An Extended Visual Cryptography Algorithm for General Access Structures

Kai-Hui Lee and Pei-Ling Chiu

Abstract—Conventional visual secret sharing schemes generate noise-like random pixels on shares to hide secret images. It suffers a management problem, because of which dealers cannot visually identify each share. This problem is solved by the extended visual cryptography scheme (EVCS), which adds a meaningful cover image in each share. However, the previous approaches involving the EVCS for general access structures suffer from a pixel expansion problem. In addition, the visual cryptography (VC)-based approach needs a sophisticated codebook design for various schemes. In this paper, we propose a general approach to solve the above-mentioned problems; the approach can be used for binary secret images in noncomputer-aided decryption environments. The proposed approach consists of two phases. In the first phase, based on a given access structure, we construct meaningless shares using an optimization technique and the construction for conventional VC schemes. In the second phase, cover images are added in each share directly by a stamping algorithm. The experimental results indicate that a solution to the pixel expansion problem of the EVCS for GASs is achieved. Moreover, the display quality of the recovered image is very close to that obtained using conventional VC schemes.

Index Terms—Extended visual cryptography (EVC), general access structures, optimization, pixel expansion, visual secret sharing scheme.

I. INTRODUCTION

VISUAL cryptography (VC), which was proposed by Naor and Shamir, allows the encryption of secret information in the image form [1]. By applying the concept of secret sharing, a secret image can be encrypted as \( n \) different share images printed on transparencies, which are then distributed to \( n \) participants. By stacking \( n \) transparencies (shares) directly, the secret images can be revealed and visually recognized by humans without any computational devices and cryptographic knowledge. On the other hand, any one share or a portion of shares can leak nothing related to the secret image. VC is a very good solution for sharing secrets when computers cannot be employed for the decryption process.

In the past decade, many research results on the threshold visual secret sharing scheme (also known as \( k \)-out-of-\( n \) VSS scheme or \( (k,n) \)-VSS scheme) have been proposed [2]–[9]. Ateniese et al. (denoted as Ateniese in short) [10] proposed the concept of general access structure (GAS) and also developed a VC-based solution for some GASs. Afterward, Hsu et al. reported the formulation of an unexpanded VCS for a GAS problem as an optimization model [11], [12]. Liu et al. (denoted as Liu in short) proposed a recursive approach to construct VCS for GASs [13]. By using the GAS, dealers can define reasonable combinations of shares as decryption conditions rather than specifying the number of shares. For example, if there are four participants—one CEO, one manager, and two employees—sharing a secret, the CEO may expect to decrypt the secret with any one colleague who holds one of the other shares. The manager is allowed to obtain the secret with only two employees. The two employees are restricted access to the secret. Due to these flexibilities, dealers can also set the number of shares as the decrypting condition. Hence, the \( (k,n) \)-VSS scheme can be treated as a special case of the GAS.

Conventional VSS schemes generate noise-like random pixels on shares to hide secret images. In this manner, the secret can be perfectly concealed on the share images. However, these schemes suffer from a management problem—dealers cannot identify each share visually. Hence, researchers have developed the extended visual cryptography scheme (EVC [14], also known as the friendly VC scheme [15], [16]), which adds a meaningful cover image on each share to address the management problem. Ateniese presented a general technique to implement \((k,k)\)-threshold EVCS as well as various interesting classes of access structures for binary secret images [14]. Fang [15] and Chen et al. [16] proposed VC-based and random-grid-based techniques, respectively, for \((k,k)\)-EVCS with a progressive decryption effect. Wang et al. developed a matrix extension algorithm for \((k,n)\)-EVCS by modifying any existing VCS with random-looking shares, which were then utilized as meaningful shares [17].

The pixel expansion problem is a common disadvantage with most of the VSS schemes. When the VC-based approach is employed, each secret pixel within a secret image is encrypted in a block consisting of \( mn \) subpixels in each constituent share image. Thus, the area of a share is \( mn \) times that of the original secret image. The contrast of the recovered images will be decreased to \( 1/mn \) simultaneously. The pixel expansion problem not only affects the practicability of storage/transmission requirements for shares but also decreases the contrast of the recovered secret images. To the best of our knowledge, the existing EVCS algorithms for GASs cannot avoid the pixel expansion problem. Therefore, we are motivated to find a solution to this problem.

In this paper, we propose a novel encryption algorithm of EVCS for general access structures to cope with the pixel expansion problem. The proposed algorithm is applicable to binary...
secret/cover images, and no computational devices are needed during the decryption phase. In order to avoid pixel expansion, we do not adopt the traditional VC-based approach to encrypt secret images. The encryption process can be divided into two phases. The first phase of the algorithm, which uses optimization techniques for a given access structure, constructs a set of noise-like shares that are pixel-expansion-free. We identified and formulated the problem in this phase as a combinatorial optimization problem and then developed a simulated-annealing-based algorithm to solve it. The second phase of the algorithm directly adds a cover image on each share via a stamping algorithm. In this manner, the pixel expansion can be removed entirely. Recently, Weir and Yan also used the similar technique to solve the alignment problem of VC schemes [18]. Finally, we present implementation results to evaluate the effectiveness of the proposed algorithm. Moreover, we compare our results with other approaches.

The remainder of this paper is organized as follows. Section II presents a review of background and related work. Section III introduces the proposed encryption algorithm. In Section IV, we summarize and conclude our work in Section V.

II. BACKGROUND AND RELATED WORK

A. Background of the General Access Structure

In this section, we briefly review some of Atieniese’s basic definitions and notations on GAS that will be used throughout the paper [10].

Suppose \( \mathcal{P} = \{t_1, \ldots, t_n\} \) is a set of \( n \) participants, and \( 2^\mathcal{P} \) denotes the power set of \( \mathcal{P} \). \( \Gamma_{\text{Qual}} \) denotes the set of subsets of \( \mathcal{P} \) from which we desire to be able to reconstruct the secret; thus, \( \Gamma_{\text{Qual}} \subseteq 2^\mathcal{P} \). Each set in \( \Gamma_{\text{Qual}} \) is said to be a qualified set, while each set not in \( \Gamma_{\text{Qual}} \) is called a forbidden set (denoted as \( \Gamma_{\text{Forb}} \)). Obviously, \( \Gamma_{\text{Qua}} \setminus \Gamma_{\text{Forb}} = 2^\mathcal{P} \) and \( \Gamma_{\text{Qual}} \setminus \Gamma_{\text{Forb}} = \emptyset \). Based on the definitions, a VCS/EVCS for an access structure \( (\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}}) \) on \( \mathcal{P} \) is defined as a technique that can yield \( n \) shares. When we stack together the shares associated with the participants in any set \( X \subseteq \Gamma_{\text{Qual}} \), we can recover the secret image, but any \( X \in \Gamma_{\text{Forb}} \) has no information related to the secret image on the stacked image.

\( \Gamma_0 \) consists of all the minimal qualified sets

\[
\Gamma_0 = \{ A \in \Gamma_{\text{Qual}} : A' \not\in \Gamma_{\text{Qual}}, \forall A' \subset A \}. 
\]

In traditional secret sharing schemes, \( \Gamma_{\text{Qual}} \) is monotonously increasing and \( \Gamma_{\text{Forb}} \) is monotonously decreasing, the access structure \( (\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}}) \) is said to be strong, and \( \Gamma_0 \) is termed a basis. In a strong access structure

\[
\Gamma_{\text{Qual}} = \{ C \subseteq \mathcal{P} : B \subseteq C \text{ for some } B \in \Gamma_0 \}
\]

and we say that \( \Gamma_{\text{Qual}} \) is the closure of \( \Gamma_0 \). For example, if there are four participants sharing a secret image (i.e., \( \mathcal{P} = \{t_1, t_2, t_3, t_4\} \) and \( \Gamma_{\text{Qual}} = \{\{t_1, t_2, t_3\}, \{t_1, t_2, t_4\}, \{t_1, t_3, t_4\}, \{t_2, t_3, t_4\}\} \), stacking any set of subsets \( \{t_1, t_2, t_3\}, \{t_1, t_2, t_4\}, \{t_1, t_3, t_4\} \), or \( \{t_2, t_3, t_4\} \) can reveal the secret image; otherwise, no information can be displayed.

B. Reviews VCS/EVCS for GASs

1) Atieniese’s VCS/EVCS for GASs: In 1996, Atieniese first proposed a VC-based approach of VCS for GASs. They mapped an access structure of VCS to a graph and then found both the lower and the upper bounds on the size of the shares (i.e., the pixel expansion factor) by the graph. They gave minimum pixel expansion factors as well as basis matrices for VCS for strong access structures for at the most four participants [10].

Then, in 2001, they developed a general method that can extend conventional \((k, k)\)-VCSs to \((k, k)\)-EVCSs based upon the basis matrices of VCSs [14]. They proved that their VC-based encoding approach had an optimal pixel expansion factor. Their method consisted of two steps: first, they found out a set of basis matrices to satisfy the VCS for a given access structure. Second, they adopted the theory of hypergraph coloring to obtain the resultant matrices for the EVCS according to the basis matrices. They also discussed some applications of their technique on various interesting classes of access structures. Example 1, which is adopted from Example 5.1 of [14], shows Atieniese’s solution procedures.

Example 1: Let \( \mathcal{P} = \{t_1, t_2, t_3, t_4, t_5\} \) and let \( \Gamma_{\text{Qual}} \) be closure of \( \Gamma_0 \), where \( \Gamma_0 = \{\{t_1, t_2, t_3, t_4\}, \{t_1, t_5\}\} \). Assume that \( \Gamma_{\text{Forb}} = 2^\mathcal{P} \setminus \Gamma_{\text{Qual}} \). A VCS for \( (\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}}) \) can be obtained using basis matrices \( S_b \) and \( S_f \) which are illustrated in the left part (without gray background) of matrices \( S_a^{b} \) and \( S_b^{b} \), respectively. Then, by the theory of hypergraph coloring and Atieniese’s algorithm, a set of basis matrices, \( S_a^{b} S_b^{b} \), and \( S_b^{b} \), are as follows:

\[
S_a^{b} = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

As with other conventional VC-based approaches, Atieniese’s VCS approach for GASs also suffers from the pixel expansion problem. It will expand the size of shares 2–8 times for strong access structures on at the most four participants [10]. Moreover, their extended method for \( (\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}}) \)-EVCSs will introduce extra pixel expansion that will enlarge the size of the shares and decrease the contrast of the recovered images. As shown in Example 1, two extra columns (marked as gray background) have to be added to the basis matrices of VCS for EVCS; the pixel expansion factor is, therefore, increased from 8 to 10. The contrast value is decreased from 1/8 to 1/10. The contrast of meaningful shares is fixed and is only 10% in this access structure.

Besides this, there are other drawbacks of Atieniese’s approach: black secret pixels cannot be completely recovered (e.g., access structures 12, 15, and 17 in [10, Table 1]); the aspect ratio of the recovered image cannot be maintained; the hypergraph coloring problem, which is NP-hard, needs to be solved; this approach needs a sophisticated codebook design.
and the solution approach has to rely on the basis matrices of the existing VCSs cannot be generalized.

2) Liu’s VCS for GASs: In 2010, Liu [13] proposed a step construction to construct VCS for GASs by applying (2,2)-VCS recursively. Their constructions are applicable to binary secret images and consider the VCS both under XOR and OR operations. The main idea behind Liu’s approach is that the participants can take multiple share images for sharing one secret image. Based on their idea, an access structure can be partitioned into several independent access structures to reduce the average pixel expansion (APE). Liu defined the APE as the average value of the total pixel expansions of the share images holding by each participant [13]. Moreover, a VCS for an access structure can be constructed by downsizing the size of the access structure and applying (2,2)-VCS recursively.

Example 2, which is modified from [13, Example 4], shows Liu’s construction. Example 2: Let $P = \{i_1, i_2, i_3, i_4\}$ and let $\Gamma_{\text{Qunl}}$ be closure of $\Gamma_0$, where $\Gamma_0 = \{\{i_1, i_2, i_3\}, \{i_1, i_2, i_4\}, \{i_1, i_3, i_4\}\}$. First, $\Gamma_0$ can be partitioned into $\Gamma_0 = \{\{i_1, i_2, i_3\}, \{i_1, i_2, i_4\}\}$ and $\Gamma_0 = \{\{i_1, i_3, i_4\}\}$. Then, the two schemes $\Gamma_{\alpha}$-VCS and $\Gamma_{\beta}$-VCS will be constructed individually. For the step construction of Construction 3, the construction trees are shown in Fig. 1. For the $\Gamma_{\alpha}$-VCS, apply Construction 2 for $\{\{i_1, i_2\}, \{i_3\}, \{i_4\}\}$, i.e., apply the (2,2)-VCS, and denote the first share distributed to a virtual participant $\{i_1, i_2\}$ as $s_1^R$. The second share is duplicated and then distributed to participants $i_3$ and $i_4$ simultaneously because they are equivalent participants in the access structure. Then, take $s_1^R$ as the secret image, and construct a step construction of $\{i_1, i_2\}$-VCS. Finally, we apply a (2,2)-VCS and distribute the two shares to participants $i_1$ and $i_2$, respectively. The construction procedures of $\Gamma_0 - \text{VCS}_{\text{OR}}$ is similar to that of $\Gamma_0 - \text{VCS}_{\text{OR}}$. Eventually, participants $i_1$, $i_3$, and $i_4$ take two shares, one results in a pixel expansion of 2 times and the other results in pixel expansion of 4 times. Participant $i_2$ only take one share which has 4 times pixel expansion.

For the $\Gamma_{\alpha} - \text{VCS}_{\text{OR}}$, the APE = $(4 + 4 + 2 + 2)/4 = 3$. For the $\Gamma_{\beta} - \text{VCS}_{\text{OR}}$, the APE = $(2 + 0 + 4 + 4)/4 = 5/2$. Hence, the APE of $\Gamma_{\alpha} - \text{VCS}_{\text{OR}}$ is 11/2. The pixel expansion for both schemes $\Gamma_0$-VCS and $\Gamma_{\alpha}$-VCS is 4, and the contrast is 1/4.

Liu claimed that his approach could improve the average pixel expansion and contrast properties as compared with Ateneise’s approach. However, a comparison with the conventional VCS showed that Liu’s approach had some additional drawbacks: first, the participants may take multiple share images with different pixel expansions for one secret image. This differs from conventional VC schemes and will increase administrative inconvenience and difficulty. Second, the decryption process is more complicated than conventional VCSs and needs the help of other devices. For example, for the VCS$_{\text{OR}}$, the participants need to first enlarge the smaller share to the other larger shares. This will lead to an alignment problem while shares were printed on transparencies. Finally, recovered images cannot maintain the same aspect ratio as the original secret image.

II. PROPOSED ENCRYPTION ALGORITHM

In this section, we present a two-phased encryption algorithm of $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$-VCS for GASs. The solution procedures are shown in Fig. 2. In the first phase, it generates intermediate shares (i.e., $I_1, \ldots, I_n$ in Fig. 2) of $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$-VCS. These intermediate shares ($I$-shares) have a meaningless appearance and no pixel expansion. In the second phase, cover images will be added in these $I$-shares to yield the resultant shares of $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$-VCS (i.e., $P_1, \ldots, P_n$ in Fig. 2). Each module in Fig. 2 will be described in the following section.

A. Phase I: Generating I-Shares

1) Problem Statement: This phase aims to construct a pixel-expansion-free VCS for a given access structure $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$. The main idea behind the solution approach is as follows. A security system employs $n'$ different keys to protect a secret and distributes these keys to $n$ participants. Each key may be duplicated and will be distributed to at least one participant. Each participant is allowed to hold at least one key. Considering this scenario, if someone wants to access the secret, he/she has to collect $n'$ different keys from a set of participants. If he/she misses any one type of key, the secret will remain protected. To construct such a security system for an access structure $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$, the dealer has to carefully distribute $n'$ different keys to any set of participants $X, X \in \Gamma_{\text{Qunl}}$, and has to guarantee that any set of participants $Y, Y \in \Gamma_{\text{Forb}}$, will not hold all types of keys. Hence, the method of distribution of $n'$ different keys to $n$ participants will provide a solution for a given access structure. Before applying the above-mentioned scenario to $(\Gamma_{\text{Qunl}}, \Gamma_{\text{Forb}})$-VCS, there are three key points to be overcome: first, what constructions can be used to encrypt secret images without pixel expansion? Next, how many keys...
(shares) can satisfy the requirement of the given access structure? Finally, how should the keys be distributed?

For example, there are 4 participants, \( P = \{i_1, i_2, i_3, i_4\} \), sharing a secret image upon the qualified set \( \Gamma_{\text{Qual}} = \\{\{i_1, i_2, i_3, i_4\}\} \) and the forbidden set \( \Gamma_{\text{Forb}} = 2^P \setminus \Gamma_{\text{Qual}} \). (We keep this relation in the rest of the paper.) Intuitively, we can utilize the construction of a \((4,4)\)-VCS to yield 4 shares, \( s_1, s_2, s_3, \) and \( s_4 \) (called basis shares), and then distribute one basis share to each participant. For example, participant \( i_1 \) holds the share \( s_1, i_2 \) holds the share \( s_2, \) and so on. Finally, we have 4 I-shares \( I_1, I_2, I_3, \) and \( I_4 \) (according to \( \Gamma_{\text{Qual}} = \\{\{i_1, i_2, i_3, i_4\}\} \)), the secret image which is the same as the result of the \((4,4)\)-VCS, can be revealed.

In this paper, construction of \((n', n')\)-VCSs is adopted to be the encryptor in Fig. 2. The I-shares of a \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCS are synthesized from the basis shares that are the products of the encryptor. The construction set \( C \) holds the relation between the basis shares and the I-shares.

**Definition 1:** Assume that the \((n', n')\)-VCS, where \( n' \geq 2 \) is to be used to construct a \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCS with \( n \) participants, \( P = \{i_1, \ldots, i_n\}, n \geq 2 \). Construction set \( C \) denotes the constitution member of \( n \) shares of the \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCSs. \( c_\alpha \) represents the constitution member of participant \( \alpha \), \( c_\alpha = \{s_j, 1 \leq j < n'\} \), \( c_\alpha \subset C \). Basis share \( s_j, 1 \leq j < n' \), is the resultant share of the \((n', n')\)-VCS. Shares of participant \( i_\alpha \) can be defined as \( I_\alpha = \cup_{s_j \in c_\alpha} s_j \).

By Definition 1, the construction set of the above example can be expressed as \( C = \{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}\} \). As black and white pixels are presented by logical “1” and “0,” respectively, the stacking shares will be limited in pixel-wise “OR”-ed operation. Hence, the recovered image for \( \Gamma_{\text{Qual}} = \{\{i_1, i_2, i_3, i_4\}\} \) should be the stacking result of \( s_1, s_2, s_3, \) and \( s_4 \) which is the same as that of the \((4,4)\)-VCS.

**Example 3:** Suppose there are 4 participants, \( P = \{i_1, i_2, i_3, i_4\} \), who share a secret image and the qualified set is \( \Gamma_{\text{Qual}} = \{\{i_1, i_2, i_3\}, \{i_1, i_2, i_4\}, \{i_1, i_3, i_4\}, \{i_2, i_3, i_4\}\} \). Then the \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCSs can be constructed by utilizing the construction of the \((3,3)\)-VCS to yield 3 basis shares \( s_1, s_2, \) and \( s_3 \). The construction set can be set as \( C = \{\{s_1\}, \{s_2\}, \{s_3\}\} \). By Definition 1, we can obtain 4 shares for the \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCS, that is \( I_1 = \{s_1\}, I_2 = \{s_2\}, I_3 = \{s_3\}, \) and \( I_4 = \{s_4\} \). Stacking any one subset in \( \Gamma_{\text{Qual}} \) can get all the basis shares, otherwise it will leak at least one.

Assume that the qualified set of Example 3 is altered to \( \Gamma_{\text{Qual}} = \{\{i_1, i_2\}, \{i_1, i_2, i_3\}, \{i_1, i_2, i_4\}, \{i_1, i_3, i_4\}, \{i_2, i_3, i_4\}\} \). We can also employ the construction of the \((3,3)\)-VCS to construct the \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCS. Now, the construction set can be modified as \( C = \{\{s_1\}, \{s_2, s_3\}, \{s_2\}, \{s_3\}\} \). The I-shares for the \((\Gamma_{\text{Qual}}, \Gamma_{\text{Forb}})\)-VCS become \( I_1 = \{s_1\}, I_2 = \{s_2, s_3\}, I_3 = \{s_2\}, \) and \( I_4 = \{s_3\} \).
\(x_{\alpha j}\)  \(x_{\alpha j} = 1\) presents
\(s_j \in c_\alpha\), otherwise \(s_j \notin c_\alpha, x_{\alpha j} \in \{0, 1\}\),
\(1 \leq \alpha \leq n, 1 \leq j \leq n'.\)

\(C\)  Construction set, \(C = \{c_\alpha, 1 < \alpha < n\}\),
where \(c_\alpha = \{s_j / x_{\alpha j} = 1, 1 \leq j \leq n'\}\).

Object:  
\[
\text{minimize } \quad n' \\
\text{subject to: } \\
\sum_{1 \leq j \leq n'} x_{\alpha j} \geq 1, \quad \forall 1 \leq j \leq n'(1) \\
\sum_{\forall \alpha \leq n} x_{\alpha j} \geq 1, \quad \forall X \in \Gamma_{\text{quad}}(2) \\
\sum_{\forall \alpha \leq n} c_\alpha = S, \quad \forall X \in \Gamma_{\text{quad}}(3) \\
\sum_{\forall \alpha \leq n} c_\alpha \neq S, \quad \forall X \in \Gamma_{\text{forb}}(4) \\
x_{\alpha j} \in \{0, 1\}, \quad \forall 1 \leq \alpha \leq n, 1 \leq j \leq n'(5) \\
n' \geq 2, \quad n' \in \mathbb{Z}. \quad (6)
\]

Constraint (1) ensures that each share of the \((\Gamma_{\text{quad}}, \Gamma_{\text{forb}})\)-VCS consists of at least one basis share. Constraint (2) limits each basis share to be assigned to at least one participant. Constraints (3) and (4) examine whether the construction set \(C\) obeys the access structure \((\Gamma_{\text{quad}}, \Gamma_{\text{forb}})\). Constraints (5) and (6) limit the ranges of the decision variables.

Suppose there are \(n\) participants sharing a secret. If \(n'\) is known, there are \(2^n - 1\) possible combinations to construct the share of a participant. The complexity to solve the optimization problem is \(O(2^n)\). When \(n\) and \(n'\) increase, the solution space of the optimization problem will grow very fast.

3) Algorithm for GAS Solver:  In this section, we develop an algorithm based on the simulated annealing (SA) [19] approach to solve the proposed mathematic optimization formulation for the GAS problem.

We transform and relax the original mathematical model for the GAS solver to simplify solution procedures. Our solution approach adopts an iterative improvement framework. First, we treat the decision variable \(n'\) as a given variable and the original optimization problem is transformed into a decision problem. In each iteration, we try to find a solution with a given \(n'\) and refine \(n'\) according to the possibility of a solution being found or not.

We continue to relax Constraints (3) and (4) of the proposed model by penalizing the energy function. The penalized energy function \(E_p\) can be defined as follows:

\[E_p = N_f + \Gamma_{\text{quad}}/(1 + N_q)\]

where \(N_f = \sum_{\forall \alpha \leq n, \forall \alpha \leq n, \forall X \in \Gamma_{\text{quad}}} c_\alpha - 1\) and \(N_q = \sum_{\forall \alpha \leq n, \forall \alpha \leq n, \forall X \in \Gamma_{\text{forb}}} c_\alpha - 1\). Here, \(N_q\) denotes the amount of a set (which belongs to the qualified set) that can recover the secret image. If a solution satisfies Constraint (3), \(N_q\) will be \(\Gamma_{\text{quad}}\); otherwise, \(0 \leq N_q < \Gamma_{\text{quad}}\). Notation \(N_f\) represents the amount of an insecure forbidden set in \(\Gamma_{\text{forb}}\). If a solution is secured, \(N_f\) should be equal to 0. This implies that the solution satisfies Constraint (4). The penalized energy function \(E_p\) can be used not only to measure whether a solution violates Constraints (3) and (4) but also to represent how close the solution is to its feasible status. In the proposed SA-based algorithm, we will use the penalized energy function to replace the original energy function; Constraints (3) and (4) can, therefore, be omitted during the solution process. It should be noted that \(E_p\) for a feasible solution must be less than 1.

The proposed iterative improvement framework is as listed in Algorithm 1. Initially, the algorithm starts to find a solution for the given GAS by the procedure \(\text{GASsolver}(\cdot)\) with an initial guess \(n' = n\) in Steps 1 and 2. In each iteration (i.e., from Steps 2 to 10), the algorithm proceeds to find a minimum \(n'\) by decreasing or increasing the value of \(n'\) by 1. If a solution is found (i.e., \(Z_{\text{best}} < 1\), \(Z_{\text{best}}\) denotes the best-found energy function in the last iteration), the algorithm stops while \(n' > n\) or in the case of \(n' < n\), it decreases the value of \(n'\) by 1 and proceeds to the next iteration with the lower \(n'\). On the contrary, if a solution is not found, the algorithm stops while \(n' < n\) or it increases the value of \(n'\) by 1 and proceeds to the next iteration while \(n' > n\). At the end of the procedure, the algorithm outputs a minimum \(n'\) and a construction set \(C\) as the optimal solution of the problem. If no solution can be found for a given access structure, the solution procedure will be terminated while \(n' = n_{\text{max}}\), where \(n_{\text{max}}\) is a given parameter that prevents Algorithm 1 from falling into an infinite loop.

Algorithm 1: SA-based algorithm for GAS solver

Procedure \(\text{GASsolver}(n, n_{\text{max}}, \Gamma_{\text{quad}}, \Gamma_{\text{forb}})\)
1. \(n' \leftarrow n\).
2. Call \(\text{GASSA}(n, n', \Gamma_{\text{quad}}, \Gamma_{\text{forb}}, C_{\text{best}}, Z_{\text{best}})\).
3. If \(Z_{\text{best}} \geq 1\) then //No solution in last turn
4. \(n' \leftarrow n' + 1\).
5. If \(n' = n_{\text{max}}\) then Stop and Output “No solution found”
6. If \(n' < n\) then goto Step 11.

Else
7. \(C \leftarrow C_{\text{best}}\). //Found a solution in last turn
8. If \(n' > n\) then goto Step 11.
9. \(n' \leftarrow n' - 1\). //Improvement
10. End If
12. Output \(n', C\).

Algorithm 2 presents the pseudocode of the proposed SA-based algorithm, \(\text{GASSA}(\cdot)\). In Step 1, the initial guess can be generated as follows:

\[\forall 1 < i < n, \forall 1 < j < n', x_{i,j} \leftarrow 1.\]

The initial guess can satisfy Constraints (1) and (2) at the same time. Steps 2 and 3, evaluate the value of \(F_p\) for the initial guess and store the results of the initial guess as the current best-found solution (i.e., \(Z_{\text{best}}\) and \(C_{\text{best}}\)). Step 4 sets the initial value for annealing parameters \(t\) and \(r\). From Steps 5 to 20, the main loop
of the SA heuristic is executed repeatedly. At each state, the solution procedure selects a share \( i \), \( 1 \leq i \leq n \), and alters the current state of \( e_i \) to a neighbor of the current state randomly, except for an empty set. Steps 9 to 17 evaluate the value of the energy function \( E \) for the new state and decide whether to accept the new state or not. The objective value with the minimum energy value should be saved as the best-found solution in Step 15. After \( r \) solution iterations, the annealing parameters are modified in Step 19. The solution procedure will be terminated while \( t < t_f \) or a solution without penalty (i.e., \( Z_{\text{old}} < 1 \)) is found. When the algorithm has stopped, the best-found solution \( C_{\text{best}} \) corresponding to \( Z_{\text{best}} \) is a solution to this problem, if \( Z_{\text{best}} \) receives no penalty.

Algorithm 2: Pseudocode of proposed SA-based algorithm

Procedure \( \text{GAS-SA}(n, n', \Gamma_{\text{qaur}}, \Gamma_{\text{Forb}}, C_{\text{best}}, Z_{\text{best}}) \)

1. \( \forall \leq i \leq n, \forall \leq j \leq n', \forall \leq i \leq 1 \).
2. Calculate \( E_p \) for the above initial guess.
3. \( Z_{\text{old}} \leftarrow E_p, Z_{\text{best}} \leftarrow Z_{\text{old}}, C_{\text{best}} \leftarrow C \).
4. \( t \leftarrow t_0, r \leftarrow 0 \).
5. While \( Z_{\text{old}} \geq 1 \) and \( t \geq t_f \) do
6. Repeat \( r \) times
7. Randomly select a share \( e_i, e_i \in C \).
8. Alter the solution configuration of \( C_i \) randomly.
9. Calculate \( E_p \) for the new configuration.
10. \( Z_{\text{new}} \leftarrow E_p \).
11. \( \Delta E \leftarrow Z_{\text{new}} - Z_{\text{old}} \).
12. Generate a random number \( \rho \) uniformly distributed in \( (0, 1) \).
13. If \( \Delta E < 0 \) or \( \rho < e^{(-\Delta E/\gamma)} \) then
14. \( Z_{\text{old}} \leftarrow Z_{\text{new}} \).
15. If \( Z_{\text{old}} < Z_{\text{best}} \) then \( Z_{\text{best}} \leftarrow Z_{\text{old}} \),
16. \( C_{\text{best}} \leftarrow C \).
17. If \( Z_{\text{old}} < 1 \) then goto Step 21
18. else recover the action in Step 8.
19. End Repeat
20. \( t \leftarrow \alpha_T \times t, r \leftarrow \beta_T \times r \).
21. End While
22. Output \( C_{\text{best}}, Z_{\text{best}} \).

4) Share Constructors: The share constructor consists of two modules in Fig. 2: the encryptor and the share synthesizer. The encryptor adopts Yang’s construction for the \((n', n')\)-ProbVSS scheme directly (i.e., [6, Construction 3]). Yang’s construction rule is summarized as follows:

Let \( C_0 \) and \( C_1 \) be the two white and black sets consisting of \( n' \times 1 \) matrices for a \((n', n')\)-VCS. Then, \( C_0 \) and \( C_1 \) are constructed as \( C_0 = \{ \mu_{0,i}, i \in \text{Even}, 0 \leq i \leq n' \} \) and \( C_1 = \{ \mu_{1,i}, i \in \text{Odd}, 0 \leq i \leq n' \} \). Let \( C_0 \) (respectively, \( C_1 \)) be the code distributing patterns for white (respectively, black) secret pixels. Chosen probability for each \( \mu_{0,i} \) and \( \mu_{1,i} \) is \( \frac{1}{2n' - 1} \).

The procedure of the share constructor is shown as Algorithm 3. The first step (i.e., the encrypt in Fig. 2) generates the basis shares by employing Yang’s \((n', n')\)-ProbVSS scheme.

The second step (i.e., the share synthesizer in Fig. 2) produces I-shares by stacking the basis shares upon construction set \( C \).

Algorithm 3: Proposed algorithm of share constructor

Procedure \( \text{share_constructor}\{n, n', C, S\} \) // SI: Secret Image
1. Encrypting SI by Yang’s \((n', n')\)-ProbVSS to yield a set of basis shares \( \{s_1, \ldots, s_{n'}\} \).
2. \( \forall 1 \leq i \leq n, c_a \in C, I_a \leftarrow \text{∪}_{s_j \in c_a} s_j \).
3. Output I-shares \( I_1, \ldots, I_n \).

Example 4: Suppose there are 4 participants, \( \mathcal{P} = \{i_1, i_2, i_3, i_4\} \), sharing a secret image and the qualified set \( \Gamma_0 = \{i_2, i_3\} \). Solving the problem by utilizing the \( \text{GAS solver} \) procedure, it can obtain \( n' = 3 \) and construction set \( C = \{s_1, s_2, s_3\} \). By Yang’s construction rule, the codebook for \((3, 3)\)-VCS are

\[
C_0 = \begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
C_1 = \begin{bmatrix}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 \\
\end{bmatrix}
\]

The chosen probability for each 3 \( \times 1 \) column vector is \( 1/4 \). Then, \( C_0 \) and \( C_1 \) are employed to produce 3 basis shares, \( s_1, s_2, and s_3 \). Finally, distributing basis shares \( s_1 and s_2 \) to participants \( i_1 and i_4 \) respectively, a composite share is generated by stacking \( s_2 and s_3 \) together for participant \( i_2 \) respectively, \( i_3 \).

The shared and recovered images that are constructed in this phase have the following properties.

Property 3: The recovered image has no pixel expansion.

Proof: The recovered image is produced by stacking all shares constructed by Yang’s \((n', n')\)-ProbVSS scheme directly. Yang has shown that this construction is free of pixel expansion. That proves this property.

Property 4: The black secret pixels can be perfectly reconstructed in the recovered images.

Proof: By observing the code distributing patterns, \( C_1 = \{\mu_{1,i}, i \in \text{Odd}, 0 \leq i \leq n'\} \), it can be found that each column matrix contains at least one “1” (black pixel). Hence, the stacked image can reconstruct all the black secret pixels completely.

Property 5: The I-shares are secure.

Proof: Because each I-share is composed of the basis shares, this property can be evaluated by the following two facts: first, the basis shares are secure. Second, each I-share is only composed of a part of the basis shares. Hence, any single I-share cannot reveal anything related to the secret image. That proves the security property.

B. Phase II: Stamping Cover Images

In this section, we have proposed a stamping algorithm to stamp cover images on I-shares \( I_1, \ldots, I_n \), that were produced in the first phase. There are two major differences between our approach and the existing research for the EVCS. First, we put
cover images on shares directly rather than redesign a codebook for a specific VC scheme; hence, our approach does not introduce any pixel expansion in this phase. Second, the black pixel density of the cover images can be adjusted on demand in a fine-grained fashion.

Suppose \( C^x_{\alpha,y} \) denotes a collection of pixel colors for \( \alpha \)-share \( \alpha \), cover image \( \alpha \), and secret image in coordinate \( (x,y) \). Notation \( C^x_{\alpha,y} = \{0, 1, 1\} \) expresses there are a white, black, and black pixels on \( \alpha \)-share \( \alpha \), cover image \( \alpha \), and secret image in \( (x,y) \). The stamping algorithm sticks extra black pixels on \( \alpha \)-shares \( \alpha \) where collection of pixel colors is \( C^x_{\alpha,y} = \{0, 1, 1\} \) or \( C^x_{\alpha,y} = \{0, 1, 0\} \). That means the stuck black pixels only appear in the region where shared pixels are white and cover pixels are black. Hence, having more black pixels appear in the region, the cover images can be revealed. Besides, there are two major issues have to be considered in the algorithm. First, it preserves the security property for \( \alpha \)-shares. Second, it keeps the contrast of reconstructed images as high as possible.

To preserve the security condition, on the resultant shares (called stamped shares), the algorithm has to maintain an equal density for cover-pixels between two regions where collection of pixel colors are \( C^x_{\alpha,y} = \{0, 1, 1\} \) and \( C^x_{\alpha,y} = \{0, 1, 0\} \), respectively. Otherwise, a portion of the secret image will be disclosed on stamped shares. To meet this condition, we first calculate the amount of extra cover-pixels for each share. Suppose \( \Delta d_{\alpha} \) presents the desired density for cover pixels on \( \alpha \)-share \( \alpha \). The amount of extra cover-pixels \( \delta^\alpha_{k} \) and \( \delta^\alpha_{m} \) will be scattered in the regions where \( C^x_{\alpha,y} = \{0, 1, 1\} \) and \( C^x_{\alpha,y} = \{0, 1, 0\} \), respectively. The amount of cover-pixels are

\[
\delta^\alpha_k = \Delta d_{\alpha} A^\alpha_k \quad \text{and} \quad \delta^\alpha_m = \Delta d_{\alpha} A^\alpha_m
\]

respectively, where \( A^\alpha_k \) and \( A^\alpha_m \) denote the amount of pixels in the regions where \( C^x_{\alpha,y} = \{0, 1, 1\} \) and \( C^x_{\alpha,y} = \{0, 1, 0\} \), respectively. In the stamping phase, we randomly add \( \delta^\alpha_k \) and \( \delta^\alpha_m \) cover pixels on their corresponding regions of \( \alpha \)-share \( \alpha \), the cover image can be displayed, and the security condition of resultant shares can be satisfied.

Because the cover and secret images are binary images, adding extra black pixels on \( \alpha \)-shares may increase the black pixels in the white region of the recovered image. Thus it may decrease the contrast of the image. Hence, the stamping algorithm tries to add black cover-pixels on the same coordinate of each \( \alpha \)-share to reduce the overall number of extra cover-pixels in the recovered image. Notation \( f_{x,y} \) denotes overlapped frequency of black cover-pixels at coordinate \( (x,y) \). Notation \( f_{x,y} = k \) means there are \( k \) cover images which have a black pixel at \( (x,y) \). Before scatter cover pixels on \( \alpha \)-shares, we first calculate \( f_{x,y} \) for each coordinate. According to the value of \( f_{x,y} \), randomly select candidate coordinates from the region where \( C^x_{\alpha,y} = \{0, 1, 1\} \) to add extra cover pixels to \( \alpha \)-shares. In this manner, coordinates with larger \( f_{x,y} \) have higher priority to be added cover pixels. This method ensures that cover pixels have a high probability of being allocated to the same coordinate such that the contrast value of the recovered image is high. A detailed description of the stamping algorithm is provided in Algorithm 4. Table I lists the notations included in Algorithm 4.

**Algorithm 4: Proposed stamping algorithm**

Procedure Stamping \( \{I_1, \ldots, I_n, \Delta d_1, \ldots, \Delta d_n\} \)

1. \( \forall 1 \leq \alpha \leq n, \text{calculate } C^x_{\alpha,y} \text{ for each coordinate } (x,y). \)
2. \( \forall 1 \leq \alpha \leq n, \text{calculate } \delta^\alpha_k = \Delta d_{\alpha} A^\alpha_k \text{ and } \delta^\alpha_m = \Delta d_{\alpha} A^\alpha_m. \)
3. Calculate \( f_{x,y} \) for each coordinate \( (x,y) \).
4. \( \forall (x,y), e_{x,y} = 0 \)
5. For \( r \leftarrow 0 \) downto 1
6. Begin
7. \( \delta^\alpha_k \text{ and } \delta^\alpha_m \text{ } \) 6.6 If \( \text{random}() > \rho \) then goto Step 6.1
8. \( \delta^\alpha_m \text{ } \) 6.5 For \( \alpha \leftarrow 1 \) to \( n \)
9. \( \delta^\alpha_m \text{ } \) 6.6 Begin
10. \( \delta^\alpha_k \text{ } \) 6.6.1 If \( C^x_{\alpha,y} = \{0, 1, 1\} \text{ and } \delta^\alpha_k > 0 \) then \( \text{Add a black pixel in } (x,y) \) of \( I_\alpha \).
11. \( \delta^\alpha_m \text{ } \) 6.6.2 If \( C^x_{\alpha,y} = \{0, 1, 0\} \text{ and } \delta^\alpha_m > 0 \) then \( \text{Add a black pixel in } (x,y) \) of \( I_\alpha \).
12. \( \delta^\alpha_m \text{ } \) 6.7 EndFor
13. \( \delta^\alpha_m \text{ } \) 6.8 If \( \sum_{1 \leq \alpha \leq n} \delta^\alpha_k = 0 \text{ and } \sum_{1 \leq \alpha \leq n} \delta^\alpha_m = 0 \) then goto Step 9.
14. \( \delta^\alpha_m \text{ } \) 6.9 EndFor
15. \( \delta^\alpha_m \text{ } \) 6.10 If \( \sum_{1 \leq \alpha \leq n} \delta^\alpha_k \neq 0 \text{ or } \sum_{1 \leq \alpha \leq n} \delta^\alpha_m \neq 0 \) then goto Step 4.
16. \( \delta^\alpha_m \text{ } \) 6.11 \( \forall 1 \leq \alpha \leq n, \text{Output } I_\alpha \) as stamped shares \( P_\alpha \).

Steps 1–3 calculate all required parameters. Step 4 initializes indicators \( e_{x,y} \) for all coordinates. Steps 5–7 stamp cover pixels on \( I_\alpha \) in iterations. In Step 6.2, each coordinate will be randomly selected once at most based on its priority \( f_{x,y} \). In Step 6.4, the use of constant \( \rho \) and function \( \text{random}() \) can avoid the clustering of cover-pixels in coordinates with higher priority. Constant \( \rho \) ranges between 0.7 and 1.0. Steps 6.5–6.7 add cover pixels on selected coordinates \( (x,y) \) of \( I_1, \ldots, I_n \) on demand. Please note that black cover-pixels will only be added on candidate coordinate \( (x,y) \) of \( I_\alpha \) that has a white pixel on it (i.e.,
$C^α_{x,y} = \{0, 1, \text{ or } \ast\}$. Step 6.6.1 (Step 6.6.2) adds an extra cover pixel to $I_a$ if the desired density for cover image in the region of black (white) secret pixels does not reach (i.e., parameter $δ^α_w(δ^α_w')$ is greater than zero). In Step 8, if the required densities of cover-pixels is not reached in corresponding share, the algorithm performs Steps 4–7 again until all required cover-pixels are stamped on shares $I_1, \ldots, I_k$. Step 9 outputs all the resultant stamped shares $I_1, \ldots, I_k$.

The main loop (i.e., Steps 4–7) comprises the majority of the algorithm’s time complexity. The execution time of the loop depends on the selection of a random number generator and constant $μ$. In the best case, the main loop will be only executed one time, the time complexity of the algorithm is $O(nhw)$, where $h$ and $w$ are the height and width of a secret image, respectively. Suppose the loop will be executed $φ$ times in the worst case; the time complexity of the stamping algorithm is $O(φn^2hw)$. Besides, in order to uniformly distribute cover pixels over shares, all images are divided into smaller blocks (e.g., $8 \times 8$ pixels) at the beginning of the algorithm.

**Example 5:** Suppose there are 2 participants, $P = \{τ_1, τ_2\}$, sharing a secret image and the qualified set $Γ_0 = \{I_1, I_2\}$. The example images are shown in Fig. 3. The incremental pixel densities of the cover image for the 2 participants are $Δd_1 = 25\%$ and $Δd_2 = 30\%$.

From Fig. 3, we have $A_{w_1}^1 = 30$, $A_{w_2}^2 = 25$. By using Step 2, we can calculate $δ^1_w = 6 \times 0.25 = 1.5 \leq 2$, $δ^2_w = 36 \times 0.25 = 7.5 \leq 8$, $δ^2_w' = 15 \times 0.3 = 4.5$, $δ^1_w' = 25 \times 0.3 = 7.5 \geq 8$. Parameter $f_{x,y}$ for each coordinate $(x, y)$ can be expressed as Fig. 3(a).

In the first iteration, coordinates where $f_{x,y} = 2$ have the highest priority to be candidates for adding cover pixels. Suppose coordinate $(1,5)$ is chosen and the collections of pixel colors are $C^1_{x,y} = C^2_{x,y} = \{0, 1, 0\}$. Parameters were updated as $δ^1_w = 0$, $δ^2_w = 2$, $δ^2_w' = 5$, and $δ^2_w' = 2$. In the second iteration, the stopping condition is not satisfied, the algorithm continues to perform the second round stamping process. In the first iteration of the second round, coordinates (3,4), (8,8), (6,6), and (5,8) were chosen and cover pixels were added on $I_2$. Parameters were updated as $δ^2_w = 1$ and $δ^2_w' = 0$. In the second iteration, the cover pixel was added to coordinate (6,4) in $I_2$. Finally, the resultant shares $P_1$ and $P_2$ were produced and then the algorithm stops because of the stopping condition is reached.

An example of resultant shares $P_1$ and $P_2$ are shown in Fig. 3(g) and (h). Obviously, the contrast of cover image on shares $P_1$ and $P_2$ are high enough to reveal the cover images.

## IV. Experimental Results

In this section, we first evaluate the performance of the proposed optimization model by comparing with the previous VC results for GASs. Then, we assess the performance of the proposed encryption algorithm for EVCS in terms of the contrast of the recovered secret images. Finally, we demonstrate the results of our implementation of EVCS by examples.

### A. Performance of GAS Solver

The proposed algorithm for GAS solver is coded in $C$ language. The parameters of the cooling schedule for Algorithm 1 are $α_T = 0.75$ and $β_T = 1.1$. The initial values include $r_0 = 100$ and $t_0 = 1$. The frozen temperature $t_f = t_0/10$.

In the following subsections, we examine the performances of the GAS solver in two different VC scenarios and compare them with Ateniese’s and Liu’s results. In both comparisons, we test all strong access structures with up to four participants as listed on [10, Table 1].

1) **Comparison With Conventional VCS for GASs:** First, we test the conventional VCS scenario restricting each participant to hold a single share image for sharing one secret image. The solution results of the proposed GAS solver are listed in Table II. In Table II, all strong access structures have a corresponding construction set $C$, which means that a pixel-expansion-free VCS for one of these access structures can be constructed via a $(n', n')$-VCS.

Fig. 4 illustrates the implementation results for minimal qualified set 5. The secret image can be revealed by stacking qualified participants, as shown in Fig. 4(i). However, it cannot be accessed by any forbidden participants, as shown in Fig. 4(g) and (h). It proves the effectiveness of the proposed GAS solver algorithm.

Although Ateniese has proved that their VC-based approach has optimal pixel expansion, his approach still results in pixel expansion of 2–8 times, as listed on Table II. Our proposed algorithm can totally remove the pixel expansion; hence, our approach is superior to Ateniese’s method in terms of the pixel expansion.

2) **Comparison With Liu’s Scenario:** In the second comparison, we adopt Liu’s scenario that allows each participant to carry multiple share images for sharing one secret image. For fairness, before the start of the encryption process, the minimal qualified sets listed on Table II are also partitioned into several (6,5) and (8,5) in $I_2$. Parameters were updated as $δ^1_w = 0$, $δ^1_w = 0$, $δ^2_w = 3$, and $δ^2_w = 2$. In Step 8, because the stopping condition is not satisfied, the algorithm continues to perform the second round stamping process. In the first iteration of the second round, coordinates (3,4), (8,8), (6,6), and (5,8) were chosen and cover pixels were added on $I_2$. Parameters were updated as $δ^2_w = 1$ and $δ^2_w = 0$. In the second iteration, the cover pixel was added to coordinate (6,4) in $I_2$. Finally, the resultant shares $P_1$ and $P_2$ were produced and then the algorithm stops because of the stopping condition is reached.

An example of resultant shares $P_1$ and $P_2$ are shown in Fig. 3(g) and (h). Obviously, the contrast of cover image on shares $P_1$ and $P_2$ are high enough to reveal the cover images.
TABLE II
Solution Results and Comparisons for Conventional VCS for GASs

<table>
<thead>
<tr>
<th>No.</th>
<th>Minimal qualified sets</th>
<th>Pixel expansion factor</th>
<th>Our solution results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>{(s_1), (s_2)}</td>
</tr>
<tr>
<td>2</td>
<td>12,13</td>
<td>2</td>
<td>{(s_1), (s_2), (s_3)}</td>
</tr>
<tr>
<td>3</td>
<td>12,13,23</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4)}</td>
</tr>
<tr>
<td>4</td>
<td>12,13,23,14</td>
<td>4</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5)}</td>
</tr>
<tr>
<td>5</td>
<td>12,13,23,34</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6)}</td>
</tr>
<tr>
<td>6</td>
<td>12,13,14</td>
<td>2</td>
<td>{(s_1), (s_2), (s_3), (s_4)}</td>
</tr>
<tr>
<td>7</td>
<td>12,14,23,23,34</td>
<td>4</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6)}</td>
</tr>
<tr>
<td>8</td>
<td>12,13,23,24,34</td>
<td>5</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>9</td>
<td>12,13,14,23,24</td>
<td>4</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6)}</td>
</tr>
<tr>
<td>10</td>
<td>12,13,14,23,24,34</td>
<td>5</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
</tbody>
</table>

Note: The minimal qualified sets with prepartitioning are the same as those listed on Table II.

TABLE III
Our Outcomes by Combining Liu’s Partition Results

<table>
<thead>
<tr>
<th>No.</th>
<th>n’</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>{(s_1), (s_2)}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>{(s_1), (s_2), (s_3)}</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4)}</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5)}</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6)}</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6)}</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>{(s_1), (s_2), (s_3), (s_4), (s_5), (s_6), (s_7)}</td>
</tr>
</tbody>
</table>

Note: The qualified sets with prepartitioning are the same as those listed on Table II.

TABLE IV
Comparison by Pixel Expansion of Recovered Image

<table>
<thead>
<tr>
<th>No.</th>
<th>Pixel expansion factor</th>
<th>Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ateniese</td>
<td>Liu</td>
<td>Our</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>5</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>6</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>7</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>8</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>9</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>10</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>11</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>12</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>13</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>14</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>15</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>16</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>17</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>18</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

Fig. 4. A parts of Implementation results for minimal qualified set 5. (a) Secret image; (b) share 1; (c) share 2; (d) share 3; (e) share 4; (f) shares 1 and 2; (g) shares 1 and 3; (h) shares 1 and 4.

subsets, which are the same as Liu’s partitioning results. The different parts of the qualified sets are separated by a semicolon, as shown in Table II.

Table III lists our solution outcomes by combining Liu’s prepartitioning results. Due to the prepartitioning, a qualified set is divided into several smaller subsets; hence, each subset can be solved by employing the previous solution results. For example, in Table III, the minimal quality set 5 can be divided into \{i_2, i_3\} and \{"i_1, i_2\}; \{"i_1, i_3\}\}. Hence, the construction result for minimal quality sets 1 and 2 can be adopted for \{i_2, i_3\} and \{"i_1, i_2\}; \{"i_1, i_3\}\}, respectively. The results listed in Table III indicate that our solution approaches can also be applicable to Liu’s encryption scenario.

Table IV shows that Ateniese’s and Liu’s approaches have the same pixel expansion for the recovered image. By combining Liu’s prepartitioning results, Ateniese’s approach can have a smaller pixel expansion factor than his own previous results in some access structures. Our approach is still pixel-expansion-free which means that by applying our method, the dealers do not need to resize their shares before decryption. Table IV also indicates that all the approaches have the same contrast values for the recovered images. The above comparison results demonstrate that the proposed encryption approach has a great performance in terms of pixel expansion for recovered images and APE for shares.

B. Contrast of Images Recovered by EVCS

In this subsection, we evaluate the performance of the EVCS algorithm by the contrast of recovered images. Cover images and the secret image for this experiment are shown in Figs. 6 and 7. All figures are in 512 x 384 pixels and have been reduced to 25% of the original size due to space limitation. The incremental pixel density \(\Delta d_\alpha\) for the cover image \(\alpha\) is set at the same value in this experiment.

Table V lists a part of the contrast value of the recovered images. The resultant contrast values are compared with the theoretical upper bound of the recovered image. Obviously, these contrast values obtained by the proposed EVCS algorithm are very close to its upper bound. Actually, the use of the stamping
algorithm reduces only slightly the contrast of the recovered images. This degradation is proportional to the parameter $\Delta d_c$, regardless of the VCS’s access structure. This verifies the effectiveness of the proposed stamping algorithm.

### C. Demonstration of Our Results for EVCS

In this subsection, we demonstrate the implementation results of the proposed construction algorithm for the conventional EVCS. To evaluate whether the stamping algorithm can be applied to various types of cover images, cover images 1 is replaced with the inversed version of Fig. 6(a). Cover images 2 and 3 are as presented in Figs. 8(a) and 6(c), respectively.

The implementation results of access structure 4 in Table II are shown in Fig. 8. The parameters $\Delta d_c$ for cover images 1, 2, and 3 are set to 15%, 20%, and 20%, respectively. Fig. 8(b), (c), and (d) show that the cover images can be stamped and can be clearly revealed on meaningless shares. Fig. 8(e), (f), and (g) cannot display any information related to the secret image due to the forbidden combinations of decryption. On contrary, the qualified set $\{i_1, i_2, i_3\}$ can recover the secret image, as shown in Fig. 8(h). This example demonstrates the feasibility of our hybrid approach. This indicates that an excellent display quality of the recovered images is possible by employing the proposed stamping algorithm.

The proposed stamping algorithm may result in some dim traces of cover images on the recovered images. There are three ways to remove or conceal the traces to promote the display quality for recovered images. First, use graphs drawn by lines...
be the cover images, for example, the cover images in Fig. 6. Second, adjust the density of cover images as low as possible. Finally, adopt a pair of complementary images to be the cover images in each qualified set. This method was also used in [20].

V. CONCLUSION

In this paper, we have proposed a two-phased encryption algorithm for the EVCS for general access structures. From the point of view of pixel expansion, our approach successfully solves the open questions. The experimental results also show that in most of the cases, our approach has better performances than those proposed in previous research in terms of the display quality of the recovered image, which includes contrast, perfect reconstruction of black secret pixels, and maintenance of the same aspect ratio as that of the original secret image.

The proposed algorithm has four advantages. First, the algorithm is a generic approach. It can construct the EVCS for general access structures without the need to design a sophisticated codebook. The second advantage is the modularity. Each phase in the encryption procedure is less coherent, so it can be individually designed and also can be replaced separately. The third advantage is the first phase of the proposed algorithm: this phase is applicable not only to the extended VC schemes but also to the conventional VC schemes. The fourth advantage is that because the density of the cover images is adjustable, it is very helpful for modifying the display quality of the cover images.

The major contributions of our work are as follows: First, this is the first solution that addresses the pixel problem of the EVCS for general access structures. Second, we formulate the construction problem of VCS for general access structures as a mathematical optimization problem such that the problem can be solved by optimization techniques. Third, we have developed a hybrid method for EVCS constructions. By adopting the proposed stamping algorithm, all existing VC schemes can be modified to form their extended VC schemes without redesigning codebooks.

ACKNOWLEDGMENT

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. And they are especially grateful to Prof. Carlo Blundo for his kind help during the review process of this paper.

REFERENCES


Kai-Hui Lee received the Ph.D. degree in electronic engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2002. He joined the Department of Computer Science and Information Engineering, Ming Chuan University, Taipei, Taiwan, as an associate professor in 2003. His current research interests include visual cryptography, wireless networks, and network resource management.

Pei-Ling Chiu received the Ph.D. degree in information management from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2007. She joined the Department of Risk Management and Insurance, Ming Chuan University, Taipei, Taiwan, as an associate professor in 2008. Her research focuses on visual cryptography, wireless sensor networks, and optimizing technologies.