Jarvis' kernel [10], or the Stucki's kernel [18].

To show that the Algorithm 14 is a valid VC for grayscale image as defined in Definition 5.1, we must show that the security condition and contrast condition are satisfied. For a vigorous proof, we direct the reader to [24]. Here we give an intuitive explanation to security condition and another proof for the contrast condition.

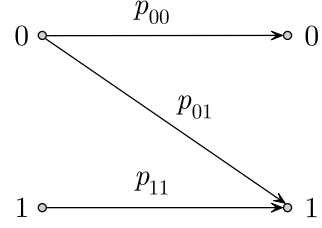
For the security condition, we should note that the binary VC algorithm is chosen to be a secure one. So from any $q < k$ shares, one cannot infer the input to VC encoder $y[\mathbf{n}]$. Referring to Fig. 5.3, no matter how $y[\mathbf{n}]$ is obtained from $J[\mathbf{n}]$, as long as we cannot infer $y[\mathbf{n}]$ from q shares, then it is impossible to infer $J[\mathbf{n}]$ from the shares. So, given $q < k$ shares, it is impossible to infer the gray level $J[\mathbf{n}]$.

For the contrast condition, we utilize the linear model for error diffusion depicted in Fig. 3.12. Furthermore, the VC encoding and stacking should also be modeled. For example, for $(2, 2)$ case, given a black secret pixel, one will get a black target pixel. Given a white secret pixel, the chances of getting a white target pixel is $1/2$. Thus, we can use a binary channel model as shown in Fig. 5.4. For $(2, 2)$ case, we should have $p_{00} = 1/2$, $p_{01} = 1/2$, and $p_{11} = 1$.

Note that the contrast condition is concerning the statistical average, so we only need to focus on the average input/output. Focusing on statistical average, the combination of VC encoding and stacking is equivalent to a linear function

$$\mathbb{E}\{\hat{y}[\mathbf{n}]\} = \kappa \mathbb{E}\{y[\mathbf{n}]\}. \quad (5.5)$$

Fig. 5.4 Binary non-symmetric model for VC encoding and stacking



We would like to note that, since the diffusion filter is shift invariant and preserves DC, i.e., $\sum_{\mathbf{m}} h[\mathbf{m}] = 1$, then

$$\begin{aligned} \mathbb{E}(\hat{e}[\mathbf{n}]) &= \mathbb{E}\left(\sum_{\mathbf{m}} e[\mathbf{m}] h[\mathbf{n} - \mathbf{m}]\right) \\ &= \mathbb{E}(e[\mathbf{n}]) \sum_{\mathbf{m}} h[\mathbf{n} - \mathbf{m}] = \mathbb{E}(e[\mathbf{n}]). \end{aligned} \quad (5.6)$$

The above second equality holds because of the stationarity of $e[\mathbf{n}]$.

Next, we calculate $\mathbb{E}(e[\mathbf{n}])$ as follows.

$$\begin{aligned} \mathbb{E}(e[\mathbf{n}]) &= \mathbb{E}(\hat{x}[\mathbf{n}] - \hat{y}[\mathbf{n}]) \\ &= \mathbb{E}(\hat{x}[\mathbf{n}]) - \kappa \mathbb{E}(\gamma \hat{x}[\mathbf{n}] + q[\mathbf{n}]) \\ &= \mathbb{E}(\hat{x}[\mathbf{n}]) (1 - \kappa \gamma) - \kappa \mathbb{E}(q[\mathbf{n}]) \\ &= (g + \mathbb{E}(\hat{e}[\mathbf{n}])) (1 - \kappa \gamma) - \kappa \mathbb{E}(q[\mathbf{n}]) \end{aligned} \quad (5.7)$$

$$= g(1 - \kappa \gamma) + \mathbb{E}(e[\mathbf{n}]) (1 - \kappa \gamma) - \kappa \mathbb{E}(q[\mathbf{n}]) \quad (5.8)$$

From (5.7) to (5.8), we use the result from (5.6). Solving for $\mathbb{E}(e[\mathbf{n}])$, we obtain

$$\mathbb{E}(e[\mathbf{n}]) = \frac{1 - \kappa \gamma}{\kappa \gamma} g - \frac{1}{\gamma} \mathbb{E}(q[\mathbf{n}]). \quad (5.9)$$

Based on (5.9), it results in

$$\begin{aligned} \mathbb{E}(\hat{y}[\mathbf{n}]) &= \kappa \mathbb{E}(y[\mathbf{n}]) = \kappa \mathbb{E}(\gamma (x[\mathbf{n}] + \hat{e}[\mathbf{n}]) + q[\mathbf{n}]) \\ &= \kappa \gamma (g + \mathbb{E}(e[\mathbf{n}])) + \kappa \mathbb{E}(q[\mathbf{n}]) \\ &= \kappa \gamma g + (1 - \kappa \gamma) g - \kappa \mathbb{E}(q[\mathbf{n}]) + \kappa \mathbb{E}(q[\mathbf{n}]) \\ &= g. \end{aligned}$$

Note that $\mathbb{E}(q[\mathbf{n}])$ can be nonzero. Since the noise transfer function (NTF) between $\hat{y}[\mathbf{n}]$ and $q[\mathbf{n}]$ is high-pass (see Eq. (8) in [11]), this nonzero mean value is suppressed by the loop. Hence it doesn't affect the mean value of the output $\hat{y}[\mathbf{n}]$.

So, we have shown that the average value of the stacking result $\hat{y}[\mathbf{n}]$ is equal to the input gray level. If we have two constant images $x_1[\mathbf{n}] = g_1$ and $x_2[\mathbf{n}] = g_2$ with $g_1 < g_2$, then it follows that $\mathbb{E}(\hat{y}_1[\mathbf{n}]) < \mathbb{E}(\hat{y}_2[\mathbf{n}])$. So, the contrast condition is satisfied. For a proof without assuming stationarity for $e[\mathbf{n}]$, please see [24].

The performance of the Yang's algorithm, Wang's algorithm and AbS-based probabilistic VC on constant grayscale image are shown in Fig. 5.5, Fig. 5.6, and Fig. 5.7, respectively. It is evident that AbS-based probabilistic VC provides more uniform and visually pleasing target image.

The RAPSD results are shown in Fig. 5.8, Fig. 5.9, and Fig. 5.10, respectively. Comparing these three figures, we see that AbS-based probabilistic VC produces less low-frequency noise. Low frequency noise is the major reason of quality degradation on target image.

To further characterize the influence of low-frequency noise, we use the *residual variance* measure introduced in Sect. 3.6 [24]. For the (2, 2)-threshold probabilistic VC, the result is shown in Fig. 5.11.

For the fidelity test, we use the Baboon image as shown in Fig. 5.12. The HPSNR is found to be 19.45 dB and the MSSIM is 0.658. The result in Fig. 5.12e shows

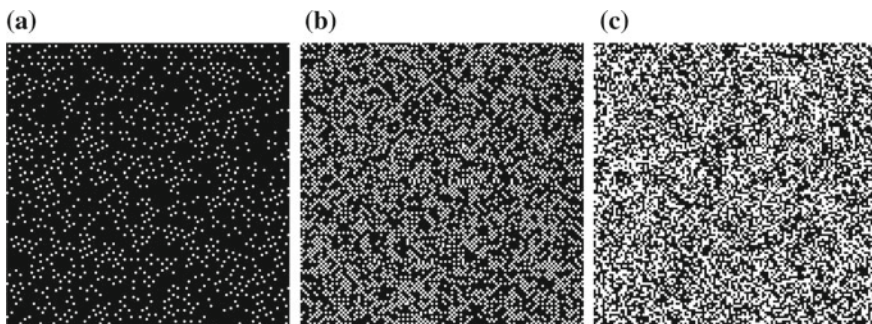


Fig. 5.5 Experimental result for Yang's (2, 2)-threshold probabilistic VC, using constant grayscale images $J[\mathbf{n}] = g$. **a** $g = 7/8$. **b** $g = 1/2$. **c** $g = 1/8$

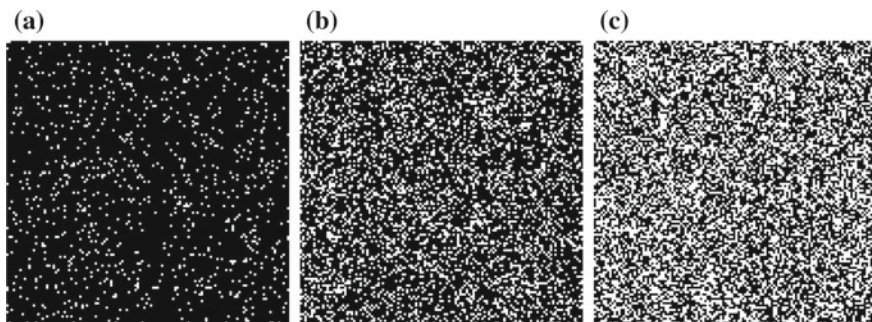


Fig. 5.6 Experimental result for Wang's (2, 2)-threshold probabilistic VC, using constant grayscale images $J[\mathbf{n}] = g$. The pixel expansion is 1 **a** $g = 7/8$. **b** $g = 1/2$. **c** $g = 1/8$

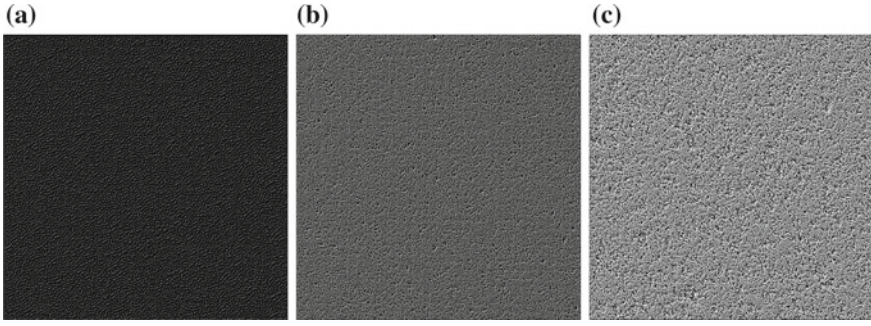
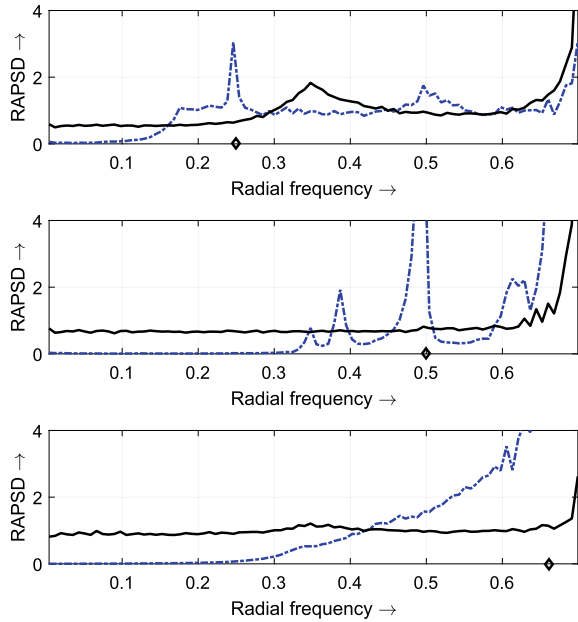


Fig. 5.7 Experimental result for (2, 2)-threshold AbS-based probabilistic VC, using constant grayscale images $J[n] = g$. **a** $g=7/8$. **b** $g=1/2$. **c** $g=1/8$

Fig. 5.8 RAPSD result for Yang's (2, 2)-threshold probabilistic VC, using constant grayscale images $J[n] = g$. From top $g=7/8$, $g=1/2$ and $g=1/8$



that, AbS-based probabilistic reproduces more details on target image than Yang's probabilistic VC. One may also notice that, the darker regions are better reproduced on target image. The white regions exhibit clusters of black and white pixels. This can be remedied by AbS-based vector VC in next chapter.

Fig. 5.9 RAPSD result for Wang's (2, 2)-threshold probabilistic VC, using constant grayscale images $J[\mathbf{n}] = g$. From top $g = 7/8$, $g = 1/2$ and $g = 1/8$

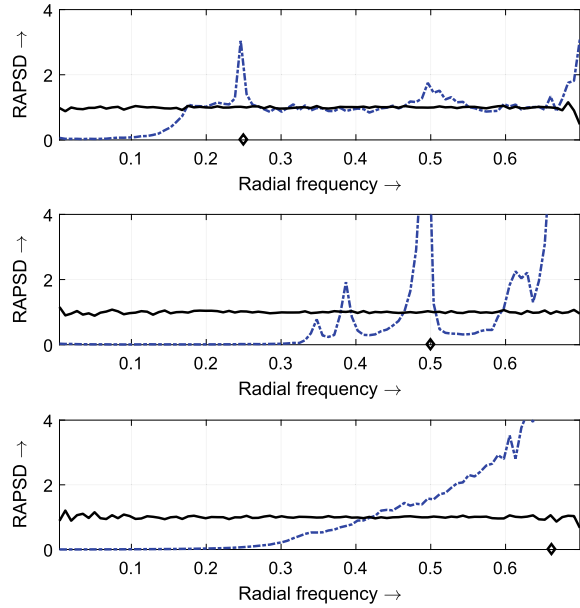


Fig. 5.10 RAPSD result for (2, 2)-threshold AbS-based probabilistic VC, using constant grayscale images $J[\mathbf{n}] = g$. From top $g = 7/8$, $g = 1/2$ and $g = 1/8$

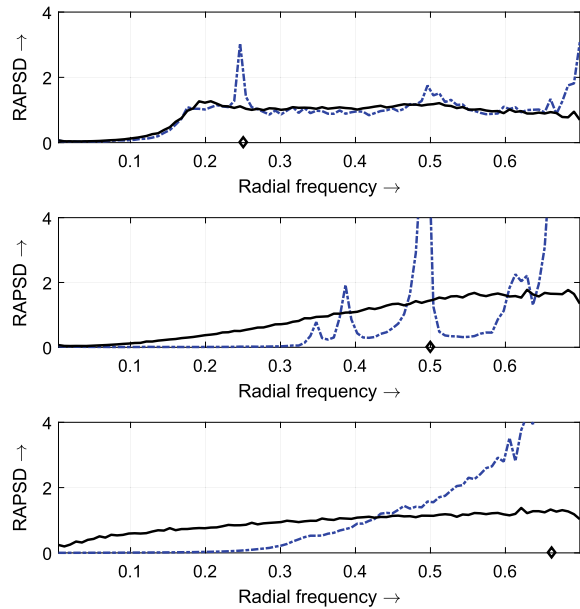
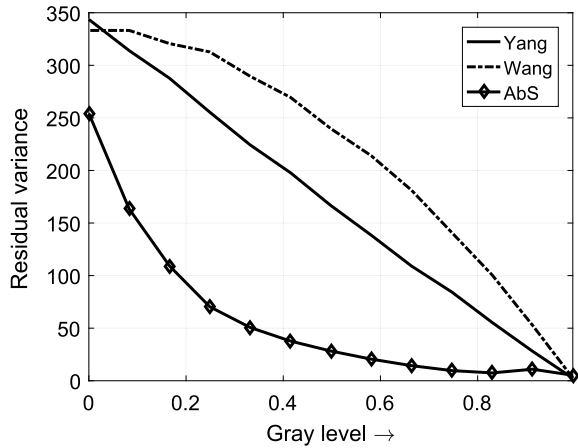


Fig. 5.11 Residual variance result for $(2, 2)$ -threshold probabilistic VC schemes, using constant grayscale images $J[\mathbf{n}] = g, g \in [0, 1]$



5.3 Random Grid VC for Grayscale Secret Image

The random grid VC also suffers from bad visual quality for grayscale images. For the random grid framework, we review three algorithms. The first one is the (n, n) -threshold random grid VC constructed by Chen and Tsao [3], which is the application of binary scheme to halftoned grayscale image. The second one is a recent algorithm constructed by Wu et al. [23], where efforts have been made to generate blue noise shares. The third algorithm is the AbS-based approach, which intends to produce blue noise target images directly.

5.3.1 Applying Binary Scheme to Grayscale Image

Application of random grid approach to grayscale images started from Shyu's work [17], where the $(2, 2)$ -threshold random grid is extended to grayscale and color secret images. The secret image is first converted to halftone image and then binary random grid VC is applied to these halftone images. The special features of the halftone image are not considered. An experimental result on Lena image is shown in Fig. 5.13. Many details and low contrast edges are not visible in the target image Fig. 5.13c.

5.3.2 Blue Noise Approach

Wu et al. propose to improve the random grid approach by producing blue noise shares [22, 23]. In [22], the authors use a blue noise pattern generation algorithm—

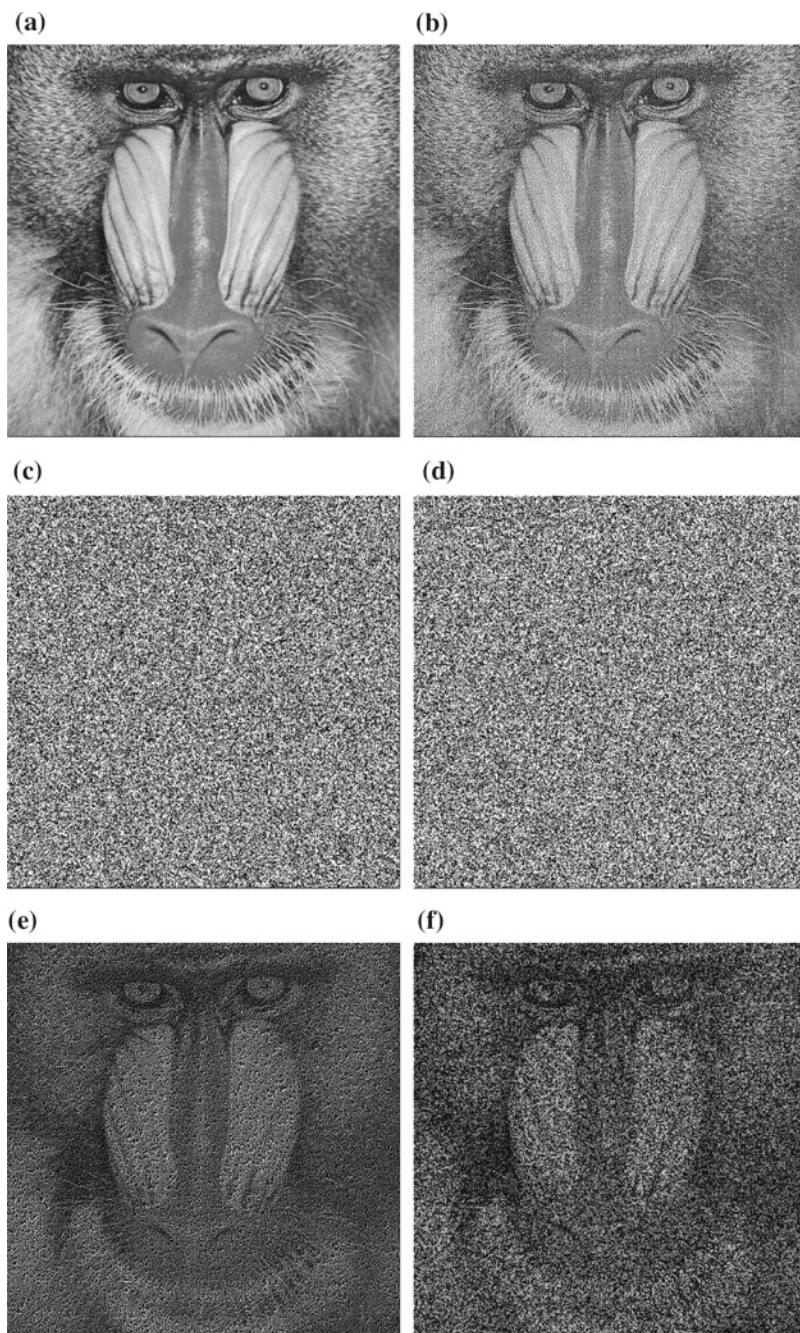


Fig. 5.12 Experimental result for $(2,2)$ -threshold AbS-based probabilistic VC, using Baboon image. **a** Original secret image. **b** Halftone image. **c** Share 1. **d** Share 2. **e** Target image from AbS-based probabilistic VC. **f** Target image using Yang's algorithm

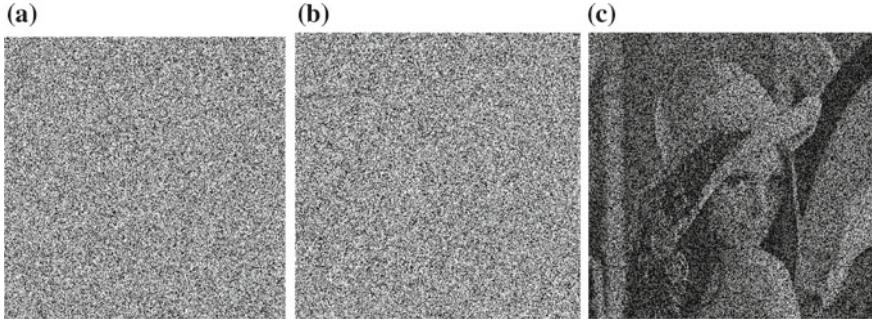


Fig. 5.13 Experimental result for (2, 2)-threshold random grid, using halftone secret image. **a** Share 1. **b** Share 2. **c** Target image

the void and cluster algorithm, as a post-processing step to generate random grid. In [23], a random noise balanced error diffusion (RNBED) scheme is adopted. This scheme is shown to improve the visual quality for binary image (binary logo, text, or thresholded grayscale image) quite well. We briefly review the RNBED-based algorithm.

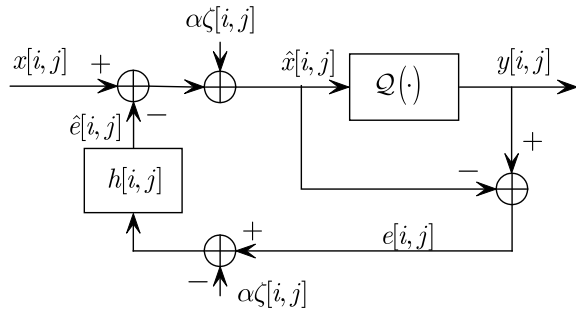
The RNBED was originally proposed as a solution to data hiding in halftoning process [16]. According to the bits to hide, one adjusts the threshold by injecting noise into the image to be halftoned. Wu et al. modified the RNBED to generate share image so that the black pixels are separated apart.

Recall from Sect. 3.4 the structure of error diffusion halftoning. The RNBED injects random noise into this diffusion loop, as shown in Fig. 5.14. The parameter α controls the strength of the IID random noise $\zeta[i, j] \in \{0, 1\}$. To ensure that the random grid $y[i, j]$ has equal number of white pixels and black pixels, the input gray level $x[i, j]$ is set such that the average is $1/2$. For example, one can use

$$x[i, j] \stackrel{IID}{\sim} \mathcal{U}[0.5 - u, 0.5 + u] \quad (5.10)$$

for a preselected parameter u .

Fig. 5.14 Block diagram for random noise balanced error diffusion (RNBED)



The other ingredient is the classification of secret blocks according to the number of black pixels in it. The collection of all blocks having t pixels and b black pixels is denoted as a set $B_{t,b}$. It is expected that each type of block should be treated differently since different number of black pixels are clustered together. So a strength α_b is designed for block type $B_{t,b}$, where $b = 0, \dots, t - 1$. Combining the RNBED random grid generator and the $(2, n)$ -threshold random grid VC (Algorithm 4), we get the algorithm described in Algorithm 15. The algorithm $\text{Rnbcd}(x[n], \alpha)$ executes one step of the RNBED algorithm using parameter α and output one sample $y[n]$, as shown in Fig. 5.14.

Algorithm 15 $(2, n)$ -threshold random grid VC using RNBED [23].

Input:

Binary secret image: $x[n] \in \{0, 1\}$.

Parameters: $u, \alpha_0, \dots, \alpha_4$.

Output:

n binary share images: $s_1[n], \dots, s_n[n]$.

- 1: Block partition: Divide the image $x[n]$ into non-overlapping 2×2 blocks, and classify each block as one of the five types $B_{4,b}$, $b = 0, \dots, 4$.
 - 2: **for** $k \leftarrow 1$ to n **do**
 - 3: $g_k[n] \stackrel{IID}{\sim} \mathcal{U}[0.5 - u, 0.5 + u], \forall n$.
 - 4: **end for**
 - 5: **for** each block B **do**
 - 6: Set $\alpha = \alpha_b$, if $B \in B_{4,b}$, $b = 0, \dots, 4$.
 - 7: **for** each pixel $x[n] \in B$ **do**
 - 8: **if** $x[n] = 1$ **then**
 - 9: $s_k[n] \leftarrow \text{Rnbcd}(g_k[n], \alpha), k = 1, \dots, n$.
 - 10: **else**
 - 11: $d \sim \mathcal{U}\{1, 2, \dots, n\}$.
 - 12: $s_d[n] \leftarrow \text{Rnbcd}(g_d[n], \alpha)$.
 - 13: $s_k[n] \leftarrow s_d[n], \forall k \in \{1, \dots, n\}$, and $k \neq d$.
 - 14: **end if**
 - 15: **end for**
 - 16: **end for**
 - 17: Output share images $s_1[n], \dots, s_n[n]$.
-

Using parameter values $u = 0.1$, $\alpha_0 = \alpha_4 = 0.26$ and $\alpha_1 = \alpha_2 = \alpha_3 = 0.25$ as suggested in [23], and using thresholded Lena image as secret image, we get the experimental results as shown in Fig. 5.15, for $(2, 2)$ -threshold case. We observe that, for the region on the target image that corresponds to the white regions in the secret image, the black pixels are evenly distributed. This can be understood from Algorithm 15. For a white secret pixel, one randomly chooses one random grid and copies it to all other shares. If this random grid has evenly distributed black dots, then the stacking results also have evenly distributed black dots.

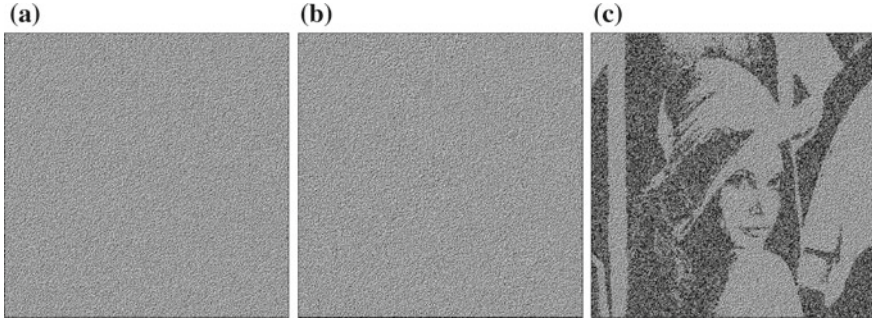


Fig. 5.15 Experimental result for (2, 2)-threshold RNBED random grid, using binary secret image. **a** Share 1. **b** Share 2. **c** Target image

5.3.3 *AbS-Based Algorithm for Random Grid Visual Cryptography*

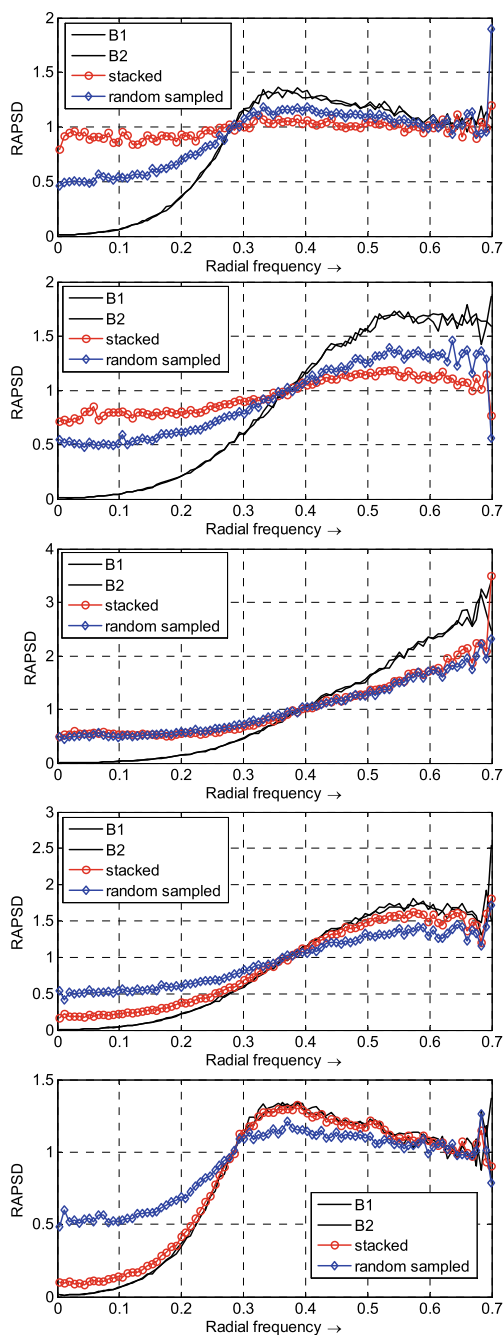
In contrast to producing blue noise shares, the AbS framework can be considered as utilizing the blue noise technique on the target image. Also, we note that two blue noise patterns, when stacked or randomly sampled to form a third image, then the resulting image is no longer of blue noise. This is illustrated in Fig. 5.16 of the experimental result.

In this experiment, n blue noise patterns are generated using Random noise balanced error diffusion (RNBED) [8, 16, 23]. Based on these n blue noise images, another two images are generated. The first image is the stacking of these n blue noise images. The second one is generated by random sampling from these blue noise images at each pixel location. We then use RAPSD to characterize the spectral property. The result is shown in Fig. 5.16.

When $g \leq \frac{1}{2}$, even though the images B_1 and B_2 are blue noise, but the stacked image shows significant spectral components in low frequency range. As g approaches 1, the low frequency components decrease on the stacked image. This can be explained as follows. When g is small (near zero), black pixels dominate, and stacking will produce more black pixels on the stacked image. Note that the black pixels on the two shares are independent of each other, so stacking is a random mixing of black pixels from the two shares. This random mixing may produce clusters of black pixels, thus increases the wavelength between the minority pixels (the white pixels). This increase in wavelength is responsible for the significant low frequency component on RAPSD of the stacked image. As g increases to be above $\frac{1}{2}$, white pixels dominate and the black pixel is the minority pixel. Even though more black pixels are produced after stacking, the wavelengths between the black pixels don't increased significantly.

The above analysis also verifies that the algorithms in [22, 23] produce good results on binary image or thresholded grayscale image that are featured with large and connected black regions and white regions. According to the encryption process,

Fig. 5.16 The RAPSD of blue noise patterns B_1 and B_2 , the stacked image and random sampled image. $n = 2$. From top to bottom, the gray levels are $g = \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}$



the reconstructed pixels corresponding to the white regions in the secret image are equal to the corresponding pixels in the first share. So, if the first share is of blue noise, then the reconstructed white area is blue noise. But this good performance cannot be transferred to halftone image, where the gray level is represented as relative frequency of occurrence of white and black pixels. Note that the halftone secret image can be thought of as a random point process, so even though the shares are of blue noise, the stacking operation is equivalent to using the white pixels to select the points in share 1 and ‘copy’ them to the reconstructed image. This random selection also destroys the blue noise property. In addition, The low frequency noise due to random selection doesn’t change with g .

These experimental results suggest that, for halftone secret image, instead of producing blue noise shares, one should try to produce blue noise stacking result directly.

Since the random grid approach is also pixel based, the block diagram for AbS-based random grid VC is the same as that of the AbS-based probabilistic VC. Please refer to Fig. 5.3. Here we specialize the block ‘VC Encoding’ and ‘Gamut Mapping’ for random grid VC.

A brief review of (n, n) -threshold can be found from Sect. 2.4 (Algorithm 3). In general, any (n, n) -threshold random grid VC can be used in AbS framework.

Next, we determine the gamut mapping for (n, n) -threshold random grid VC. Since random grid VC is pixel based, it is not difficult to determine $\beta_{\min} = \mathbb{E}\{\hat{y}[\mathbf{n}] | y[\mathbf{n}] = 0\} = 0$. To calculate β_{\max} , we note that $s_1[\mathbf{n}], \dots, s_{n-1}[\mathbf{n}]$ can be equivalently treated as independently generated. So,

$$\beta_{\max} = \gamma \cdot \mathbb{E}\{\hat{y}[\mathbf{n}] | y[\mathbf{n}] = 1\} = \gamma \Pr\{\hat{y}[\mathbf{n}] = 1 | y[\mathbf{n}] = 1\} \quad (5.11)$$

$$= \gamma \Pr\{\bigvee_{i=1}^{n-1} s_i[\mathbf{n}] = 1 | y[\mathbf{n}] = 1\} = \gamma \prod_{i=1}^{n-1} \Pr\{s_i[\mathbf{n}] = 1\} \quad (5.12)$$

$$= \frac{\gamma}{2^{n-1}}. \quad (5.13)$$

γ is a user specified parameter [24]. So in the (n, n) -threshold AbS-based random grid VC, the dynamic range of the input must be mapped into the range $[\beta_{\min}, \beta_{\max}] = [0, \frac{\gamma}{2^{n-1}}]$, where $\gamma \leq 1$.

The experimental result for AbS-based random grid VC is quite similar to the experimental result for probabilistic VC (Figs. 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12). So we omit them. This similarity can be explained by the equivalence between probabilistic VC and random grid VC established by Yang et al. [25].

5.4 Remarks

Probabilistic VC can effectively solve the size expansion problem, but it comes with the price of bad visual quality, especially for grayscale secret image. Producing blue noise pattern for share images is shown to be able to improve the quality of target

image, but is not suitable for halftone image with complicated contents. We define the grayscale VC directly and construct AbS-based probabilistic VC and AbS-based random grid VC, in order to produce blue noise on target image. This framework is shown to meet the requirement of a grayscale VC. One drawback of AbS-based probabilistic VC is that the performance deteriorates for bright region. We also remark that, the AbS approach intends to let neighboring pixels to cooperate, in order to improve the quality of the target image. This idea is similar to block encoding, or vector VC, which will be introduced in next chapter.

References

1. Blundo, C., Santis, A.D., Naor, M.: Visual cryptography for grey level images. *Inf. Process. Lett.* **75**(6), 255–259 (2000)
2. Chen, H.H., Lee, C.C., Lee, C.C., Wu, H.C., Tsai, C.S.: Multi-level visual secret sharing scheme with smooth-looking. In: *Proceedings of the 2nd International Conference on Interaction Sciences (ICIS'09)*, ACM, New York, NY, USA, pp. 155–160 (2009)
3. Chen, T.H., Tsao, K.H.: Visual secret sharing by random grids revisited. *Pattern Recognit.* **42**(9), 2203–2217 (2009)
4. Chen, Y.F., Chan, Y.K., Huang, C.C., Tsai, M.H., Chu, Y.P.: A multiple-level visual secret-sharing scheme without image size expansion. *Inf. Sci.* **177**(21), 4696–4710 (2007)
5. Chow, Y.W., Susilo, W., Wong, D.S.: Enhancing the perceived visual quality of a size invariant visual cryptography scheme. In: *Proceedings of the 14th International Conference on Information and Communications Security (ICICS'12)*, Hong Kong, China, pp. 10–21 (2012)
6. Escbbach, R., Fan, Z., Knox, K.T., Marcu, G.: Threshold modulation and stability in error diffusion. *IEEE Signal Process Mag.* **20**(4), 39–50 (2003)
7. Floyd, R.W., Steinberg, L.: An adaptive algorithm for spatial gray-scale. *Proc. Soc. Inf. Disp.* **17**(2), 75–78 (1976)
8. Guo, J.M., Tsai, J.J.: Data-hiding in halftone images using adaptive noise-balanced error diffusion. *IEEE Multimed.* **18**(2), 48–59 (2011)
9. Hou, Y.C., Tu, S.F.: A visual cryptographic technique for chromatic images using multi-pixel encoding method. *J. Res. Pract. Inf. Technol.* **37**(2), 179–191 (2005)
10. Jarvis, J.F., Judice, C.N., Ninke, W.H.: A survey of techniques for the display of continuous tone pictures on bilevel displays. *Comput. Graph. Image Process.* **5**(1), 13–40 (1976)
11. Kite, T.D., Evans, B.L., Bovik, A.C.: Modeling and quality assessment of halftoning by error diffusion. *IEEE Trans. Image Process.* **9**(5), 909–922 (2000)
12. Lau, D., Arce, G.: *Modern Digital Halftoning*, 2nd edn. Taylor & Francis, Boca Raton, FL (2001)
13. Lee, C.C., Chen, H.H., Liu, H.T., Chen, G.W., Tsai, C.S.: A new visual cryptography with multi-level encoding. *J. Visual Lang. Comput.* **25**(3), 243–250 (2014)
14. Liu, F., Guo, T., Wu, C., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. *J. Visual Commun. Image Represent.* **23**(2), 331–342 (2012)
15. Muecke, I.: *Greyscale and colour visual cryptography*. Master's thesis, Dalhousie University (1999)
16. Pei, S.C., Guo, J.M.: Data hiding in halftone images with noise-balanced error diffusion. *IEEE Signal Process. Lett.* **10**(12), 349–351 (2003)
17. Shyu, S.J.: Image encryption by random grids. *Pattern Recognit.* **40**(3), 1014–1031 (2007)
18. Stucki, P.: MECCA—a multiple-error correcting computation algorithm for bilevel image hard-copy reproduction. Technical Report RZ1060, IBM Research Laboratory, Zurich, Switzerland (1981)

19. Tu, S.F., Hou, Y.C.: Design of visual cryptographic methods with smooth looking decoded images of invariant size for grey-level images. *Imaging Sci. J.* **55**(2), 90–101 (2007)
20. Wang, D.S., Yi, F., Li, X.B.: Probabilistic visual secret sharing schemes for grey-scale images and color images. *Inf. Sci.* **181**(11), 2189–2208 (2011)
21. Wu, C.W., Thompson, G.R., Stanich, M.J.: Digital watermarking and steganography via overlays of halftone images. *Proc. SPIE* **5561**, 152–163 (2004)
22. Wu, X.T., Sun, W.: Improving the visual quality of random grid-based visual secret sharing. *Signal Process.* **93**(5), 977–995 (2013)
23. Wu, X.T., Liu, T., Sun, W.: Improving the visual quality of random grid-based visual secret sharing via error diffusion. *J. Visual Commun. Image Represent.* **24**(5), 552–566 (2013)
24. Yan, B., Xiang, Y., Hua, G.: Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (AbS) approach. *IEEE Trans. Image Process.* **28**(2), 896–911 (2019)
25. Yang, C.N., Wu, C.C., Wang, D.S.: A discussion on the relationship between probabilistic visual cryptography and random grid. *Inf. Sci.* **278**, 141–173 (2014)

Chapter 6

Improving Visual Quality for Vector Schemes



6.1 Vector Visual Cryptography

For the three classical VC schemes reviewed in Chap. 2, each secret pixel is independently encoded. Vector visual cryptography, however, treats a block of pixels as a unit and encodes them in one single step. By encoding a block, one can design VC algorithms to transfer the local feature of that block to local feature of corresponding block on target image, hence may get better perceptual quality for target image.

Focusing on size-invariant schemes, the assignment to shares from basis matrices is probabilistic. So, the global contrast between black and white region is only preserved in statistical sense, i.e., in the sense of the whole ensemble of black pixels and white pixels. But locally, there is a large variation, or variance, in local contrast distribution.

Considering the two existing problems outlined above: sample-wise encoding and probabilistic approach, one possible solution is to encode a local block directly, so that the local contrast can be better preserved [1, 2, 7, 11].

Hou et al. extended the basic Ito algorithm to vector VC [7]. Blocks having r successive white/black pixels are taken from the image and encoded. This can ensure that for r successive white/black blocks, the local contrast is guaranteed. But these r white/black blocks may not be adjacent to each other. So, even though this scheme is block-based, but the encoding of each type of block is still probabilistic and only ensures that globally the contrasts between different types of blocks are preserved.

In [11], image blocks are classified according to the number of black pixels in them. Then a counter is assigned to each block type. For example, for the type- b block having b black pixels, Tu et al. use this counter to ensure that the basis matrix \mathbf{B}_1 for black pixel is used exactly b times in encoding this type of block. Here the contrast is guaranteed for each type of block, but is not guaranteed for a small local region. This algorithm improves Hou's algorithm, but the contrast is still a contrast between different type of blocks that may not be spatially adjacent. Furthermore, both Hou's algorithm and Tu's algorithm use halftone image as secret image.

Chen et al. designed vector VC for grayscale image directly. The average gray level of an image block is used to select the corresponding block on the stacked image: an image block with darker average gray level is mapped to a block on the stacked image having more black pixels in it [1]. However, each block is processed independently so that the loss of contrast in one block cannot be compensated by other blocks.

Loss of contrast due to VC is a key issue especially when the original secret image already has a low contrast. In [8], an implicit histogram equalization is utilized in block encoding to extend the histogram of the target image (after inverse halftoning). But each block is still independently encoded.

To remedy the existing problems outlined above, we propose to use AbS framework and vector VC. For halftone secret image, using local blackness preserving VC [12], the loss of contrast in one block can be compensated by other adjacent blocks. For grayscale secret image, an AbS-based vector VC can be used to let the target image to approximate the secret image directly [13].

6.1.1 Hou's Block Encoding Algorithm

Hou's block encoding algorithm was proposed in [6, 7, 11], which was later improved by Liu et al. in [9]. These algorithms receive halftone image as secret image and are based on block classification. Different encoding matrices are used for different block types.

In Tu and Hou's work [11], the secret image is partitioned into small blocks of size $m = \ell^2$. Then these blocks are classified according to a local feature—the number of black pixels b in it. Let \mathbf{x} be such a small block, then the three types (or sets) are:

$$\mathbf{M}_0 = \left\{ \mathbf{x} : \text{such that } \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} x[i, j] < \frac{m}{2} \right\}, \quad (6.1)$$

$$\mathbf{M}_1 = \left\{ \mathbf{x} : \text{such that } \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} x[i, j] = \frac{m}{2} \right\}, \quad (6.2)$$

$$\mathbf{M}_2 = \left\{ \mathbf{x} : \text{such that } \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} x[i, j] > \frac{m}{2} \right\}. \quad (6.3)$$

Let \mathbf{B}_0 and \mathbf{B}_1 be two basis matrices used in binary deterministic VC. For a block $\mathbf{x} \in \mathbf{M}_0$, one uses matrix \mathbf{B}_0 . For a block $\mathbf{x} \in \mathbf{M}_2$, one uses matrix \mathbf{B}_1 . For a block $\mathbf{x} \in \mathbf{M}_1$, one uses \mathbf{B}_0 and \mathbf{B}_1 alternately. This algorithm is summarized in Algorithm 16.

Tu and Hou's algorithm uses different frequencies of \mathbf{B}_0 and \mathbf{B}_1 to generate contrast between different types of blocks. If all blocks of the same type, say \mathbf{M}_1 , are clustered together, then the generated contrast is significant enough to distinguish

Algorithm 16 Block encoding algorithm proposed by Tu and Hou [11].**Input:**Halftone secret image: x [\mathbf{n}].Basis matrices $\mathbf{B}_0, \mathbf{B}_1$ for binary (k, n) threshold VC.**Output:**Share images : s_1 [\mathbf{n}], \dots , s_n [\mathbf{n}].1: Partition \mathbf{x} into small blocks of size $m = \ell^2$.2: $c \leftarrow 0$.3: **for** every block \mathbf{m} **do**4: $b \leftarrow$ number of black pixels in block \mathbf{x} [\mathbf{m}].5: **switch** (b)6: **case** $b < m/2$:7: $\mathbf{B} \leftarrow \mathbf{B}_0$.8: **case** $b > m/2$:9: $\mathbf{B} \leftarrow \mathbf{B}_1$.10: **default:**11: $\mathbf{B} \leftarrow \mathbf{B}_c$.12: $c \leftarrow 1 - c$.13: **end switch**14: Randomly permute the columns of \mathbf{B} , and assign i -th row to share block s_i [\mathbf{m}], $i = 1, \dots, n$.15: **end for**

this type from other types of blocks. But if the blocks belonging to \mathbf{M}_1 are scattered randomly in an image, just like the case for halftone image, then the contrast is usually not significant enough in a local region. Locally, one may not be able to distinguish between type \mathbf{M}_0 block and type \mathbf{M}_1 block, since both of them can be encoded by \mathbf{B}_0 .

The algorithm in [6] is also a block encoding algorithm. The image blocks are classified according to their blackness, i.e., the number of black pixels in it. For a block with m pixels, one gets $m + 1$ types of blocks \mathbf{M}_i , $i = 0, 1, \dots, m$. Let \mathbf{B}_0 and \mathbf{B}_1 represent the basis matrices for white and black secret pixels in the basic deterministic VC [10]. When encoding a type- i block, the basis matrix \mathbf{B}_1 is utilized

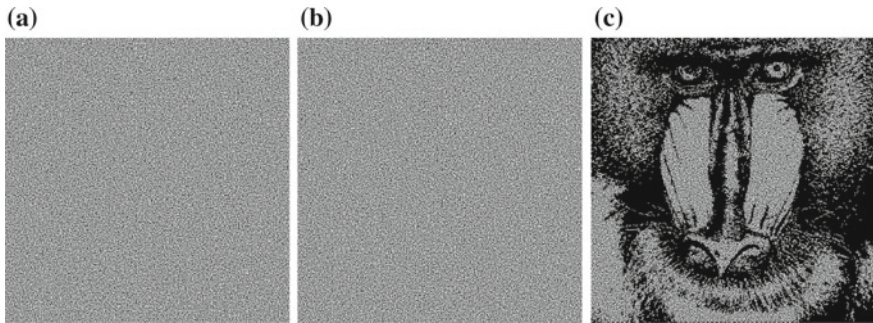


Fig. 6.1 Experimental result for Hou's $(2, 2)$ -threshold block VC, using binary secret image. **a** Share 1. **b** Share 2. **c** Target image

i times and the basis matrix \mathbf{B}_0 is utilized $m - i$ times. So in a ‘local’ neighborhood of m blocks (same type), the contrast between black and white is maintained. Compared to [11] that quantizes b to three levels, the algorithm in [6] doesn’t quantize b , but tries to express all b types of blocks using different frequencies of \mathbf{B}_0 and \mathbf{B}_1 . This algorithm is summarized in Algorithm 17.

Algorithm 17 Block encoding algorithm proposed by Hou and Tu [6].

Input:

Halftone secret image: x [\mathbf{n}].
 Basis matrices $\mathbf{B}_0, \mathbf{B}_1$ for binary (k, n) -threshold VC.

Output:

Share images : s_1 [\mathbf{n}], \dots , s_n [\mathbf{n}].

- 1: Partition \mathbf{x} into small blocks of size $m = \ell^2$.
 - 2: Initialize counter $c_i \leftarrow 0, \forall i \in \{0, \dots, m\}$.
 - 3: **for** every block \mathbf{m} **do**
 - 4: $b \leftarrow$ number of black pixels in block \mathbf{x} [\mathbf{m}].
 - 5: **if** $(c_b \bmod m) < b$ **then**
 - 6: $\mathbf{B} \leftarrow \mathbf{B}_1$.
 - 7: **else**
 - 8: $\mathbf{B} \leftarrow \mathbf{B}_0$.
 - 9: **end if**
 - 10: $c_b \leftarrow c_b + 1$.
 - 11: Randomly permute the columns of \mathbf{B} , and assign the i -th row to share block s_i [\mathbf{m}], $\forall i = 1, \dots, n$.
 - 12: **end for**
-

The experimental result for $(2, 2)$ -threshold case is shown in Figs. 6.1 and 6.2 for binary and halftone secret images, respectively.

If we only look at type b block in secret image for a given b , then for every m such blocks, the first b blocks are encoded using basis matrix \mathbf{B}_1 and the remaining $m - b$ blocks are encoded using basis matrix \mathbf{B}_0 . So the distribution of \mathbf{B}_1 and \mathbf{B}_0 over all type b blocks is not uniform. Liu et al. modified this algorithm and made the

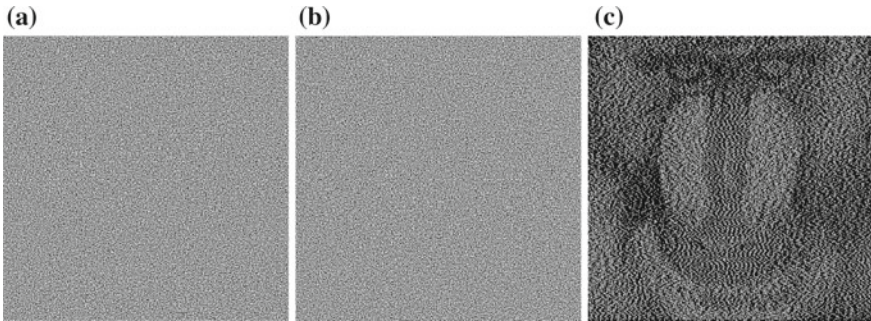


Fig. 6.2 Experimental result for Hou’s $(2, 2)$ -threshold block VC, using halftone secret image. **a** Share 1. **b** Share 2. **c** Target image

distribution of \mathbf{B}_0 and \mathbf{B}_1 more even. In Liu's algorithm [9], for a type b block, one randomly chooses a matrix from $\{\mathbf{B}_1, \mathbf{B}_0\}$ with probabilities

$$\Pr\{\mathbf{B}_1\} = \frac{b}{m}, \quad \Pr\{\mathbf{B}_0\} = 1 - \frac{b}{m}. \quad (6.4)$$

To make sure that for every m type- b blocks, the matrix \mathbf{B}_1 is used exactly b times, the probabilities are modified to include the number of \mathbf{B}_1 already used till now. Let \hat{c}_b be a counter that records the number of \mathbf{B}_1 already used till now for type b . Then the probabilities are:

$$\Pr\{\mathbf{B}_1\} = \frac{b - \hat{c}_b}{m}, \quad \Pr\{\mathbf{B}_0\} = 1 - \frac{b - \hat{c}_b}{m}, \quad (6.5)$$

where $0 \leq \hat{c}_b \leq b$. Using this probabilistic choice of \mathbf{B}_1 and \mathbf{B}_0 , the distribution of usage of these two basis matrices are even. In addition, it can guarantee that for a consecutive m type- b blocks, the matrix \mathbf{B}_1 is used exactly b times. Liu's modified algorithm is summarized in Algorithm 18.

Algorithm 18 Block encoding algorithm proposed by Liu [9].

Input:

Halftone secret image: $x[\mathbf{n}]$.

Basis matrices $\mathbf{B}_0, \mathbf{B}_1$ for binary (k, n) -threshold VC.

Output:

Share images $s_1[\mathbf{n}], \dots, s_n[\mathbf{n}]$.

- 1: Partition \mathbf{x} into small blocks of size $m = \ell^2$.
 - 2: Initialize counter $c_b \leftarrow m, \forall b \in \{0, \dots, m\}$.
 - 3: Initialize counter $\hat{c}_b \leftarrow b, \forall b \in \{0, \dots, m\}$.
 - 4: **for** every block \mathbf{m} **do**
 - 5: $b \leftarrow$ number of black pixels in block $\mathbf{x}[\mathbf{m}]$.
 - 6: $r \leftarrow$ a random integer from $\{0, \dots, c_b - 1\}$.
 - 7: **if** $r < \hat{c}_b$ **then**
 - 8: $\mathbf{B} \leftarrow \mathbf{B}_1$.
 - 9: $c_b \leftarrow c_b - 1$.
 - 10: $\hat{c}_b \leftarrow \hat{c}_b - 1$.
 - 11: **else**
 - 12: $\mathbf{B} \leftarrow \mathbf{B}_0$.
 - 13: $c_b \leftarrow c_b - 1$.
 - 14: **end if**
 - 15: **if** $c_b = 0$ **then**
 - 16: $c_b \leftarrow m$, and $\hat{c}_b \leftarrow b$.
 - 17: **end if**
 - 18: Randomly permute the columns of \mathbf{B} , and assign the i -th row to share block $s_i[\mathbf{m}]$, $\forall i = 1, \dots, n$.
 - 19: **end for**
-

The experimental result for $(2, 2)$ -threshold case is shown in Figs. 6.3 and 6.4 for binary and halftone secret images, respectively.

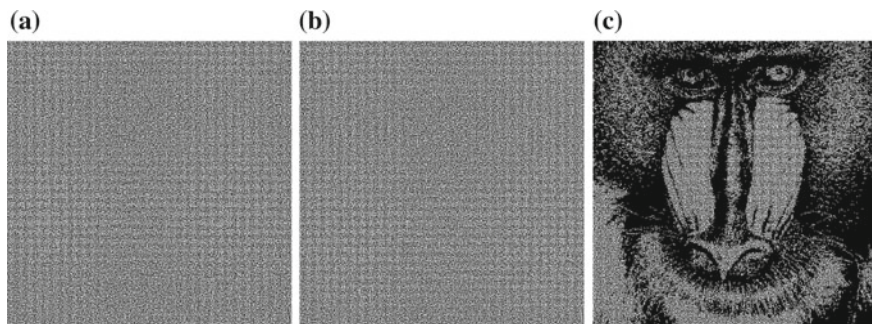


Fig. 6.3 Experimental result for Liu's (2, 2)-threshold block VC, using binary secret image. **a** Share 1. **b** Share 2. **c** Target image

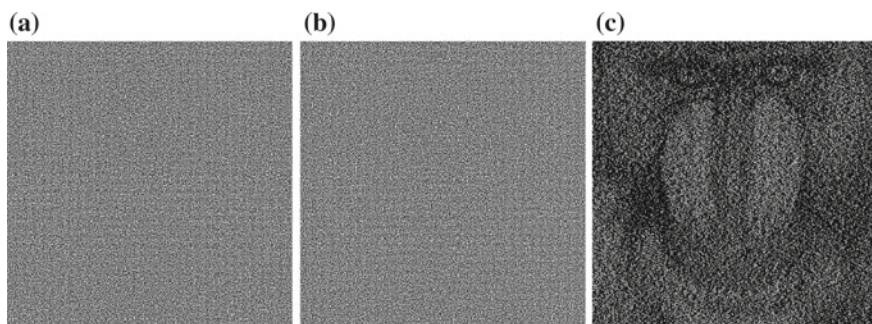
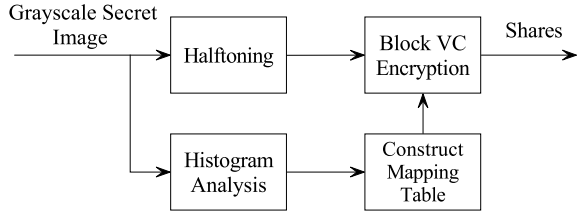


Fig. 6.4 Experimental result for Liu's (2, 2)-threshold block VC, using halftone secret image. **a** Share 1. **b** Share 2. **c** Target image

6.1.2 Lee's Block Encoding Algorithm

Lee's block encoding is also based on block classification. But there are two new features compared with Hou's scheme. The first one is the direct design of mapping from secret image to target image, instead of using the combination of \mathbf{B}_0 and \mathbf{B}_1 . As a result, this mapping can render more gray levels. Second, there is an implicit histogram equalization inside the scheme. Using this equalization, the loss of local contrast can be improved especially for secret image having skewed histogram and low contrast.

The block diagram for Lee's algorithm is shown in Fig. 6.5. The grayscale secret image is first classified to be one of the three types: left-skewed, right-skewed or centered, according to the histogram. Based on this classification, one constructs a mapping table from the secret blocks to the target blocks. The block VC encryption step uses this mapping to execute encryption.

Fig. 6.5 Block diagram for Lee's algorithm

The histogram classification can be done by dividing the domain of the histogram into three equal intervals, and counting the number of pixels in each interval. Let the range of the histogram be $[0, 255]$ for a 8-bit quantized grayscale image, and the histogram be $h(x)$, $x \in [0, 255]$. Dividing this interval to three equal subintervals $l_l = [0, 85]$, $l_c = [86, 170]$, and $l_r = [171, 255]$, then one can calculate:

$$h_l = \sum_{x \in l_l} h(x), \quad h_c = \sum_{x \in l_c} h(x), \quad h_r = \sum_{x \in l_r} h(x). \quad (6.6)$$

A histogram is classified as left-skewed if $h_l = \max\{h_l, h_c, h_r\}$, etc.

According to the type of the histogram, one can also construct a mapping table from secret block to target block to stretch or equalize the histogram of target image (after inverse halftoning). For a 2×2 block, the secret image block belongs to one of the following according to the number of black pixels in it (we use 'xByW' to denote a block with x black pixels and y white pixels).

$$\mathbf{x} \in \{0B4W, 1B3W, 2B2W, 3B1W, 4B0W\}. \quad (6.7)$$

But for target image, there are only three types of blocks

$$\mathbf{z} \in \{2B2W, 3B1W, 4B0W\}. \quad (6.8)$$

To make full use the output blocks, Lee et al. use the mapping as depicted in Fig. 6.6. For example, for a secret image with left-skewed histogram, one uses the mapping in 6.6a. This mapping tries to produce more blocks of type 2B2W on target image. Note that 2B2W is the brightest block that we can produce on target image. So using this mapping, the histogram of the target image is sketched to the right. The histogram (after inverse halftoning) is somewhat equalized.

The above mapping is between secret blocks and target blocks, which should be realized by the VC encryption and stacking. For VC encryption, Lee et al. use the following basis matrices

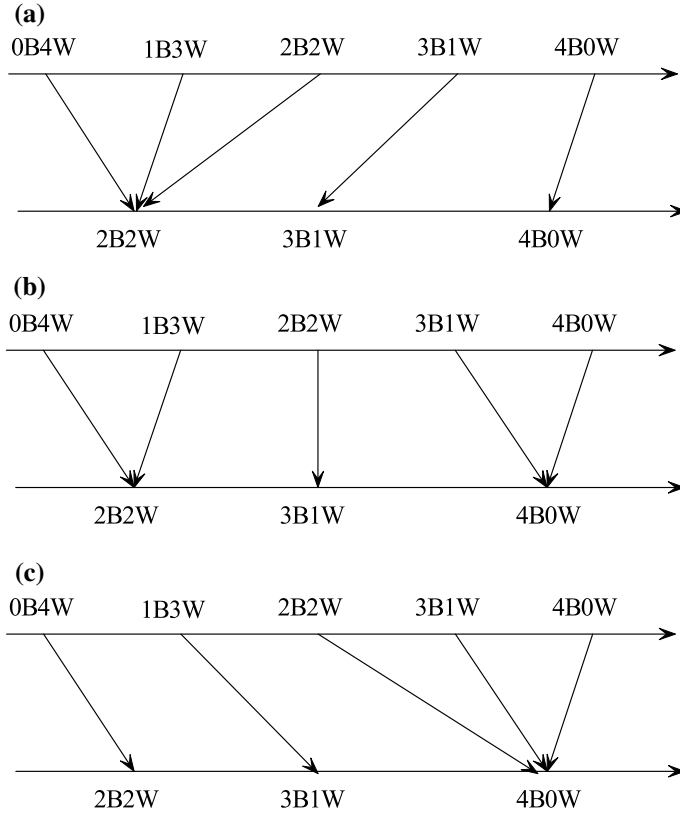


Fig. 6.6 Mapping between secret block and target block in Lee's algorithm. **a** For left-skewed secret image. **b** For centered secret image. **c** For right-skewed secret image

$$\begin{aligned}
 \mathbf{B}_{2B2W} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \\
 \mathbf{B}_{3B1W} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \\
 \mathbf{B}_{4B0W} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},
 \end{aligned} \tag{6.9}$$

where the subscript indicates the pattern that can be produced on target image. For example, for a left-skewed secret image, if the type of the secret block is 2B2W, then according to Fig. 6.6, we should produce a 2B2W on target image. So, we choose the basis matrix \mathbf{B}_{2B2W} , randomly permute the columns of it, and distribute each row to one share.

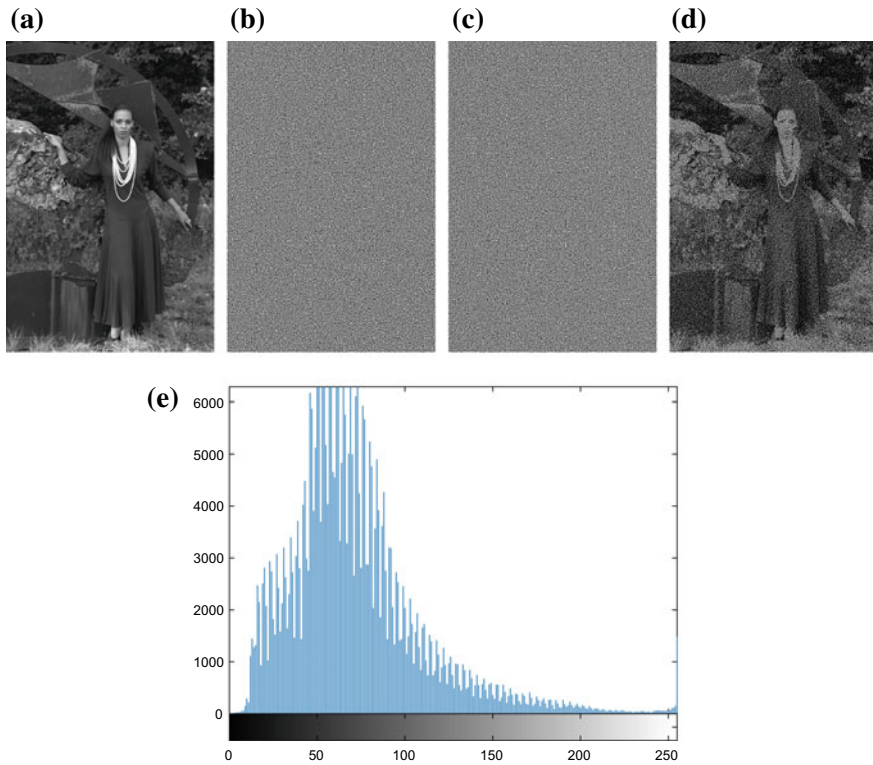


Fig. 6.7 Experimental result of Lee's algorithm on a left-skewed image. **a** Secret image. **b** Share 1. **c** Share 2. **d** Target image. **e** Histogram

The experimental results for Lee's algorithm on a left-skewed image and a centered image are shown in Figs. 6.7 and 6.8. Since the secret image in Fig. 6.7 has a left-skewed histogram, the whole image is biased to darker side. We may notice that the target image in Fig. 6.7d is brighter than Fig. 6.7a, due to the implicit equalization. The secret image in Fig. 6.8 has a centered histogram. The target image in Fig. 6.8d shows similar brightness and contrast as the secret image, due to the balanced mapping in Fig. 6.6b.

6.1.3 Vector VC for Binary Secret Image

Inspired by Lee's work, we can formally define vector VC, which will be improved in AbS-based vector VC. The definition is in Definition 6.1.

Definition 6.1 A (n, n, m) -threshold vector VC is defined by a set of basis matrices $\mathbf{B}_1, \dots, \mathbf{B}_t$ and a mapping \mathcal{F} from $\{0, \dots, m\}$ to

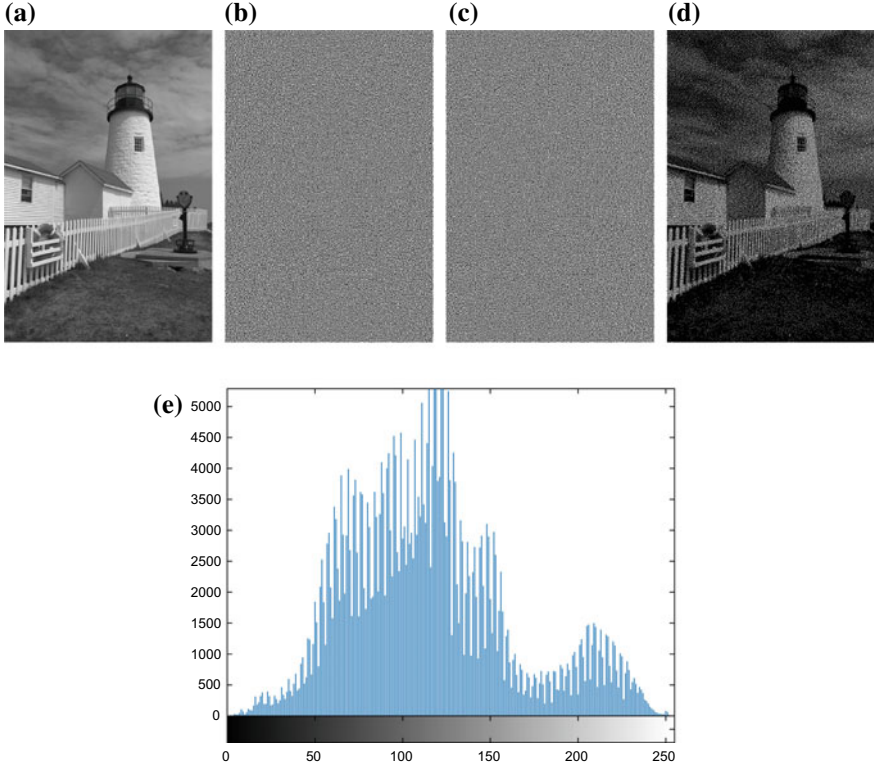


Fig. 6.8 Experimental result of Lee's algorithm on a centered image. **a** Secret image. **b** Share 1. **c** Share 2. **d** Target image. **e** Histogram

$$D = \{d : d = \mathcal{H}(\mathbf{B}_i \llbracket \{1, \dots, n\} \rrbracket), \quad i = 1, \dots, t\},$$

i.e., the set D contains all the possible number of black pixels that can be produced by stacking the n rows of \mathbf{B}_i , $i = 1, \dots, t$. When encoding a block \mathbf{y} having b black pixels, $b \in \{0, \dots, m\}$, one uses the basis matrix $\mathbf{B}_{\mathcal{F}(b)}$. After random permuting the columns, each row is assigned to a corresponding share $\mathbf{s}_1, \dots, \mathbf{s}_n$. The following two conditions should be satisfied.

1. **Contrast condition:** Let $\mathcal{B}(\mathbf{y})$ be the number of black pixels in \mathbf{y} . For any two secret blocks \mathbf{y}_1 and \mathbf{y}_2 with $\mathcal{B}(\mathbf{y}_1) < \mathcal{B}(\mathbf{y}_2)$, the stacking results are $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$, respectively. We should have $\mathcal{B}(\hat{\mathbf{y}}_1) \leq \mathcal{B}(\hat{\mathbf{y}}_2)$.
2. **Security condition:** For any $d < n$ and any d participants $\mathbf{P} = \{i_1, \dots, i_d\}$, $\mathbf{B}_i \llbracket \mathbf{P} \rrbracket \sim \mathbf{B}_j \llbracket \mathbf{P} \rrbracket$, $\forall i, j \in \{1, \dots, t\}$, and $i \neq j$.

Designing the basis matrices for vector VC is quite similar to the designing of basis matrices for grayscale secret image, as discussed in Chap. 5 when introducing Wang's algorithm. The basic idea is concatenation of basis matrices for binary VC [13]. For

example, the matrices in (6.11) is constructed by concatenating the two basis matrices for binary VC:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The vector VC defined in this section is a building block for AbS-based vector VC.

6.2 Local Blackness Preserving Visual Cryptography

The basic idea of local blackness preserving VC is to apply AbS framework to halftone VC. The loss of contrast due to VC encryption and stacking is distributed to and compensated by neighboring blocks. To improve contrast, we first equalize the grayscale secret image and then use error diffusion algorithm to obtain a halftone image. Then the blackness preserving VC is applied to this halftone image.

6.2.1 VC Encryption and Local Blackness Preservation

Let $x[\mathbf{n}]$ be the halftone secret image, obtained from equalizing and halftoning a grayscale image. Let $z[\mathbf{n}]$ be the target image by stacking all the shares $s_1[\mathbf{n}]$ and $s_2[\mathbf{n}]$. Our purpose is to ensure that in a small region of $z[\mathbf{n}]$, the ratio between black pixels and white pixels is the same as that of in the same region of $x[\mathbf{n}]$. To attain this goal, we take a small block of size $K \times K$ in $x[\mathbf{n}]$, say \mathbf{x} , and encode it into shares. But after stacking the shares, the corresponding block \mathbf{z} may have different blackness than \mathbf{x} due to the loss of contrast in VC. Let's use $\mathcal{B}(\mathbf{x})$ to measure the number of black pixels in a block \mathbf{x} , and let \mathbf{x} and \mathbf{z} be two corresponding blocks on secret image and target image, respectively. If $\mathcal{B}(\mathbf{z}) - \mathcal{B}(\mathbf{x}) = c$, where $c > 0$, then it means that the target block \mathbf{z} has more black pixels than the corresponding secret block \mathbf{x} . To compensate for this increase of black pixels in current block, we borrow c black pixels from the neighboring blocks. By doing this, we can preserve the local blackness of the target image. The vector VC part is similar to the vector VC in last section (Definition 6.1).

The key step in this algorithm is blackness compensation. An example is shown in Fig. 6.9 to illustrate the basic idea. The current block (marked with a dash rectangle) has blackness 1 (Fig. 6.9a). But after VC encoding and stacking, the blackness of the stacking result is 2 (Fig. 6.9b). If we leave the neighbors unchanged, then the local blackness on the stacked image will be higher than that of the original halftone image. To preserve the local blackness, one black pixel is borrowed from one of the neighboring blocks. As to which black pixels to borrow, one natural way is to borrow from a block with more black pixels. In this example, we see that the block right below the current block has four black pixels. So we borrow one black pixel from

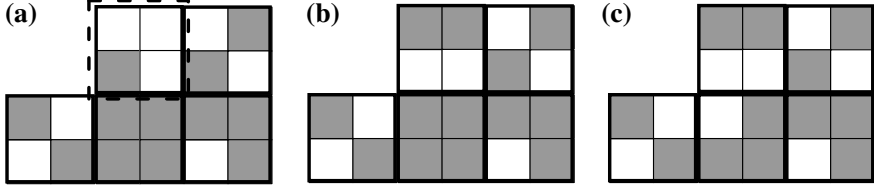


Fig. 6.9 An example of local blackness preservation. **a** A neighborhood having five blocks, where the current block is marked with dashed rectangle. **b** After VC encoding and stacking of current block. **c** After borrowing one black pixel from a neighboring block, the one right below the current block

this block by flipping one of its black pixels to white. Now, the current block has two black pixels, and the block right below it has three black pixels. But the total number of pixels in the whole neighborhood remains the same.

In this algorithm, we consider neighborhood system that is similar to those in the error diffusion. The difference is that here each neighbor is a $K \times K$ block, while in error diffusion, each neighbor is a pixel. After flipping a black pixel in a neighboring block, we make the total number of black pixels unchanged in this small neighborhood. In general, we may need to flip more than one black pixels in this neighborhood, depending on the difference $\mathcal{B}(\mathbf{z}) - \mathcal{B}(\mathbf{x})$.

Let's use $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \text{VcEnc}(\mathbf{B})$ to denote VC encoding using a basis matrix \mathbf{B} in a deterministic VC. The first row of \mathbf{B} is distributed to Share 1 and is then reshaped as a block \mathbf{s}_1 . The second row of \mathbf{B} is distributed to Share 2 and is then reshaped as a block \mathbf{s}_2 .

We also need a bit flipping algorithm, which receives the four neighboring blocks and flip one black pixel in one of the blocks having the largest number of black pixels. Let this algorithm be denoted as $(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4) \leftarrow \text{BitFlipping}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4)$. From the four neighboring blocks $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$, we find the one having the maximum number of black pixels:

$$k = \arg \max \{ \mathcal{B}(\mathbf{z}_i), i \in \{1, 2, 3, 4\} \}. \quad (6.10)$$

If there is no black pixels to borrow, i.e., if

$$\max \{ \mathcal{B}(\mathbf{z}_i), i \in \{1, 2, 3, 4\} \} = 0,$$

we return the original blocks: $\hat{\mathbf{z}}_i = \mathbf{z}_i, i = 1, 2, 3, 4$. Otherwise, we flip an arbitrary black pixel in block \mathbf{z}_k .

Let the three basis matrices used in vector VC be:

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \mathbf{B}_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \mathbf{B}_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad (6.11)$$

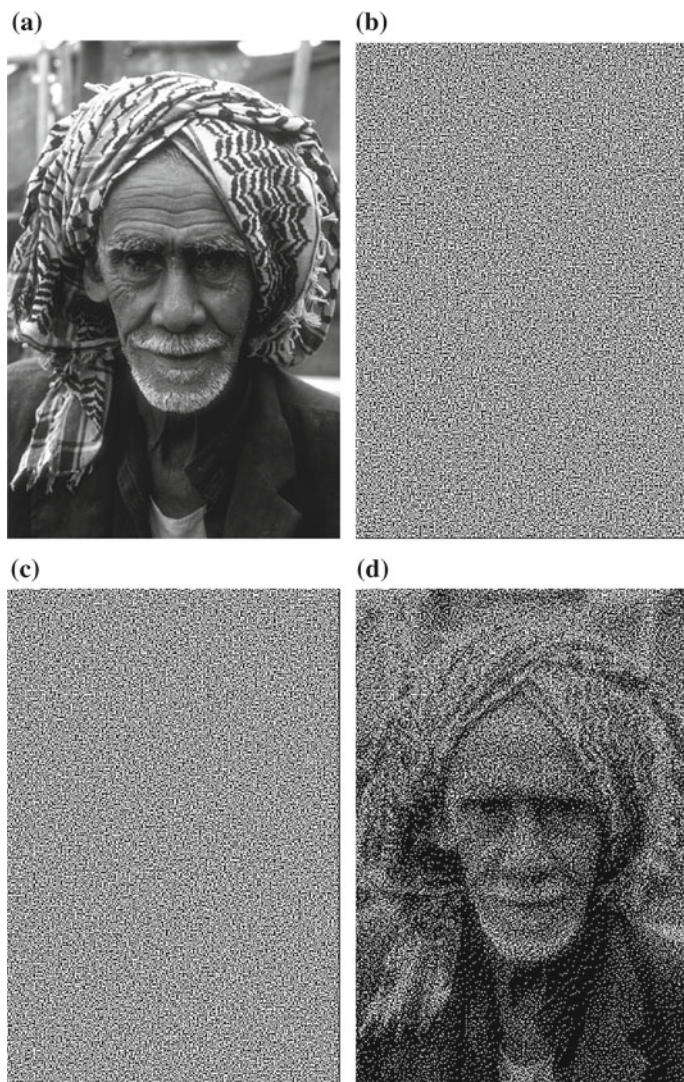


Fig. 6.10 Experimental result of local blackness preserving VC algorithm on a grayscale image. **a** Secret image. **b** Share1. **c** Share 2. **d** Target image

where the subscript describes the number of black pixels that can be produced by stacking the two rows. Using the two algorithms **VcEnc** and **BitFlipping** as building blocks, we can formally describe the local blackness preserving algorithm, as shown in Algorithm 19.

Algorithm 19 Local blackness preserving VC encoding [12].**Input:**

Halftone secret image $x[n] \in \mathbb{Z}_2^{M \times N}$.

Output:

Share images: $s_1[n] \in \mathbb{Z}_2^{M \times N}$, $s_2[n] \in \mathbb{Z}_2^{M \times N}$.

```

1: for every block  $\mathbf{m}$  do
2:    $N_{\mathbf{m}} \leftarrow$  block indices for the neighboring blocks of block  $\mathbf{m}$ .
3:   if  $\mathcal{B}(x[\mathbf{m}]) = 4$  then
4:      $(s_1[\mathbf{m}], s_2[\mathbf{m}]) \leftarrow \text{VcEnc}(\mathbf{B}_4)$ .
5:   else if  $\mathcal{B}(x[\mathbf{m}]) = 3$  then
6:      $(s_1[\mathbf{m}], s_2[\mathbf{m}]) \leftarrow \text{VcEnc}(\mathbf{B}_3)$ .
7:   else if  $\mathcal{B}(x[\mathbf{m}]) = 2$  then
8:      $(s_1[\mathbf{m}], s_2[\mathbf{m}]) \leftarrow \text{VcEnc}(\mathbf{B}_2)$ .
9:   else if  $\mathcal{B}(x[\mathbf{m}]) = 1$  then
10:     $(s_1[\mathbf{m}], s_2[\mathbf{m}]) \leftarrow \text{VcEnc}(\mathbf{B}_2)$ .
11:     $\{x_k, k \in N_{\mathbf{m}}\} \leftarrow \text{BitFlipping}(\{x_k, k \in N_{\mathbf{m}}\})$ .
12:   else if  $\mathcal{B}(x[\mathbf{m}]) = 0$  then
13:     $(s_1[\mathbf{m}], s_2[\mathbf{m}]) \leftarrow \text{VcEnc}(\mathbf{B}_2)$ .
14:     $\{x_k, k \in N_{\mathbf{m}}\} \leftarrow \text{BitFlipping}(\{x_k, k \in N_{\mathbf{m}}\})$ .
15:     $\{x_k, k \in N_{\mathbf{m}}\} \leftarrow \text{BitFlipping}(\{x_k, k \in N_{\mathbf{m}}\})$ .
16:   end if
17: end for
18: Insert the share blocks  $s_1[\mathbf{m}]$ ,  $s_2[\mathbf{m}]$  into share images  $s_1[n]$  and  $s_2[n]$  respectively.

```

The experimental result for the test image ‘man’ is shown in Fig. 6.10. The HPSNR is found to be 19.84 and the MSSIM is 0.62. More comprehensive testing can be found from [12].

6.3 AbS-Based Vector VC

The¹ local blackness preserving VC is based on halftone secret image, not the original grayscale secret image. To better represent the grayscale image on target image, one should use the grayscale image directly. From previous two sections, we also see that vector VC can preserve features of the secret block on target block, hence is able to render details of the secret image. By combining the vector VC and AbS framework, we can develop an AbS-based vector VC, so that we may get target image having higher fidelity with the secret image.

The block diagram for vector VC is shown in Fig. 6.11. It consists of three major steps: Pre-processing, Analysis and Synthesis.

1. The *pre-processing* performs gamut mapping and block partition. Recall that using vector VC, the output gamut is usually smaller than the input gamut,

¹© 2019 IEEE. Reprinted, with permission, from IEEE Transactions on Image Processing, Vol. 28, No. 2, Improving the Visual Quality of Size-Invariant Visual Cryptography for Grayscale Images: An Analysis-by-Synthesis (AbS) Approach, Bin Yan et al.

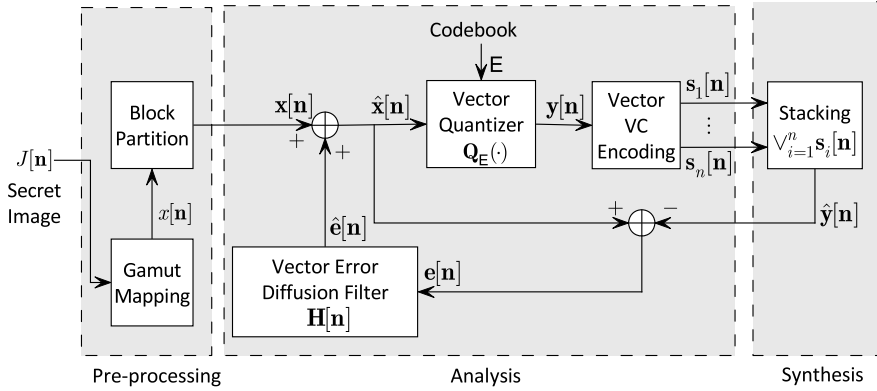


Fig. 6.11 Block diagram for AbS-based vector VC

in terms of the average block blackness. So one must restrict the input to the encryption algorithm to ensure the stability of the error diffusion loop [4]. The block partition part split the secret image into small blocks. Each block is a coding unit and is encrypted by vector VC.

2. The *analysis* step generates shares from pre-processed secret image. The modified secret image block $x[n]$ is quantized by a vector quantizer to generate a binary block. Then we use vector VC to encode the binary block into share blocks $s_i[n]$, $i = 1, \dots, n$.
3. The *synthesis* step simulates the printing, stacking and human perception process to generate a simulated target image $\hat{y}[n]$. Since the printing model and HVS model can be incorporated into the diffusion filter, it is not shown in the synthesis step.

The design of other component depends on the choice of vector VC. The definition and construction of vector VC are introduced in Sect. 6.1.3. The steps for implementing it is summarized in Algorithm 20, for the particular case of the basis matrices defined in (6.11).

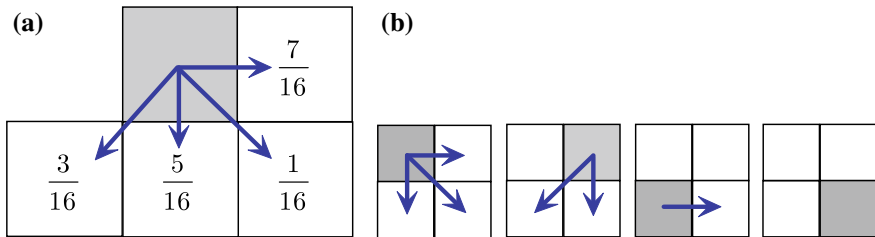
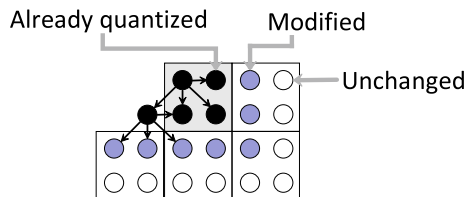
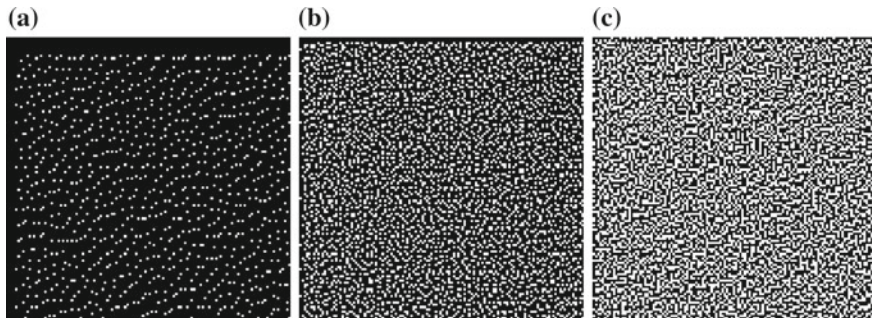


Fig. 6.12 Vector quantization by inner block error diffusion. **a** Floyd-Steinberg error diffusion kernel. **b** Inner block error diffusion

Algorithm 20 Vector VC encoding.**Input:**Binary secret block \mathbf{y} .Basis matrices $\mathbf{B}_k, k \in \{2, 3, 4\}$, as defined in (6.11).**Output:**Share blocks: $\mathbf{s}_1, \dots, \mathbf{s}_n$ 1: **switch** ($\mathcal{B}(\mathbf{y})$)2: **case** 4:3: $\mathbf{B} \leftarrow \mathbf{B}_4$.4: **case** 3:5: $\mathbf{B} \leftarrow \mathbf{B}_4$.6: **case** 2:7: $\mathbf{B} \leftarrow \mathbf{B}_3$.8: **case** 1:9: $\mathbf{B} \leftarrow \mathbf{B}_2$.10: **case** 0:11: $\mathbf{B} \leftarrow \mathbf{B}_2$.12: **end switch**13: Distribute the first row of row-permuted \mathbf{B} to share block \mathbf{s}_1 , and second row to \mathbf{s}_2 .**Fig. 6.13** Vector error diffusion by pushing error forward**Fig. 6.14** Target image from AbS-based vector VC, where the secret image is a constant image $J[\mathbf{n}] = g$. **a** $g = 7/8$. **b** $g = 1/2$. **c** $g = 1/8$

The vector quantizer receives a of grayscale image block, and produces a binary block. Given the statistics of the input block, we may be able to design an optimal vector quantizer. Alternatively, given enough training samples (image blocks), one can use the well-known LBG algorithm to train a codebook [5]. But these train-

ings are time-consuming and are not easily generalized. We use an adaptive scalar quantization approach. This adaptive quantizer is realized by a constant threshold quantizer and error diffusion. For example, for a 2×2 block from image \mathbf{x} , where the upper-left corner is position $[i, j]$. We use the constant threshold $1/2$ for this sample, and get the quantization output $\hat{x}[i, j]$. After quantizing $x[i, j]$, the error $e[i, j] = x[i, j] - \hat{x}[i, j]$ is distributed to positions $[i, j + 1]$, $[i + 1, j]$, $[i + 1, j + 1]$ and modify the samples at these positions. For position $[i, j + 1]$, the error is only distributed to $[i + 1, j]$, $[i + 1, j - 1]$ within the same block. This is referred to as inner block diffusion, as shown in Fig. 6.12.

The vector error diffusion diffuses error of current block to adjacent blocks. Recall that, in an AbS approach, this error is the error between quantizer input and target image. One of the well-developed vector error diffusion is proposed by Venkata and Evan [3]. The basic idea is to first distribute the total error among blocks according to the Floyd-Steinberg's diffusion, and then distribute the allocated error to a block evenly to all pixels in that block. Such an even distribution within a block is advantageous when one intends to produce clustered dots. In this work, however, the intention is to produce scattered dots for better visual quality. So we use sample-wise error diffusion, as illustrated in Fig. 6.13. Since the current block represents the target

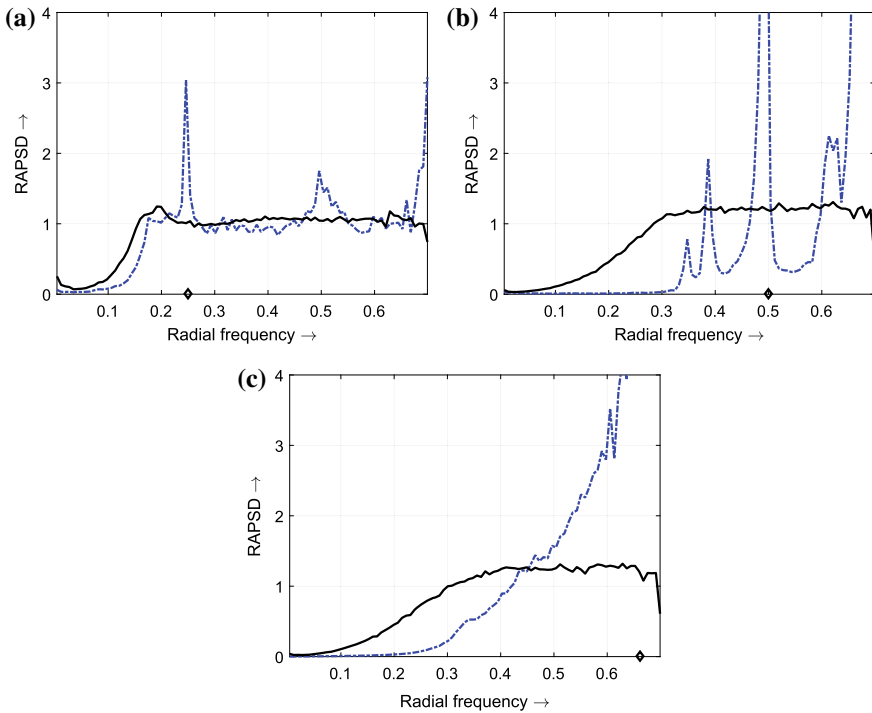


Fig. 6.15 RAPSD from AbS-based vector VC, where the secret image is a constant image $J[\mathbf{n}] = g$. Dashed curves show the RAPSD for halftone secret image. Solid curves show the RAPSD for AbS-based vector VC. **a** $g = 7/8$. **b** $g = 1/2$. **c** $g = 1/8$

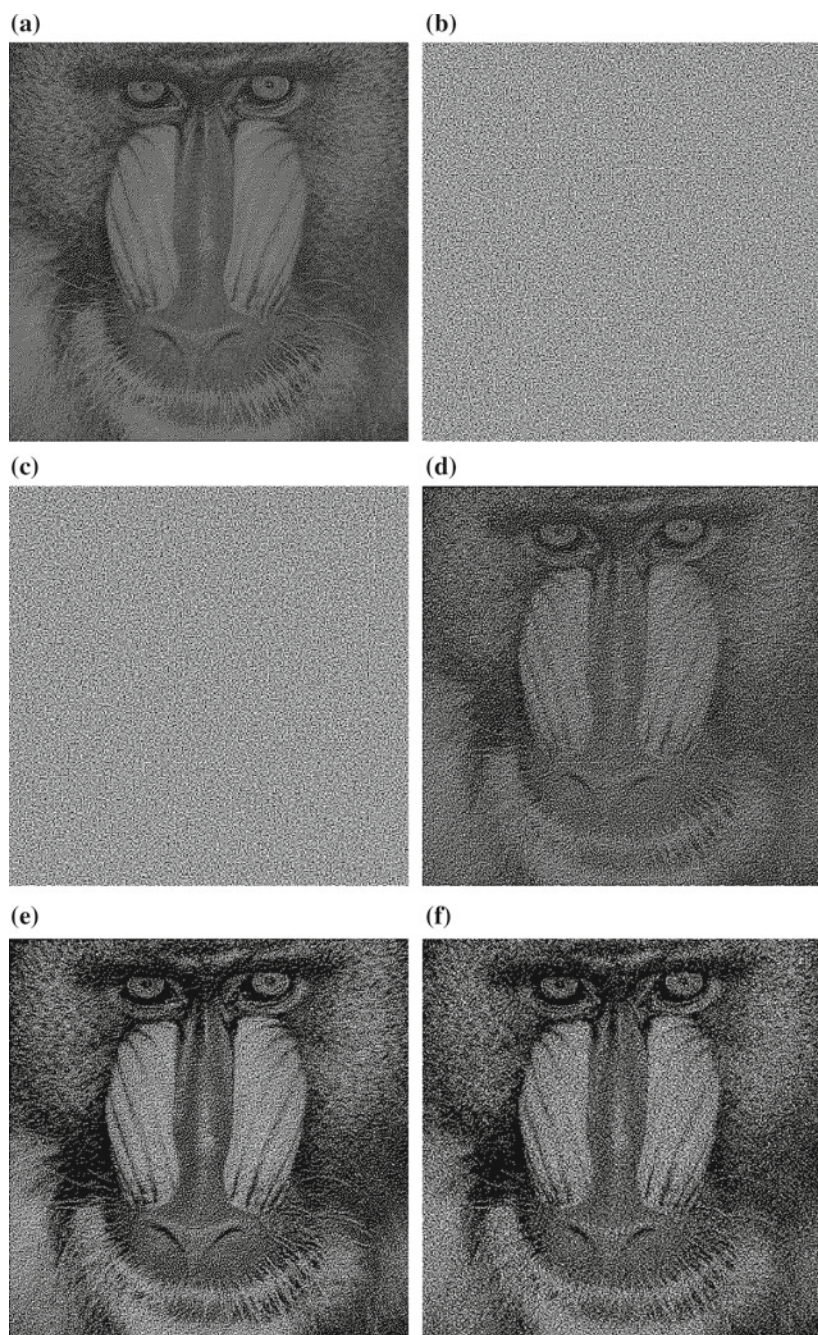


Fig. 6.16 Experimental result for $(2, 2)$ -threshold AbS-based vector VC, using Baboon image. **a** Halftone of $J[n]/2$. **b** Share 1 (no equalization). **c** Share 2 (no equalization). **d** Target image without equalization. **e** Target image of vector VC combined with equalization. **f** Target image using Lee's algorithm

block, it is not possible to modify pixels in this block. The errors for all pixels in the current block are distributed to nearby pixels in adjacent blocks.

Testing this algorithm on constant secret image, we obtain the results as shown in Figs. 6.14 and 6.15. The black dots are evenly distributed on target image. The result for RAPSD shows that, the AbS-based vector VC produces near zero low frequency components.

For fidelity test, we use the Baboon image. The testing result is shown in Fig. 6.16. The target image from AbS-based vector VC without equalization (Fig. 6.16d) is visually close to the halftone result in Fig. 6.16a. Comparing the result in Figs. 6.16e and 6.16f, we see that, by combining equalization with the AbS-based vector VC, one can render more details on target image than Lee's algorithm. For example, the fine structure of the beard is better rendered in Fig. 6.16e.

In addition to the visual inspection, the HPSNR and MSSIM also show improved performance than Hou's algorithm [6] and Liu's algorithm [9], on the Kodak test database [13].

6.4 Remarks

Instead of encoding each secret pixel independently, block VC treats each secret block as one encoding unit. Different types of blocks with different features are encoded using different basis matrices, in a hope that these features are transfer to target image. So, using block encoding and vector VC, the structure of the secret image can be better utilized than sample based VC.

We should note that, even though the AbS-based approach gets better visual quality for the target image, but currently it is limited to (n, n) -threshold VC. The reason is that the encoder needs to know the stacking results. For the more general (k, n) -threshold VC, there are

$$\sum_{i=k}^n \binom{n}{i}$$

different ways to stack the shares. Extending the AbS-based vector from (n, n) to (k, n) , or more general access structure, is an interesting research direction for improving visual quality in VC.

References

1. Chen, Y.F., Chan, Y.K., Huang, C.C., Tsai, M.H., Chu, Y.P.: A multiple-level visual secret-sharing scheme without image size expansion. *Inf. Sci.* **177**(21), 4696–4710 (2007)
2. Chow, Y.W., Susilo, W., Wong, D.S.: Enhancing the perceived visual quality of a size invariant visual cryptography scheme. In: *Proceedings of the 14th International Conference on Information and Communications Security (ICICS'12)*, pp. 10–21. Hong Kong, China (2012)

3. Damera-Venkata, N., Evans, B.L.: FM halftoning via block error diffusion. In: Proc. Int. Conf. Image Process. **2**, 1081–1084. Thessaloniki, Greece (2001)
4. Escbbach, R., Fan, Z., Knox, K.T., Marcu, G.: Threshold modulation and stability in error diffusion. IEEE Signal Process. Mag. **20**(4), 39–50 (2003)
5. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer Academic Publishers, Norwell, MA, USA (1991)
6. Hou, Y.C., Tu, S.F.: Visual cryptography techniques for color images without pixel expansion (in Chinese). J. Inf. Technol. Soc. **4**(1), 95–110 (2004)
7. Hou, Y.C., Tu, S.F.: A visual cryptographic technique for chromatic images using multi-pixel encoding method. J. Res. Pract. Inf. Technol. **37**(2), 179–191 (2005)
8. Lee, C.C., Chen, H.H., Liu, H.T., Chen, G.W., Tsai, C.S.: A new visual cryptography with multi-level encoding. J. Vis. Lang. Comput. **25**(3), 243–250 (2014)
9. Liu, F., Guo, T., Wu, C., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. J. Vis. Commun. Image Represent. **23**(2), 331–342 (2012)
10. Naor, M., Shamir, A.: Visual cryptography. In: Proceeding of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT’94), pp. 1–12. Perugia, Italy (1994)
11. Tu, S.F., Hou, Y.C.: Design of visual cryptographic methods with smooth looking decoded images of invariant size for grey-level images. Imaging Sci. J. **55**(2), 90–101 (2007)
12. Yan, B., Chen, N., Yang, H.M., Hao, J.J.: Local blackness preserving visual cryptography for grayscale secret images. J. Inf. Hiding Multimed. Signal Process. **9**, 370–382 (2018)
13. Yan, B., Xiang, Y., Hua, G.: Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (AbS) approach. IEEE Trans. Image Process. **28**(2), 896–911 (2019)

Chapter 7

Conclusion and Future Works



7.1 Summary and Conclusion

Focusing on the strict sense visual cryptography using ‘OR’ operation in stacking and grayscale secret image, this book comprehensively reviews important efforts in improving visual quality of the target image.

After a short introduction to the history of VC, we introduced several practical applications of VC, including online transaction security, privacy protection, and electronic election, etc. From these applications, the quality of the target image arises as a key problem to solve, especially for grayscale image with no pixel expansion. In Chap. 2, We comprehensively reviewed the three types of classical VC for binary image, namely, the deterministic VC, the probabilistic VC and the random grid VC. The notions of weak security are also reviewed, which provide possible tradeoff between security and perceptual quality. To move to the grayscale images, in Chap. 3, we reviewed digital halftoning techniques, including bi-level quantization, ordered dithering, error diffusion and DBS. Some quality metrics are also introduced to guide our design of quality improvement VC algorithms, including RAPSD, HPSNR, MSSIM and residual variance. Equipped with these background knowledge, we then spent three chapters introducing important algorithms for improving visual quality. In Chap. 4, we reviewed algorithms for improving the visual quality of the share images in extended VC. In Chap. 5, we reviewed algorithms for improving visual quality of probabilistic and random grid VC. In Chap. 6, we reviewed algorithms for improving visual quality of vector VC using block-wise encoding.

From these comprehensive review, we note that designing VC for grayscale secret image is quite different from designing VC for binary image. Several conclusions can be drawn from the experimental results.

1. Sample-wise encoding that is used for binary secret image usually provides inferior result on grayscale image than block encoding.
2. For grayscale and halftone secret image, one should use perceptual quality measures to evaluate the quality of the target image. Furthermore, a HVS model

should be incorporated into the VC algorithms to better address the perceptual quality.

3. Instead of focusing on a single secret pixel, the VC encoding algorithms that consider local features of grayscale image and map these features directly to the target image may produce better perceptual quality.

7.2 Future Works

Based on the above applications, review and conclusions, there are a few possible future directions that may deserve more attentions.

7.2.1 Block Encoding

We saw that block encoding can produce target image with better perceptual quality. A vector VC is formally defined in Chap. 6. But we notice that there are few works in formalizing the notion of vector VC: clear definition, construction and vigorous proof of security and perceptual quality. Hopefully, by clearly defining vector VC and optimal construction, we may explore the theoretical limit of this approach.

For block encoding, it can map features of the secret block to features of the target block. But currently, this feature is the ratio of black pixels. If more local features can be utilized and preserved by this mapping on target image, then we may further improve the quality of the target image.

7.2.2 Color VC

The notion of weak securities suggested by Iwamoto and Liu provides another possibility to improve the perceptual quality, especially for color secret image [5, 8]. However, Iwamoto's work is limited to simple color patterns. Extending the color image to color image of natural scene should be an interesting topic to explore.

Current efforts in VC for color image are limited to RGB and CMY color spaces and per-channel encoding [3, 4, 6, 9, 11]. This usually requires to print one transparency for each color channel. Developing a design framework that can encode directly in a three-dimensional color space and produce only one transparency should be a useful work, but it may not be trivial.

7.2.3 VC for QR Code

The marriage between VC and QR code is a natural choice, considering that QR code is so widely used nowadays as an information carrier. Current work is limited to utilizing the error correction capabilities of the QR code to hide a secret QR code. But this may damage the error correction margin that the QR code is designed to have. Utilizing other redundancies in QR code should be a possible direction to explore, including redundancy of un-utilized code words and redundancy of colors.

Utilizing the color QR code to design color VC is also a topic worth exploring [1, 2]. Using a color space, more freedom may exist that one can utilize to improve the perceptual quality.

7.2.4 Grayscale Secret and Cover Images

Currently, it is very difficult to use both grayscale secret image and cover images. For grayscale secret image, the shares are limited to meaningless ones. For extended VC with meaningful shares, the secret image is binary. By relaxing security level, or using color encoding, is it possible to use both grayscale secret image and cover image in a ‘OR’ operation-based VC? This is still not solved.

However, we notice that, for watermarking in halftone image, this is possible, such as [10]. But the watermarking system puts more priorities on perceptual quality, so that the security condition is very weak. This suggests that, if in a special scenario where the security level can be compromised, then it is possible to further improve the visual quality of the target image. The key is to find an application-specified tradeoff.

7.2.5 Tradeoff Between Security and Perceptual Quality

The strict sense VC with strong security and the halftone watermarking in [10] represent two ends of the security-quality tradeoff. Theoretically investigating the tradeoff between security and perceptual quality should be conducted to establish a foundation for such tradeoff. Using such a theoretical result, the designer may be well-aware that how much security should be sacrificed for a given quality improvement.

7.2.6 Printer Model and HVS Model

One of the advantage of AbS-based approach is that one can combine the printer model and HVS model into the algorithm explicitly. For the AbS approach discussed

in this book and in [12], these two models are implicitly combined, by using a low-pass diffusion kernel.

To further develop Abs-based approach, one may use HVS models, such as alpha-stable model or mixture of Gaussian model [7]. Furthermore, one may develop a DBS-like algorithm to directly optimization a target function that characterizes the perceptual distortion between the secret image and the target image.

To end this chapter, we note that the above future research problems are only problems directly related to the central topic of this book. Visual cryptography is a fascinating research field, and more fruitful results are emerging inspired by new applications in multimedia security.

References

1. Andr, P.S., Ferreira, R.A.S.: Colour multiplexing of quick-response (QR) codes. *Electron. Lett.* **50**(24), 1828–1830 (2014)
2. Blasinski, H., Bulan, O., Sharma, G.: Per-colorant-channel color barcodes for mobile applications: an interference cancellation framework. *IEEE Trans. Image Process.* **22**(4), 1498–1511 (2013)
3. Hou, Y.C.: Visual cryptography for color images. *Pattern Recognit.* **36**, 1619–1629 (2003)
4. Hou, Y.C., Tu, S.F.: Visual cryptography techniques for color images without pixel expansion (in Chinese). *J. Inf. Technol. Soc.* **4**(1), 95–110 (2004)
5. Iwamoto, M.: A weak security notion for visual secret sharing schemes. *IEEE Trans. Inf. Forensics Secur.* **7**(2), 372–382 (2012)
6. Jin, D., Yan, W.Q., Kankanhalli, M.S.: Progressive color visual cryptography. *J. Electron. Imaging* **14**(3), 1–13 (2005)
7. Lau, D., Arce, G.: *Modern Digital Halftoning*, 2nd edn. Taylor & Francis, Boca Raton, FL (2001)
8. Liu, F., Guo, T., Wu, C., Qian, L.: Improving the visual quality of size invariant visual cryptography scheme. *J. Vis. Commun. Image Represent.* **23**(2), 331–342 (2012)
9. Lou, D.C., Chen, H.H., Wu, H.C., Tsai, C.S.: A novel authenticatable color visual secret sharing scheme using non-expanded meaningful shares. *Displays* **32**, 118–134 (2011)
10. Myodo, E., Takagi, K., Miyaji, S., Takishima, Y.: Halftone visual cryptography embedding a natural grayscale image based on error diffusion technique. In: *Proceeding of the IEEE International Conference on Multimedia and Expo.*, pp. 2114–2117. Beijing, China (2007)
11. Wang, D.S., Yi, F., Li, X.B.: Probabilistic visual secret sharing schemes for grey-scale images and color images. *Inf. Sci.* **181**(11), 2189–2208 (2011)
12. Yan, B., Xiang, Y., Hua, G.: Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (Abs) approach. *IEEE Trans. Image Process.* **28**(2), 896–911 (2019)