Visual Secret Sharing between Remote Participants

Neil Buckley
Dept of Mathematics and Computer Science
Liverpool Hope University
Liverpool, England

Atulya K. Nagar
Dept of Mathematics and Computer Science
Liverpool Hope University
Liverpool, England

S. Arumugam
National Centre for Advanced Research in Discrete Mathematics (n-CARDMATH)
Kalasalingam University
Krishnankoil, Tamilnadu, India

ABSTRACT
Visual cryptography and random grids are forms of visual secret sharing that encrypt a secret image into indecipherable shares. Decryption occurs by printing them onto transparencies and stacking, but this requires participants to be in the physical presence of each other, so this paper addresses the use of visual secret sharing between remote, incommunicado agents. To this end, a prototype application has been developed to form a subpixel matrix of a photographed share that is one half of a (2, 2) scheme. It is algorithmically "stacked" with its stored complement to decrypt the secret. The implemented algorithms are presented, as well as visual results for variations of three values of three photographic condition metrics. Although only a third of the total results proved positive, recommendations are given regarding photographic conditions to significantly improve accuracy. Furthermore, we suggest a number of applications of this technology.

Keywords
Secret Sharing, Visual Cryptography, Random Grids, Augmented Reality, Mobile Technology

1. INTRODUCTION
Reliable identity authentication is a mounting concern in a world with increasing reliance on online services. Despite a decrease in online banking fraud in recent years, still £35.4 million were lost in fraudulent UK transactions in 2011 [1].

To meet this threat, banks have issued customers hand-held devices of various types. For example, HSBC's SecureKey uses its synchonrised internal clock and customer credentials to generate a code the user types to access an online account. This has, however, been met with much frustration [10].

This study proposes the visual cryptographic (VC) paradigm, formulated by Moni Naor Adi Shamir in 1994 [11], as well as the preferable (though ironically older) random grid method of Kafri and Keren [7], as authentication solutions.

Both of these methods are flavours of visual secret sharing (VSS), which separates a secret image into n shares, each a random sequence of dots, or subpixels. When k shares are printed onto transparencies and stacked, a contrast-reduced version of the secret is visually discernible.

Although reconstruction contrasts are similar for both VC and RG, the former entails further quality loss in the form of pixel expansion, whereby more than one subpixel is required to reconstruct each original secret pixel. RG does not have this problem, and several algorithms have been proposed recently [2, 14] that have brought them to the forefront of VSS research, due to increased ability to conceal secrets in access structures.

The perfect physical stacking of shares, which is here termed conventional VSS, is the ideal situation resulting in the best possible contrast in a given (k, n)-VSS scheme (VSSS). However, researchers such as [8, 9], investigate imperfect VC.

Handling imperfections lies at the heart of this research, entailing the computational stacking of a stored binary share onto a share displayed on a computer monitor (or printed) and photographed with a mobile phone.

A key advantage to using this type of cryptography is the possibility of assigning access structures to the revealing of a secret, for example if there are n participants holding shares, and only certain subsets of them are permitted to unlock the secret.

2. AIMS AND OBJECTIVES
The aim of this work is to devise effective algorithms for converting a photographed share image into a binary subpixel matrix. This matrix is then algorithmically "stacked" onto its accompanying matrix to reveal a secret image portraying a numeric code. This entails quantification and variation of various photographic conditions, which is used to compare results and make recommendations regarding the use of this technology.

The paper is organised as follows:

- In Section 4.1, the implementation of a prototype mobile application and the initial raw image preprocessing is described.
- In Sections 4.2-3, a novel computer vision algorithm is presented to approximate the boundaries of the visual share image and estimate its subpixel values.
- In Section 4.4, describe the computational stacking of the shares is explored.
- As part of the discussion in Section 5, security, robustness and computational complexity are discussed.
- Here also, applications to identity authentication are proposed and analysed.

3. RELATED WORK
The work of [9] bears similarity to this study in its attempts to extract information from a digitized, hence distorted digital share. However, they first print the share and use Fourier transform to correct the image.

They identify the share image in the scene "through the presence of peaks in the Fourier domain", taking advantage of the fact that a physical translation of the image is mathematically equivalent to a linear phase modulation in the Fourier space.

One cannot disregard the similarity of this work to QR-code recognition. Indeed, [3] brings about offline QR recognition using VC. Particularly interesting is their suggestion of using QR-like calibration symbols in shares to aid stacking and ameliorate the alignment problem of VC.

The alignment problem is further addressed by [8], who discuss the inherent robustness of VC using pixel expansion $m>2$, with non-zero contrast up to a maximal misalignment. They derive an exact formula for the resulting contrast of misaligned (n, n)-VCS, but concede difficulty for (k, n)-VCS if $k<n$. However, they concede that when individual subpixels are expanded before being printed onto transparencies, this in itself increases robustness.

In the spirit of stacking a physical share onto a digital one, [4] propose an application in an alternative to mobile phone passwords, instead stacking a VC transparency onto the screen displaying the accompanying share in a (2, 2)-VCS.

The revealed "image", in their proposal, is not pictorial, but reveals a statistically significant conglomeration of black pixels to the left, right, top or bottom of the display, inviting the user to swipe the screen in the required direction. Importantly, they cite this as an implementation of increased security in "something you have", as opposed to "something you know".

In VC implementations with full or partial computational decryption, as in the case of [9], the reconstruction can be de-noised to arrive at a better approximation to the original image. However, to aid a cleaner, higher contrast reconstruction, [13] was the first to propose the *XOR* binary operation for stacking, as opposed to *OR*.

Although this is difficult to bring out in physical share stacking, they say it is possible by polarizing light, creating a perfect reconstruction for (n, n)-VCS, and almost perfect if $k<n$.

Sivasankari and George [12] take advantage of such *XOR*-based VC in the concealment of a "subliminal message" in a secret colour image. This extra secret is hidden in the pixel colour values before channel decomposition and share generation. The reconstruction is thus close enough to the original to accurately extract the subliminal, which would be extremely difficult with a noisy *OR* reconstruction.

4. PROPOSED METHOD

4.1 Preprocessing

A (2, 2)-VSSS is used, i.e. $k=2$, $n=2$, comprising a secret image, $\mathcal{I}$ split into shares $\mathcal{H}^0$ and $\mathcal{H}^1$, such that $\mathcal{H}^0 \oplus \mathcal{H}^1 = \mathcal{I}'$, where $\oplus$ denotes *XOR* and $\mathcal{I}'$ is the reconstructed secret. However, when $\mathcal{H}^0$ is photographed, it becomes $\mathcal{H}'^0$, which is used in this paper to denote the photographed share itself, and the resultant subpixel matrix reconstruction.

If ij is the (row, column) coordinate of a subpixel, then $\mathcal{H}^{0,1}_{ij} \in \{0,1\}$ with 0 and 1 resp. white (or transparent) and black. However, this study instead proposes equating them to RGB (255, 127, 0) and (0, 127, 255), respectively, rendering them statistically distinguishable from the background and from each other, given that these combinations maximize the total colour distance, i.e.

$$\max(\Delta) = (G-R)+(B-G).$$

However, only $\mathcal{H}^0$ is displayed on the screen. $\mathcal{H}^1$ is stored in memory as a binary matrix. As stacking is computational, the user has no need to view the second share.

With the share displayed, the user takes a photograph of the screen with a mobile phone. Figure 1 shows the screen and resulting photograph. Note the degradation in image quality.

Here, also, the advice in [8] is heeded, expanding each subpixel to approximately a 5×5 pixel block (depending on the size of the share), increasing its salience.

To remove non-share pixels from the photograph, Algorithm 1 is given, performing a pixel-by-pixel and block-by-block colour analysis.

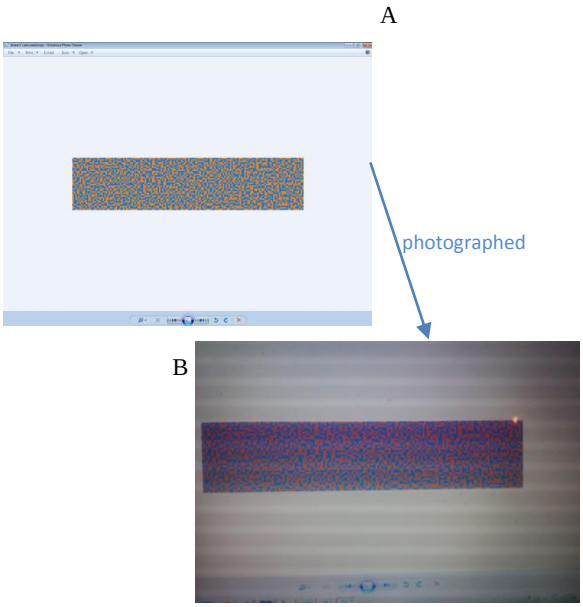


Fig 1: A: original $\mathcal{H}^0$ and B: photographed $\mathcal{H}'^0$

Algorithm 1: Removal of Non-share Pixels

Input: Photograph including $\mathcal{H}'^0$

Output: Isolated $\mathcal{H}'^0$

For *each vertical screen pixel, y,* ***do****,*
  For *each horizontal screen pixel, x,* ***do****,*
    $\{R, G, B\} \leftarrow$ *additive pixel colour components*
    $\Delta_0 \leftarrow (G-R)+(B-G)$
    $\Delta_1 \leftarrow (R-G)+(G-B)$
    If $\Delta_0 < 50$ *and* $\Delta_1 < 50$ *,* ***then****,*
      $pixel[x][y] \leftarrow$ *white*
    End If
  End For
End For

For each *vertical 15-pixel block, cy,* ***do****,*
  For *each horizontal 15-pixel block, cx,* ***do****,*
    $colour1Count \leftarrow 0$
    $colour2Count \leftarrow 0$
    For each *vertical block pixel, y,* ***do****,*
      For each *horizontal block pixel, x,* ***do****,*

```
                {R, G, B} ← pixel value
                If R<G and G<B then,
                    colour1Count ← colour1Count + 1
                Else If R>G and G>B, then,
                    colour2Count ← colour2Count + 1
                End If
            End For
        End For

        If colour1Count < 2 or colour2Count < 2, then,
            block[cx][cy] ← white End If
    End For
End For
```

Note that the various parameter choices, i.e. 50, 15 and 2, were arrived at experimentally. The results of applying the algorithm to $\mathcal{H}'^0$ in Figure 1B is given in Figure 2.

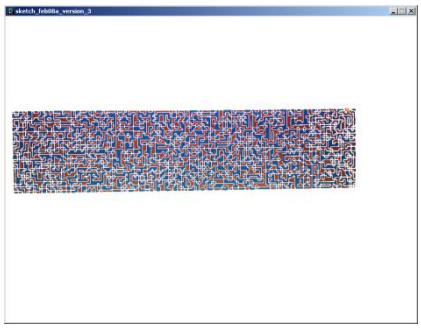


Fig 2: Result of Algorithm 1 for non-share pixel removal

It is clear that many share pixels have also been removed, making Algorithm 1 alone unsuitable for image preprocessing, however it can be effectively used to determine the margins, within which the share is approximately located. For convenience, these are denoted $M_i, i \in \{1,...,4\}$, moving clockwise from the top margin.

On the other hand, one could argue for a simplification in preprocessing by constraining the implementation to pure white pixels surrounding the share out to a reasonable distance and stipulating that the user photograph the share to be approximately centrally positioned.

In either case, a white border is drawn, concealing any interface-related components.

4.2 Side and Corner Location

Let us denote corner points, counting clockwise from the top left corner of $\mathcal{H}^0$ as $\left\{C^1_{xy},...,C^4_{xy}\right\}$. When photographed, they translate to $\left\{C'^1_{xy},...,C'^4_{xy}\right\}$.

Due to camera positional variation, $C'^i_{xy} \neq C^i_{xy}, i \in \{1,...,4\}$, and because of variation of camera distance and rotation, $C'^j_{xy} - C'^i_{xy} \neq C^j_{xy} - C^i_{xy}, i, j \in \{1,...,4\}$, so the first task is to locate the corners. This is achieved by locating a respective edge and algorithmically "rolling" along the side toward the desired corner.

Algorithm 2 is presented as the first step toward locating the first candidate for C^1, denoted $\{\mathcal{X}^1, \mathcal{Y}^1\}$. It analyses colour value differences Δ_1, Δ_2 (i.e. G-R and B-G) from the field of view's horizontal centre, starting at vertical point M_1 and working downward until constraints on $\{\Delta_1, \Delta_2\}$ are satisfied. It then moves leftward, continually analysing $\{\Delta_1, \Delta_2\}$ until its constraints fail a sufficient number of times for it to assume it has reached the corner.

(Highlighted in red are the parts of the algorithm that vary depending on respective corner to be located.)

Algorithm 2: Horizontal Location for Corner Candidate 1

```
searchLowerBound ← M1
searchUpperBound ← windowHeight/2
boundMargin ← 8
Θ ← default initial threshold
numFailures ← 0, maxFailures ← 5

Define Function calcPixelValDiff(x, y)
    {R,G,B} ← colour components of pixel (x, y)
    Return |G-R| + |B-G|
End Function

//HORIZONTAL SEARCH
For x = windowWidth/2 to M4 (working backward), do,
    found ← false
    For y = searchLowerBound to searchUpperBound, do,
        //TEST CURRENT AND NEARBY PIXELS
        If calcPixelValDiff(x, y+yd) > Θ for any yd from 0
          to 4, then,
            Mark the (x, y) location with a white spot
            Y1 ← y
            found ← true , numFailures ← 0
            //LIMIT SEARCH TO EASE COMPUTATIONAL
              COMPLEXITY
            searchLowerBound ← y – boundMargin
            searchUpperBound ← y + boundMargin
            break from this loop
        End If
    End For

    If not found, then numFailures ← numFailures + 1
    If numFailures > maxFailures, then,

        //SET X1 TO THE LAST POINT AT WHICH IT
          FOUND A VALID PIXEL COLOUR

        X1 ← x + maxFailures

        break from this loop (as now found X1)
    End If
End For
```

Similarly, Algorithm 3 locates a second candidate for this corner, analyzing $\{\Delta_1, \Delta_2\}$ from $x = M_4$ and moving upward toward C'^1_{xy}.

Algorithm 3: Vertical Location for Corner Candidate 2

searchLowerBound ← M_4
searchUpperBound ← $windowWidth/3$ *//(OR 2)*
numFailures ← 0
//VERTICAL SEARCH
For $y = windowHeight/2$ *to* M_1 *(working backward),* ***do***,
  found ← *false*
  For *x = searchLowerBound to searchUpperBound,* ***do***,
    //TEST CURRENT AND NEARBY PIXELS
    If *calcPixelValDiff(x+xd, y) > Θ for any xd from 0 to 4,* ***then***,
      Mark the (x, y) location with a white spot
      $\mathcal{X}^2 \leftarrow x$
      found ← *true* , *numFailures* ← 0
      searchLowerBound ← *x – boundMargin*
      searchUpperBound ← *x + boundMargin*
      break from this loop
    End If
  End For

  If *not found,* ***then*** *numFailures* ← *numFailures* + 1
  If *numFailures > maxFailures,* ***then***,
    $\mathcal{Y}^2 \leftarrow y + maxFailures$
    break from this loop (as now found $\mathcal{Y}^2$ *)*
  End If
End For

Algorithm 4 arrives at the third candidate by taking the average of the first two.

Algorithm 4: Corner Candidate 3

$$\mathcal{X}^3 \leftarrow \frac{\mathcal{X}^1 + \mathcal{X}^2}{2} - 4$$

$$\mathcal{Y}^3 \leftarrow \frac{\mathcal{Y}^1 + \mathcal{Y}^2}{2} - 4$$

Note the small offset of four pixels, which aids in the final corner estimate in Algorithm 5, in which the final candidates are "nudged" closer to the corner based on $\{\Delta_1, \Delta_2\}$.

Algorithm 5: Final Corner Estimate

$\mathcal{X}^4 \leftarrow \mathcal{X}^3, \mathcal{Y}^4 \leftarrow \mathcal{Y}^3$
found ← *false*
nudgeMargin ← 8
While *not found,* ***do***,
  For *xd = 0 to nudgeMargin,* ***do***,
    If *calcPixelValDiff(* $\mathcal{X}^4$ *+ xd,* $\mathcal{Y}^4$ *)* > Θ , ***then***,
      found ← *true*
      break from this loop
    End If
  End For

  If *not found,* ***then*** $\mathcal{X}^4 \leftarrow \mathcal{X}^4 + 1$
End While

found ← *false*
While *not found*
  For *yd = 0 to nudgeMargin,* ***do***,
    If *calcPixelValDiff(* $\mathcal{X}^4, \mathcal{Y}^4 + yd$*)* > Θ , ***then***,
      found ← *true*
      break from this loop
    End If
  End For

  If *not found,* ***then*** $\mathcal{Y}^4 \leftarrow \mathcal{Y}^4 + 1$
End While

$C'^1_{xy} \leftarrow \{\mathcal{X}^4, \mathcal{Y}^4\}$

The other corners are similarly located. However, because $\{\Delta_1, \Delta_2\}$ is compared to Θ, the value of Θ must be carefully chosen. Given the unpredictability of photographic conditions, this is initialized (to 50), the corners are located, and those coordinates are used to approximate the area. If the area lay below some minimum, Θ increases. This process continues until the area approximates an expected value. The result comprises Figure 3.

Fig 3: Side and corner location. (Note the algorithm's fault-tolerance with regards to the glare effect at the top right of the share.)

4.3 Grid Calculation

With $C'^i_{xy}, i \in \{1, ..., 4\}$, a grid of subpixels are built, i.e. $\mathcal{H}'^0$, approximating $\mathcal{H}^0$. Share subpixel coordinates $\mathcal{H}^{0,1}_{ij}, 2 \le j \le w, 2 \le i \le h$ imply there are w and h subpixels spanning the resp. horizontal and vertical dimensions of the share. However, it is first necessary to focus on the dimensions of an individual subpixel.

Given the inevitability of skew in $\mathcal{H}'^0$, subpixel dimensions vary, but all equal dimensions along a respective side are assumed. Working with sides 1 to 4, beginning with the top and working clockwise, subpixel dimensions are denoted $\omega_i, \eta_i, i \in \{1, ..., 4\}$ for resp. width and height.

Algorithm 6 approximately maps $\mathcal{H}'^0$ back to $\mathcal{H}^0$, i.e. determines the colour values of the individual subpixels.

Note the division of $\mathcal{I}$ into a grid of "sectors", S, such that $s_{ij} \in S, 1 \le j \le c, 2 \le i \le 2l$, where l is the number of lines of text and c is the number of characters on each line, assuming a portrayal of alphanumeric text.

Algorithm 6: Share 0 Reconstruction

$$\omega_1 \leftarrow \frac{C'^2_x - C'^1_x}{w}, \omega_3 \leftarrow \frac{C'^3_x - C'^4_x}{w}$$

$$\eta_4 \leftarrow \frac{C'^4_y - C'^1_y}{h}, \eta_2 \leftarrow \frac{C'^3_y - C'^2_y}{h}$$

$$widthDiff \leftarrow \omega_1 - \omega_3 \quad widthDiff \leftarrow \eta_4 - \eta_2$$

$$topGradient \leftarrow \frac{C'^2_y - C'^1_y}{C'^2_x - C'^1_x}$$

$$bottomGradient \leftarrow \frac{C'^3_y - C'^4_y}{C'^3_x - C'^4_x}$$

$$meanGradient \leftarrow \frac{topGradient + bottomGradient}{2}$$

$$\mathcal{H}'^0 \leftarrow empty \ \ w \times h \ \ matrix$$

Define Function *calcSubpixelColour*
(Receives parameters cx, cy, ω, η)
blackCount, whiteCount $\leftarrow 0$
For *py = 0 to* η, ***do***,
For *px = 0 to* ω, ***do***,
$\{R, G, B\} \leftarrow$ *colour components of pixel at coordinate (cx + px, cy + py)*
$\Delta \leftarrow (G - H) + (B - G)$
If Δ *> 20 and (B > G or G > R),* ***then***,
$blackCount \leftarrow blackCount + 1$
Else,
$whiteCount \leftarrow whiteCount + 1$
End If

If *blackCount – whiteCount > 10,* ***then*** *return 1*
Else If *whiteCount – blackCount > 10,* ***then*** *return 0*
End For
End For

If *blackCount – whiteCount > 30,* ***then*** *return 1*
Else, *return 0*
End Function

For each *horizontal subpixel, x,* ***do***,
$skewedCellHeight \leftarrow \omega_1 - heightDiff \,.\, y/h$
For each *vertical subpixel, y,* ***do***,
$skewedCellWidth \leftarrow \omega_1 - widthDiff \,.\, y/h$
$skewedLeftGap \leftarrow C'^1_x + y/h \left(C'^1_x - C'^4_x\right)$
$cellX1 \leftarrow skewedLeftGap + \Phi_1 x.skewedCellWidth$
$cellY1 \leftarrow C'^1_y + \Phi_2 y.skewedCellHeight ...$
$... + x.skewedCellWidth.topGradient$
$cellX2 \leftarrow cellX1 + \Phi_1.skewedCellWidth$
$cellY2 \leftarrow C'^1_y + \Phi_2 y.skewedCellHeight + ...$
$...(x+1).skewedCellWidth.topGradient$
$cellX3 \leftarrow cellX2$
$cellY3 \leftarrow cellY2 + \Phi_2.skewedCellHeight$
$cellX4 \leftarrow cellX1$
$cellY4 \leftarrow cellY3$
$\omega \leftarrow cellX2 - cellX1$
$\eta \leftarrow cellY4 - cellY1$
$\mathcal{H}'^0_{yx} \leftarrow calcSubpixelColour(cellX1, cellY1, \omega, \eta)$
End For
End For

Here, ω and η are approximated as the pixel length of the respective side divided by the number of subpixels along that side. Since this naïve approach results in such an imperfect grid, additional offsets (Φ_1, Φ_2) are introduced for resp. horizontal and vertical subpixel value approximations.

s_{ij} is defined as "locked" if less than a threshold number of its pixels remain black. With the execution of Algorithm 6 for $\Phi_1 = \Phi_2 = 0$, one would expect at least a minimum number of sectors to be resolved. If not, $\Theta \leftarrow \Theta + 5$ and Algorithms 2 to 5 are re-executed.

Given such imperfect conditions for VC, it is unlikely all sectors are locked after the first run. Therefore, the value of Φ_1 and/or Φ_2 are iteratively changed by $\Delta\Phi_i, i \in \{1, 2\}$ and Algorithm 6 is re-executed. As a rule of thumb:

$$w \gg h \Rightarrow \Delta\Phi_1 \approx 1, \Delta\Phi_2 = 0,$$

$$h \gg w \Rightarrow \Delta\Phi_1 = 0, \Delta\Phi_2 \approx \pm 0.002,$$

$$w \approx h \Rightarrow \Delta\Phi_1 \approx 1, \Delta\Phi_2 \approx \pm 0.002. \quad (1)$$

The sign of $\Delta\Phi_i$ is initialized positive, but when it reaches bounds of 0.95 and 1.15, its sign reverses.

The result of grid calculation comprises Figure 4, (although the grid lines are for demonstration purposes only).

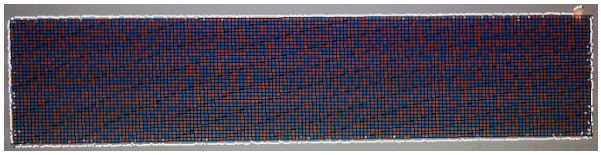


Fig 4: Calculated grid overlaid onto $\mathcal{H}'^0$

4.4 Share Stacking

With Share 0 approximation ($\mathcal{H}'^{,0}$) in place, VC stacking can be carried out with the already stored Share 1 ($\mathcal{H}^1$) to obtain the reconstructed $\mathcal{I}'$. Although physical stacking is analogous to binary *OR* calculation, it is pointed out by research such as [13] that *XOR* ($\oplus$) gives higher contrast. Indeed, they proved that if k=n, the theoretical reconstructed contrast is unity. Therefore, (2) is applied on each concurrent subpixel, (i, j):

$$\mathcal{I}'_{ij} = \mathcal{H}'^{,0}_{ij} \oplus \mathcal{H}^1_{ij} \tag{2}$$

Furthermore, due to the restrictions discussed in Section 4.3, each sector in $\mathcal{I}'$ is "locked" as soon as $H(S) > \theta$, where $H(.)$ denotes Hamming weight and θ is white-to-black ratio threshold. In experiments, θ was initialized to 0.3 and is increased slowly toward 0.5, effectively starting "strict" (i.e. requiring high contrast), but incrementally relaxing this constraint.

In this way, $\mathcal{I}'$ is gradually constructed, as illustrated below.

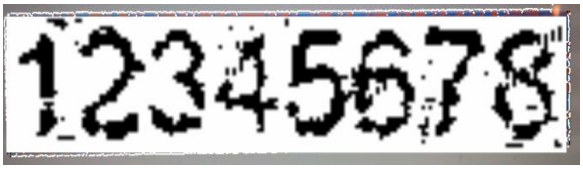


Fig 5: Calculated grid overlaid onto $\mathcal{H}'^{,0}$

5. DISCUSSION

5.1 Computational Cost and Robustness

A number of variations in photographic conditions were explored, most importantly distance, perspective, lighting and device. To these are added the metric of the ratio of w to h, i.e. dimensions of the share on the screen.

The impact of the distance of the camera from the monitor is closely tied to ω and η. With the recommended constraint on the user (discussed in Section 4.1) of ensuring $\mathcal{H}'^{,0}$ is approximately central and pervasive in the field of view, "distance", D, is simplified as:

$$D = \frac{w\omega.h\eta}{A(\mathcal{H}^0)} \tag{3}$$

where $A(.)$ denotes area (in pixels). Note here that $A(\mathcal{H}'^0) \approx w\omega.h\eta$, but the act of approximation makes little difference.

Viewing perspective, *P*, is here defined as the ordered pair:

$$P = \begin{pmatrix} \dfrac{\dfrac{C'^1_{(y)} - C'^0_{(y)}}{w\omega_0} + \dfrac{C'^2_{(y)} - C'^3_{(y)}}{w\omega_2}}{2}, \\ \dfrac{\dfrac{C'^3_{(x)} - C'^0_{(x)}}{h\eta_3} + \dfrac{C'^2_{(x)} - C'^1_{(x)}}{h\eta_1}}{2} \end{pmatrix} \tag{4}$$

where the respective elements are horizontal and vertical skew.

Lighting, *L*, is estimated as:

$$L = \frac{\sum_{i=1}^{width} \sum_{j=1}^{height} R'_{ij} + G'_{ij} + B'_{ij}}{\sum_{i=1}^{width} \sum_{j=1}^{height} R_{ij} + G_{ij} + B_{ij}} \tag{5}$$

where width and height are the window pixel dimensions, *{R, G, B}* are the original colour values, and their primes are the photographed colour values.

The devices considered comprised a smartphone set to both one and three megapixel mode, and a tablet device.

Experiments with photographs taken in one megapixel mode failed in all cases due to inability to resolve $\mathcal{H}'^{,0}$ subpixels. Interestingly, the tablet also failed in the vast majority of attempts due to photographic distortion resulting from monitor frame rate. Indeed, the result was a Moiré-like pattern, severely hindering the ability to locate edges and corners. Therefore, this study only considers three megapixel mode on a smartphone and leaves other devices to further research.

In addition, only results for w h are considered, as in Figure 1. Given the remaining dimensions of the robustness analysis, three discrete values of each are considered, with results illustrated in Tables 1, 2 and 3.

Note that $0.62 \le D \le 1.21$, representing distances of approximately 15 to 35 centimetres between the screen and the camera. Also, $0 \le |P| \le 0.04$, representing angles of approximately 0° to 30°, and $0.67 \le L \le 1.16$, representing minimal to full monitor backlight.

The application was run on a laptop PC with a dual-core 2.09 GHz processor and 3 GB RAM.

Table 1. Code recognition for high perspective values

|P| ≈ (4%, 4%)	
D = 0.62	*L= 0.7* *(FAILURE)*
	L = 0.8 *(1)*  *6.6 seconds*
	L = 1.11 *(FAILURE)*
D = 0.83	*L= 0.69* *(FAILURE)*
	L = 0.78 *(2)*  *25 seconds*
	L = 1.07 *(FAILURE)*
D = 1.21	*L = 0.69* *(FAILURE)*
	L = 0.8 *(FAILURE)*
	L = 1.06 *(FAILURE)*

Table 2. Code recognition for medium perspective values

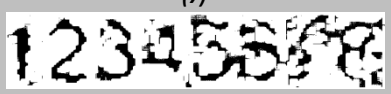
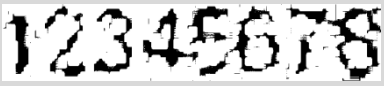
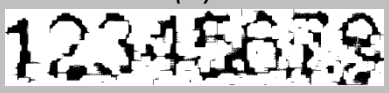
|P| ≈ (2%, 2%)	
D = 0.62	*L= 0.67* *(3)*  *13 seconds*
	L = 0.81 *(4)*  *12 seconds*
	L = 1.10 *(FAILURE)*
D = 0.83	*L= 0.69* *(5)*  *10.3 seconds*
	L = 0.8 *(6)*  *21 seconds*
	L = 1.16 *(FAILURE)*
D = 1.21	*L = 0.67* *(FAILURE)*
	L = 0.77 *(7)*  *27.7 seconds*
	L = 1.08 *(FAILURE)*

Table 3. Code recognition for low perspective values

|P| ≈ (0%, 0%)	
D = 0.62	*L= 0.71* *(8)*  *18.3 seconds*
	L = 0.81 *(9)*  *17.5 seconds*
	L = 1.10 *(FAILURE)*
D = 0.83	*L= 0.69* *(10)*  *13.2 seconds*
	L = 0.8 *(11)*  *13 seconds*
	L = 1.16 *(12)*  *13.1 seconds*
D = 1.21	*L = 0.67* *(13)*  *22.3 seconds*
	L = 0.77 *(FAILURE)*
	L = 1.08 *(14)*  *26 seconds*

Here, 14 of the 27 experiments resulted in at least partial success, but if results 2, 4, 7 and 14 are also discarded for illegibility of at least one of the eight characters, the total success rate is 37%, in an average time of 14.6 seconds.

Table 4 likewise summarises (rounded percentage) success rates and average times for the respective "low", "medium" and "high" values of *P*, *D* and *L*.

Table 4. Mean recognition rates for low, medium and high perspective, distance and lighting, respectively, for the metrics of perspective (*P*), distance (*D*) and lighting (*L*)

	Low	**Medium**	**High**
P	67% (16.2 sec.)	44% (14.1 sec.)	11% (6.6 sec.)
D	44% (13.5 sec.)	56% (14.1 sec.)	11% (22.3 sec)
L	56% (15.4 sec.)	67% (14 sec.)	11% (13.1 sec.)

It is clear that low perspective results in higher recognition rates, but a highly skewed $\mathcal{H}^{,0}$ can still be decrypted if the camera is placed close to the screen. In general, however, medium distance is preferable. The same is true for lighting. Furthermore, in all cases, high values resulted in poor recognition.

It is evident from the timings that "ideal" conditions resulted in the code being decrypted in approximately 14 seconds.

5.2 Real-world Applications

The proposed methodology can find use in any situation requiring sharing of a binary image or code, such as a password or PIN number securely over the Internet, as well as using a mobile phone to photograph a printed share.

Two examples are given. The first is in banking, providing an alternative to login devices. A bank, before sending such a device to a customer, has its internal clock synchronized with that of the bank and is loaded with some customer credentials. When the customer attempts to log in to an online account and switches on the device, both the server and device use a secret formula, based on current time and credentials, to generate an eight-digit code the customer inputs to access the account.

This is regarded as secure, as it relies on "something you have", versus knowledge alone, which due to its ethereal nature can more easily fall into the wrong hands [4]. Hence, it provides greater security.

However, the use a mobile phone is suggested here to perform the same action. The proposed implementation comprises Figure 6.

It is important to note that both the system and the app must not only generate the same verification code, but valid complementary $\mathcal{H}^i, i \in \{0,1\}$ of a (2, 2)-VCS. In conventional VC, each basis matrix column permutation is random, but to realize the above system, the results must be deterministic. Hence, a pseudorandom sequence generator can be used, which common to both agents, taking its seed from the same input parameters, i.e. user credentials and current time.

It is of course wise to approximate "current time" to within several minutes, giving the customer enough time to login.

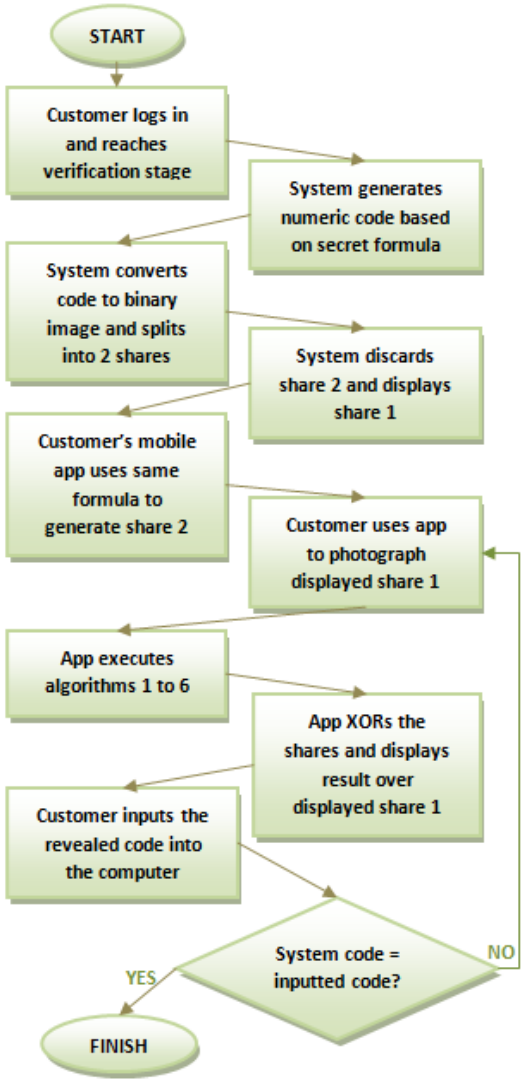


Fig 6: Banking implementation for login code generation and identity verification

Heretofore, this study have focused on a digitized version of $\mathcal{H}^0$, but the algorithms are applicable if this share is printed. Taking account of the constraints on *D*, *P* and *L*, such usage could be used much like QR-codes, but with an explicitly graphical element.

In this context, a second application is proposed, to the dissemination of door access codes to authorized personnel. Let us imagine that staff member Alice must gain access to room B101, but due to heightened security, its digital keypad changes regularly and unpredictably.

To ensure authorized staff have the correct code, a copy of $\mathcal{H}^0$ is printed and fixed permanently outside the door. Given that this share is permanent and unchanging, it is desirable to use random grids, as discussed by researchers such as [2]. Therefore, for this implementation, $\mathcal{H}^0$ is randomly generated. Treating this as a share with pixel expansion 2×2, the codebook in Table 5 is used to generate $\mathcal{H}^1$ based on $\mathcal{H}^0$ and $\mathcal{I}$.

Table 5. Share $\mathcal{H}^1$ generation based on a random grid-based visual secret sharing

$\mathcal{I}$	$\mathcal{H}^0$	$\mathcal{H}^1$
□		
□		
□		
(Et cetera. I.e., copy the $\mathcal{H}^0$ subpixel matrix into $\mathcal{H}^1$.)		
■		
■		
■		
(Et cetera. I.e., copy the binary complement of the $\mathcal{H}^0$ subpixel matrix into $\mathcal{H}^1$.)		

Alice is given a private mobile app, which she must not share with colleagues. When the door code changes, the share held by her app is automatically updated. To access B101, she activates the app, points the camera at the mounted share image, and the app extracts the code. She then inputs the code into the keypad to gain access.

The advantage of using this methodology, as opposed to, for example, the generation of key codes based on credentials, is two-fold. Firstly, if using random grids, one grid can be assigned as a "master share", able to unlock any of infinite number of secret images by *XOR* stacking with a grid associated with that the respective secret.

Secondly, access structures can be assigned to the sharing of the secret. For example, the room access in the example above might only be permitted in the presence of one employee and one manager, whose mobile apps could communicate their binary matrices with each other and digitally *XOR*, before being "stacked" with the printed share using Algorithm 2 to 6.

6. CONCLUSION AND FURTHER WORK

This paper has proposed a series of algorithms implementing visual cryptography, where share holders are not in each other's presence. This entails either the use of random grid or two separate systems of independently created complementary shares of a (2, 2) visual cryptographic scheme.

This research has resulted in and been tested on a prototype able to scan a photographed share. Within error margins, it successfully discerns individual subpixels, stores them in a binary matrix and stacks it with the complimentary stored share to retrieve the secret image.

This study has experimented with varying parameters values, i.e. distance, perspective, lighting and device, and discovered limitations. Future work will therefore address these shortcomings, increasing the robustness to varying photographic conditions on different devices. Indeed, the advice of [3] will be considered to employ calibration patterns embedded within the share to aid recognition.

7. REFERENCES

[1] APACS (2012) *Fraud – The Facts* [online] Available at: <http://www.financialfraudaction.org.uk> [Accessed 27th March 2014] pp 52-58

[2] Chen, T. & Tsao, K. (2011). *Threshold visual secret sharing by random grids*. The Journal of Systems and Software, 84(7). pp 1197-1208

[3] Fang, W. (2011). Offline QR Code Authorization Based on Visual Cryptography. *Seventh International conference on Intelligent Information Hiding and Multimedia Signal Processing*. pp 89-92

[4] Filardo, N.W. & Ateniese, G. (2011). *High-Entropy Visual Identification for Touch Screen Devices*. John Hopkins University, Baltimore. pp 1-16

[5] Ibrahim, M.H. (2012). *New Capabilities of Visual Cryptography*. International Journal of computer Science Issues, 9(5). pp 225-231

[6] Jin, D., Yan, W. & Kankanhalli, M.S. (2005). *Progressive color visual cryptography*. Journal of Electronic Imaging, 14(3). pp 1-13

[7] Kafri, O. & Keren, E. (1987) *Image encryption by multiple random grids*. Optical Letters 12(6). pp 377–379

[8] Liu, F., Wu, C.K. & Lin, X.J. (2008). *The alignment problem of visual cryptography schemes*. Designs, Codes and Cryptography, 50(2). pp 215-227

[9] Machizaud, J., Chavel, P. & Fournel, T. (2011). *Fourier-based automatic alignment for improved visual cryptography schemes*. Optics Express, 19(23). pp 22709-22722

[10] Murray-West, R.(2011) Facebook cambpaign by angry HSBC customers over new online security key, *The Telegraph* [online] Available at: <http://www.telegraph.co.uk/finance/personalfinance/consumertips/banking/8725302/Facebook-campaign-by-angry-HSBC-customers-over-new-online-security-key.html> [Accessed 27th March 2014]

[11] Naor, M. & Shamir A. (1994). *Visual Cryptography*. EUROCRYPT 1994. pp 1-12

[12] Sivasankari, S. & George, V.S. (2012). Implementation of Stenography within Visual Cryptography. *2012 International Conference on Computing and Control Engineering*. pp 1-5

[13] Tuyls, P., Hollmann, H.D.L., Van Lint, J.H. & Tolhuizen, L. (2005). *XOR-based Visual Cryptography Schemes*. Designs, Codes and Cryptography, 37(1). pp 169-186

[14] Wu, X. & Sun, W. (2013). *Improving the visual quality of random grid-based visual secret sharing*. Signal Processing, 93(5). pp 977-995