Both Shares in Color Visual Cryptography Can Be Statistically Indistinguishable from Noise

Leszek J Chmielewski[1], Mariusz Nieniewski[3] and Arkadiusz Orłowski[1,2]

[1] Warsaw University of Life Sciences – SGGW, Institute of Information Technology
[2] Military University of Technology – WAT, Institute of Mathematics and Cryptology
Warszawa, Poland

[3] University of Lodz, Faculty of Mathematics and Informatics
Łódź, Poland

ESM'2025

Ghent, Belgium, October 22-24, 2025

Contents

Cryptography and steganography

- Cryptography has been the subject of interest at least for several centuries
- Steganography, its branch: hiding a secret message in another, overt message
- Overt message is intended to hide the secret message, but it also distracts the third parties from the fact that something is hidden
- Even if the coding is unbreakable, it is still susceptible to the simplest attack – the cutoff of the transmission channel
- Therefore, hiding not only the secret, but also the fact that a transmission takes place, is of great value
- Transmitting a structured but unreadable message evokes interest of the third parties
- Our world is full of noise, so noise does not attract too much attention
- A concept of coding a color image in pure noise will be presented
- In the literature, little or no attention was paid to true randomness of shares; usually shares were described as *looking random*

Cryptography and steganography

- Cryptography has been the subject of interest at least for several centuries
- Steganography, its branch: hiding a secret message in another, overt message
- Overt message is intended to hide the secret message, but it also distracts the third parties from the fact that something is hidden
- Even if the coding is unbreakable, it is still susceptible to the simplest attack – the cutoff of the transmission channel
- Therefore, hiding not only the secret, but also the fact that a transmission takes place, is of great value
- Transmitting a structured but unreadable message evokes interest of the third parties
- Our world is full of noise, so noise does not attract too much attention
- A concept of coding a color image in pure noise will be presented
- In the literature, little or no attention was paid to true randomness of shares; usually shares were described as *looking random*

Cryptography and steganography

- Cryptography has been the subject of interest at least for several centuries
- Steganography, its branch: hiding a secret message in another, overt message
- Overt message is intended to hide the secret message,
  but it also distracts the third parties from the fact that something is hidden
- Even if the coding is unbreakable, it is still susceptible to the simplest attack – the cutoff of the transmission channel
- Therefore, hiding not only the secret, but also the fact that a transmission takes place, is of great value
- Transmitting a structured but unreadable message evokes interest of the third parties
- Our world is full of noise, so noise does not attract too much attention
- A concept of coding a color image in pure noise will be presented
- In the literature, little or no attention was paid to true randomness of shares;
  usually shares were described as *looking random*

Classic Visual Cryptography (VC)

Secret image

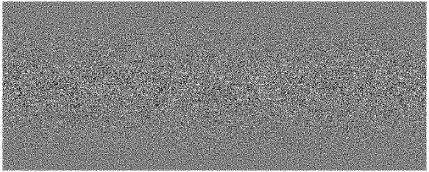
Details will be presented further

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

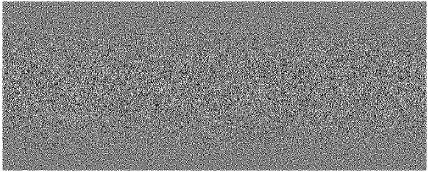
Classic Visual Cryptography (VC)

Secret image

↓ coding in two *shares*



+



Neither *share* contains information on the secret, but their structure is specific

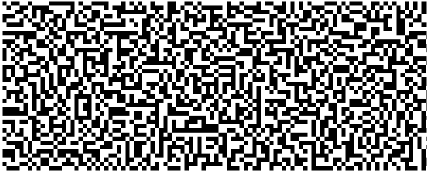
Details will be presented further

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

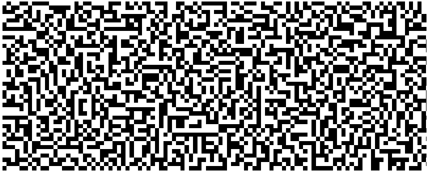
Classic Visual Cryptography (VC)

Secret image

↓ coding in two *shares*



+



Neither *share* contains information on the secret, but their structure is specific (UL corn. $\times 10$)

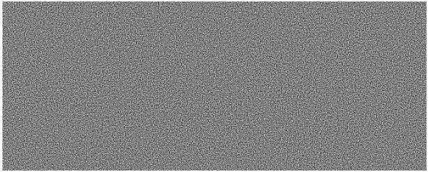
Details will be presented further

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

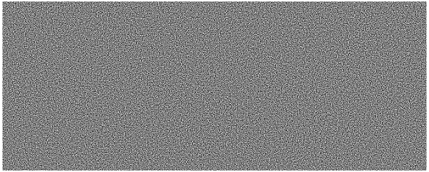
Classic Visual Cryptography (VC)

Secret image

↓ coding in two *shares*



+



↑ decoding by simply overlaying

Image decoded with unarmed eye

Neither *share* contains information on the secret, but their structure is specific

Details will be presented further

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

Meaningful shares

One of the concepts of distracting the third party from the coding process is *meaningful shares*

Secret image

Advanced mathematical theory, high quality color methods and multi-participant schemes exist

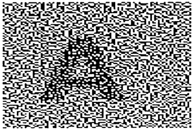
Concept, images: (Ateniese, Blundo, Santis, and Stinson 2001). Extensions: see e.g. (Dhiman and Kasana 2018).

Meaningful shares

One of the concepts of distracting the third party from the coding process is *meaningful shares*

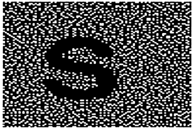
Secret image

↓ coding

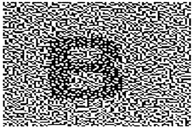


+

Decoded image



↑ decoding



Shares contain images irrelevant to the secret; their structure is still specific

Advanced mathematical theory, high quality color methods and multi-participant schemes exist

Concept, images: (Ateniese, Blundo, Santis, and Stinson 2001). Extensions: see e.g. (Dhiman and Kasana 2018).

Random shares in black-and-white

Secret image

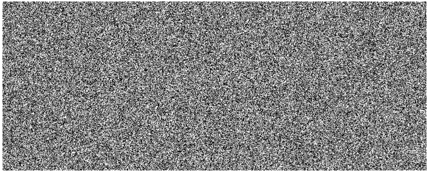
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

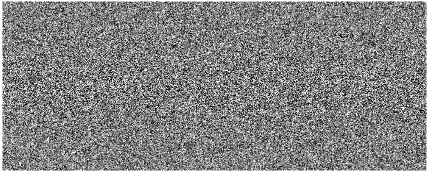
Random shares in black-and-white

Secret image

↓ coding



+



Neither *share* contains information on the secret; their contents is random

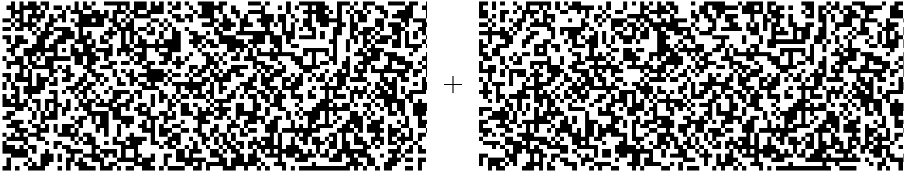
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

Random shares in black-and-white

Secret image

↓ coding



Neither *share* contains information on the secret; their contents is random (UL corn. ×10)

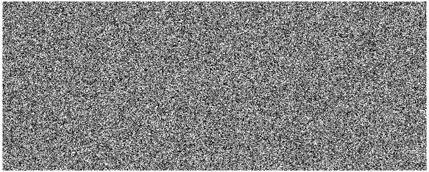
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

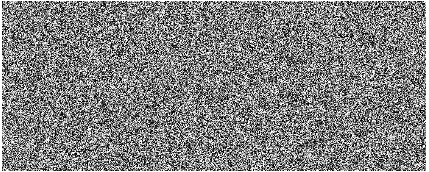
Random shares in black-and-white

Secret image

↓ coding



+



↑ decoding

Decoded, with inevitable errors

Neither *share* contains information on the secret; their contents is random

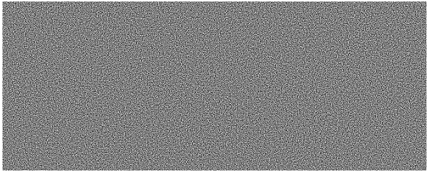
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

Errors caused by randomness

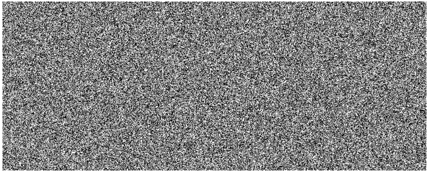
Decoded classically, no errors

↑ decoding



Decoded randomly, with errors

↑ decoding



Neither *share* contains information on the secret; their contents is random

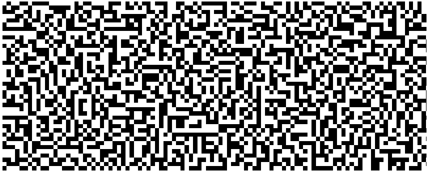
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

Errors caused by randomness

Decoded classically, no errors

↑ decoding



Decoded randomly, with errors

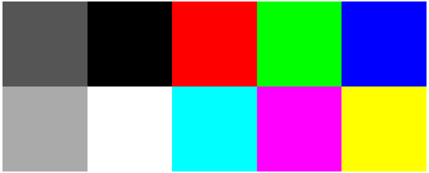
↑ decoding

Neither *share* contains information on the secret; their contents is random (UL corn. $\times 10$)
Details will be presented further

B-W: (Orłowski and Chmielewski 2019b). Color: (Chmielewski, Nieniewski, and Orłowski 2021a), CORES 2021.

Random shares in color

Secret image

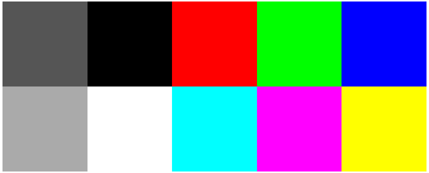


Details will be presented further

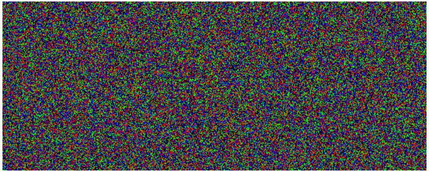
CORES '21 (Chmielewski, Nieniewski, and Orłowski 2021a)

Random shares in color

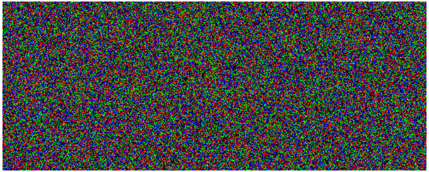
Secret image



↓ coding



+



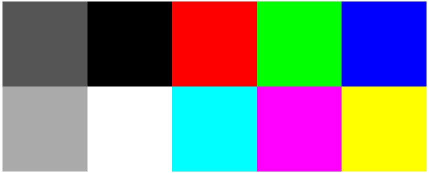
Shares have random contents made of R, G, B, K pixels

Details will be presented further

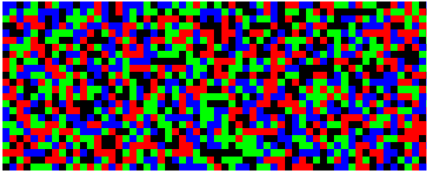
CORES '21 (Chmielewski, Nieniewski, and Orłowski 2021a)

Random shares in color

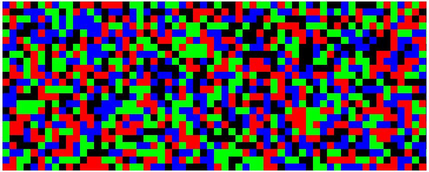
Secret image



↓ coding



+

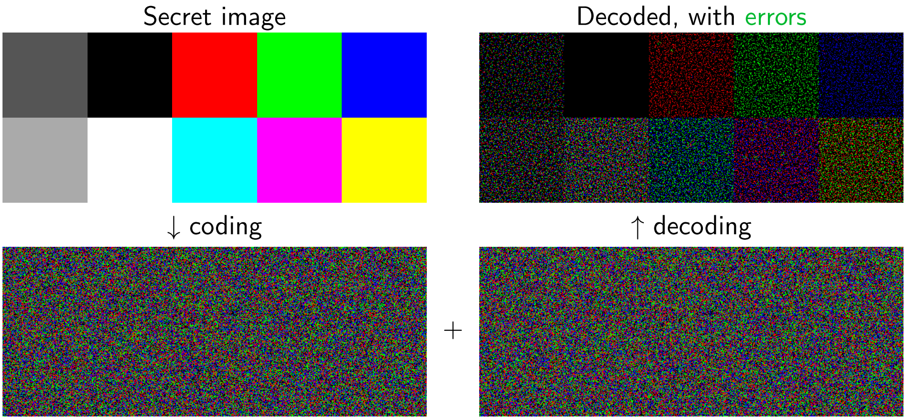


Shares have random contents made of R, G, B, K pixels (UL corn. ×10)

Details will be presented further

CORES '21 (Chmielewski, Nieniewski, and Orłowski 2021a)

Random shares in color



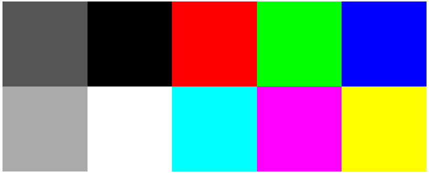
Shares have random contents made of R, G, B, K pixels

Details will be presented further

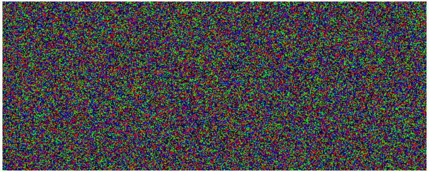
CORES '21 (Chmielewski, Nieniewski, and Orłowski 2021a)

Random shares in color

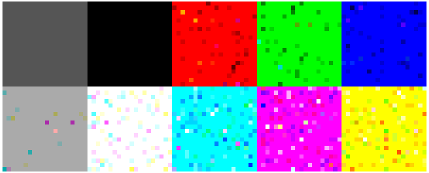
Secret image



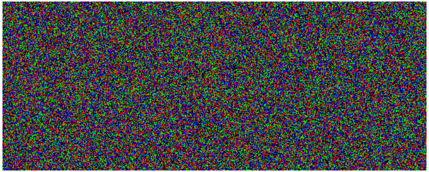
↓ coding



Can be numerically restored easily



↑ decoding



+

Shares have random contents made of R, G, B, K pixels

Details will be presented further

CORES '21 (Chmielewski, Nieniewski, and Orłowski 2021a)

Main content

- Classical coding of black-and-white images
- Coding of black-and-white images in random shares
- Randomness at a cost of errors
- Going to color: Coding of color images in random shares
- Two methods of coding: by hiding and by unhiding
- Testing the randomness – a simulated random experiment
- Histograms of p-values – small number of failures, histogram flatness
- Results

Main content

- Classical coding of black-and-white images
- Coding of black-and-white images in random shares
- Randomness at a cost of errors
- Going to color: Coding of color images in random shares
- Two methods of coding: by hiding and by unhiding
- Testing the randomness – a simulated random experiment
- Histograms of p-values – small number of failures, histogram flatness
- Results

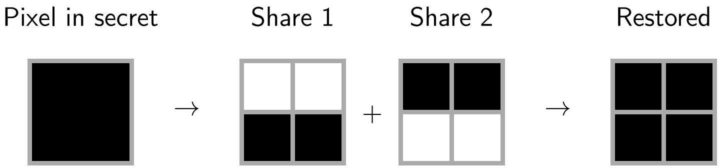
Main content

- Classical coding of black-and-white images
- Coding of black-and-white images in random shares
- Randomness at a cost of errors

- Going to color: Coding of color images in random shares
- Two methods of coding: by hiding and by unhiding
- Testing the randomness – a simulated random experiment
- Histograms of p-values – small number of failures, histogram flatness

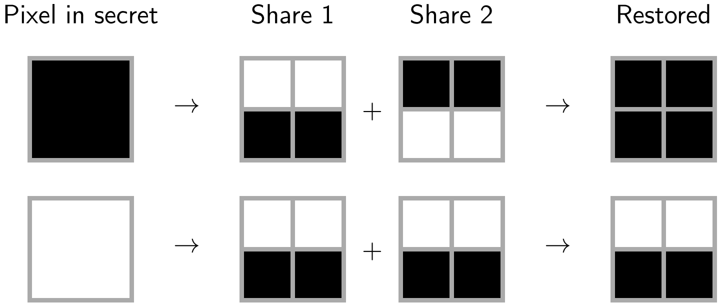
- Results

Classic visual cryptography explained



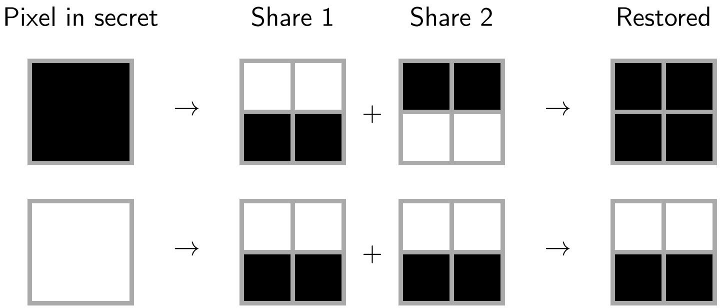
Source: (Naor and Shamir 1995; Naor and Shamir 1997).

Classic visual cryptography explained



Source: (Naor and Shamir 1995; Naor and Shamir 1997).

Classic visual cryptography explained



Share 1: drawn at random from *tiles* below. Share 2: chosen according to share 1 and secret.

Source: (Naor and Shamir 1995; Naor and Shamir 1997).

Classic visual cryptography – now we understand it

Secret image

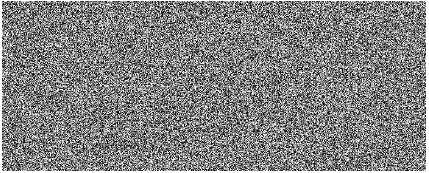
Share 1 is random within limits; share 2 is random similarly, and not correlated with secret.

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

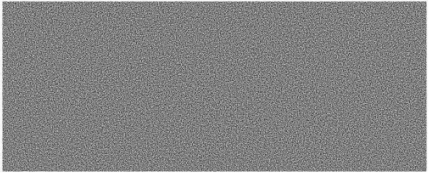
Classic visual cryptography – now we understand it

Secret image

↓ coding in two *shares*



+



Neither *share* contains information on the secret, but their structure is specific.

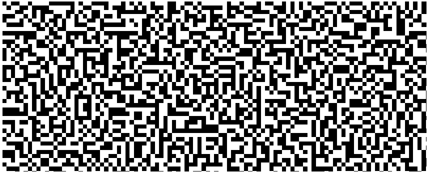
Share 1 is random within limits; share 2 is random similarly, and not correlated with secret.

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

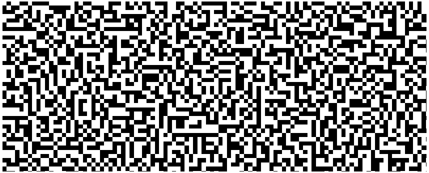
Classic visual cryptography – now we understand it

Secret image

↓ coding in two *shares*



+



Neither *share* contains information on the secret, but their structure is specific. (UL corn. ×10)

Share 1 is random within limits; share 2 is random similarly, and not correlated with secret.

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

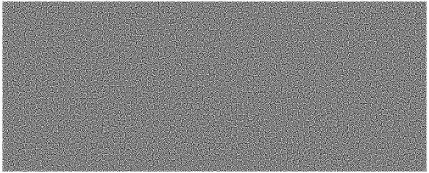
Classic visual cryptography – now we understand it

Secret image

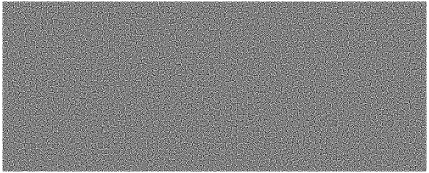
Image decoded, 2×2 larger

↓ coding in two *shares*

↑ decoding by simply overlaying



+



Neither *share* contains information on the secret, but their structure is specific.

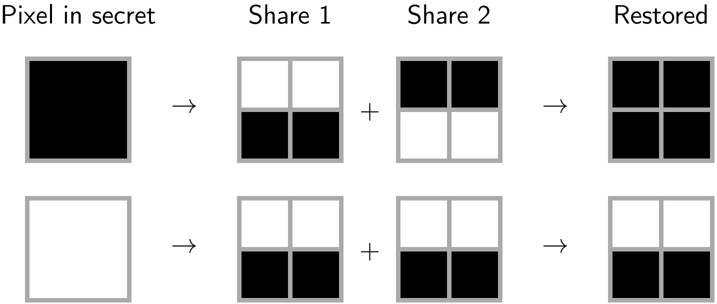
Share 1 is random within limits; share 2 is random similarly, and not correlated with secret.

Concept: (Naor and Shamir 1995; Naor and Shamir 1997). Images: (Orłowski and Chmielewski 2019a).

Random B-W visual cryptography explained

Classic: only tiles with 2 pixles white, 2 pixels black, for accurate coding:

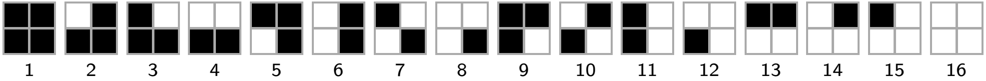
Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.

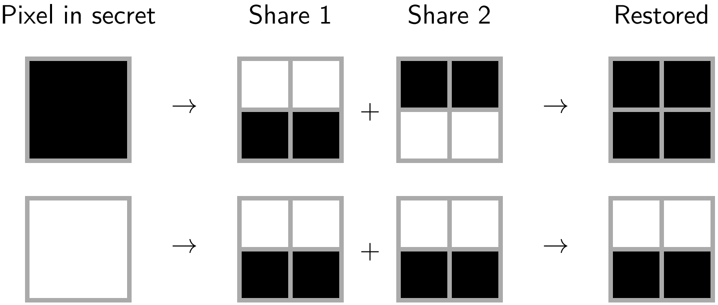


Random B-W visual cryptography explained

Random: with all possible tiles – errors possible:

Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.





we were lucky

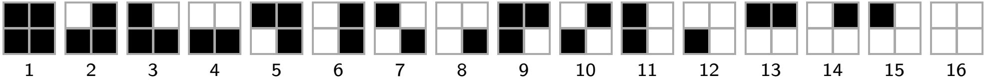
no errors

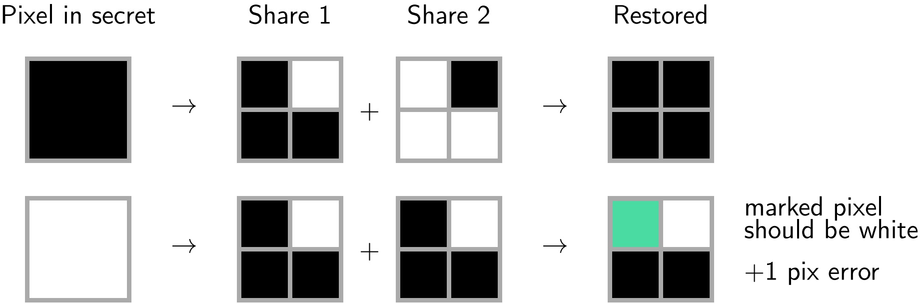
Source: (Orłowski and Chmielewski 2019a; Orłowski and Chmielewski 2019c).

Random B-W visual cryptography explained

Random: with all possible tiles – errors possible:

Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.



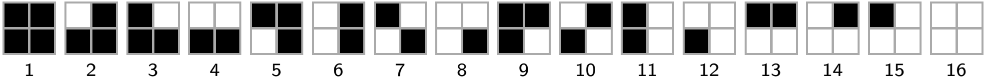


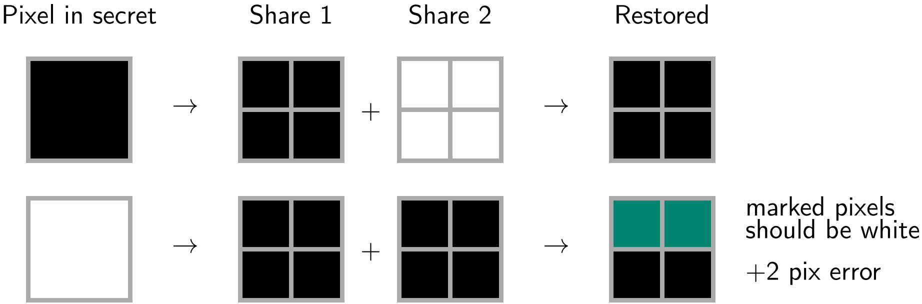
Source: (Orłowski and Chmielewski 2019a; Orłowski and Chmielewski 2019c).

Random B-W visual cryptography explained

Random: with all possible tiles – errors possible:

Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.



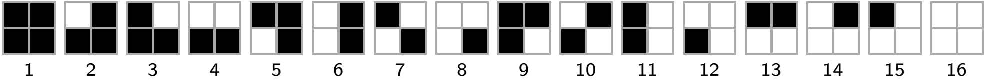


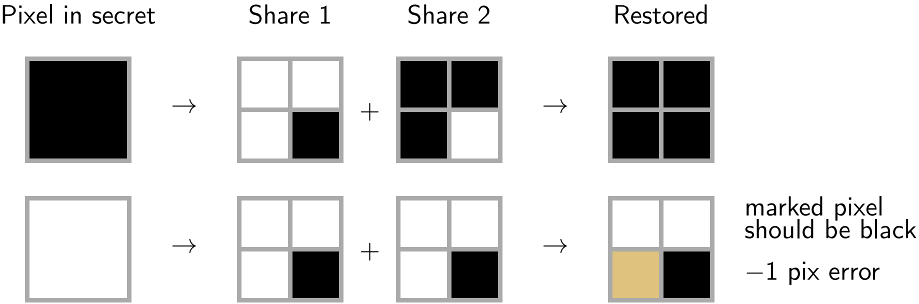
Source: (Orłowski and Chmielewski 2019a; Orłowski and Chmielewski 2019c).

Random B-W visual cryptography explained

Random: with all possible tiles – errors possible:

Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.



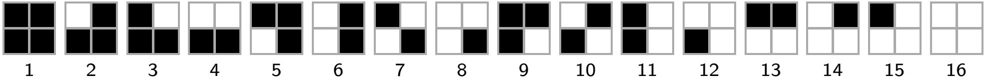


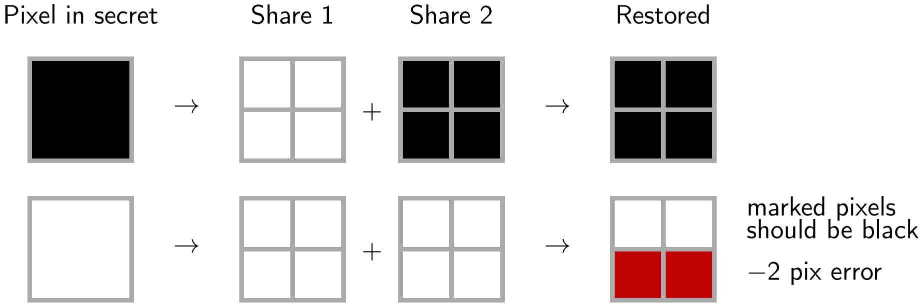
Source: (Orłowski and Chmielewski 2019a; Orłowski and Chmielewski 2019c).

Random B-W visual cryptography explained

Random: with all possible tiles – errors possible:

Share 1: drawn at random from tiles below. Share 2: chosen according to share 1 and secret.





Source: (Orłowski and Chmielewski 2019a; Orłowski and Chmielewski 2019c).

Random B-W visual cryptography: errors

Decoded classically

Decoded randomly, with errors

Random B-W visual cryptography: errors

Decoded classically

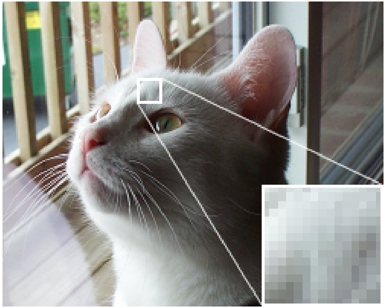
Decoded randomly, with errors

Errors: +1 pix, +2 pix, −1 pix, −2 pix

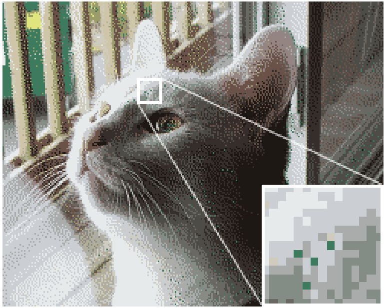
Introducing color

Dithering and color quantization

Original, full color



Dithered, 16-color palette



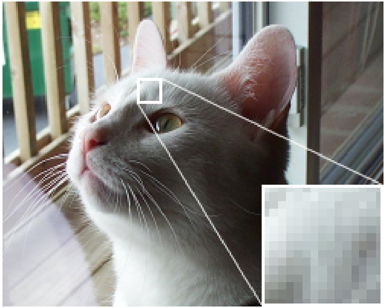
- Palettes with R, G, B and K (black) pixels only
- Transparencies treated as light emitting device, hence additive color model
- Dithering and color quantization is not in scope of this presentation

Source: (Wikipedia contributors 2021).

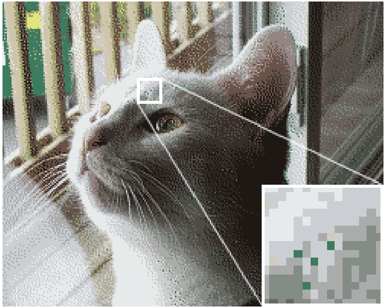
Introducing color

Dithering and color quantization

Original, full color



Dithered, 16-color palette



- Palettes with R, G, B and K (black) pixels only
- Transparencies treated as light emitting device, hence additive color model
- Dithering and color quantization is not in scope of this presentation

Source: (Wikipedia contributors 2021).

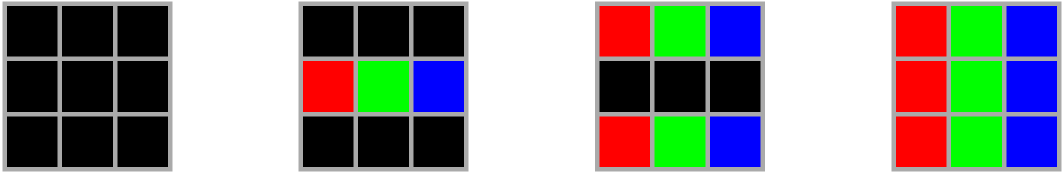
Introducing segments

- Each pixel of the secret was represented with a *tile*, 2×2
- Now it will be represented with a *segment* consisting of *tiles*
- Segment is large enough to accommodate color: 3×3 tiles.

Source: (Chmielewski, Gawdzik, and Orłowski 2019).

Introducing segments

- Each pixel of the secret was represented with a *tile*, 2×2
- Now it will be represented with a *segment* consisting of *tiles*
- Segment is large enough to accommodate color: 3×3 tiles.

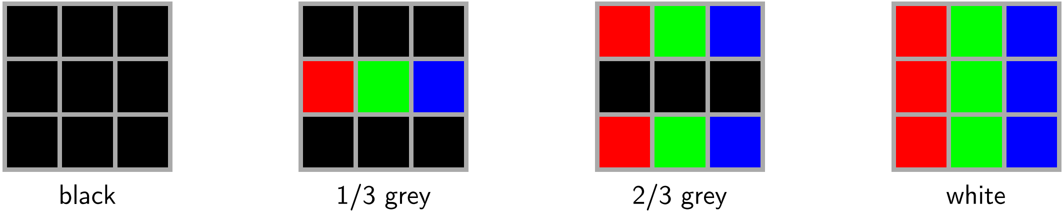


black | 1/3 grey | 2/3 grey | white

Source: (Chmielewski, Gawdzik, and Orłowski 2019).

Introducing segments

- Each pixel of the secret was represented with a *tile*, 2×2
- Now it will be represented with a *segment* consisting of *tiles*
- Segment is large enough to accommodate color: 3×3 tiles.

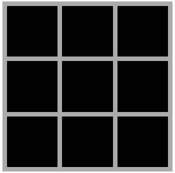


- R, G, B, C=G+B, M=R+B, Y=R+G and W=R+G+B at four levels can be represented

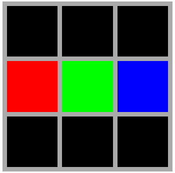
Source: (Chmielewski, Gawdzik, and Orłowski 2019).

Introducing segments

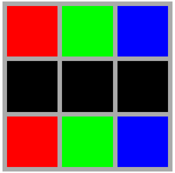
- Each pixel of the secret was represented with a *tile*, 2×2
- Now it will be represented with a *segment* consisting of *tiles*
- Segment is large enough to accommodate color: 3×3 tiles.



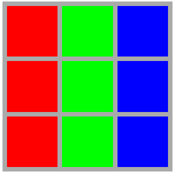
black



1/3 grey



2/3 grey



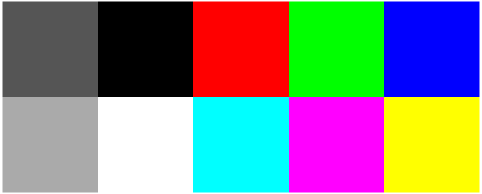
white

- R, G, B, C=G+B, M=R+B, Y=R+G and W=R+G+B at four levels can be represented
- $4^3 = 64$ color palette
- From dithering $\rightarrow$ numbers $\in \{0, 1, 2, 3\}$ of pixels necessary in R, G, B

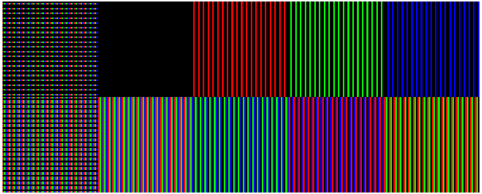
Source: (Chmielewski, Gawdzik, and Orłowski 2019).

Dithering and decomposing into color stripes

Original, full color



Dithered and decomposed



- 64-color palette is not bad, but decomposition into color stripes brings quality loss
- Decomposition is necessary – it makes it possible to represent color by R, G and B

Source: (Chmielewski, Nieniewski, and Orłowski 2021b).

Dithering and decomposing into color stripes

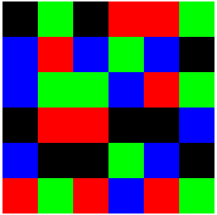
Original, full color

Dithered and decomposed

- 64-color palette is not bad, but decomposition into color stripes brings quality loss
- Decomposition is necessary – it makes it possible to represent color by R, G and B

Source: (Chmielewski, Nieniewski, and Orłowski 2021b).

A totally random color tile



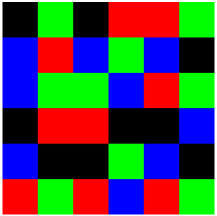
Each tile is formed of random pixels R, G, B, K.

BTW, it does not have to be 6×6 (not used here)

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

- Therefore, the first share is (pseudo)random by definition – it is formed by drawing values $\{1, 2, 3, 4\} \equiv \{R, G, B, K\}$ from a (pseudo)random number generator
- The second share is formed from the first one, but it is modified to code the secret image
- Qn: does this modification make the second share "less random"?
- Randomness in discrete sets means *no structure can be detected*
- Structure $\rightarrow$ not random; otherwise $\rightarrow$ no evidence of randomness
- Randomness cannot be *detected*; its lack can, so as many tests as possible are needed
- We shall return to this after considering the problem of errors

A totally random color tile



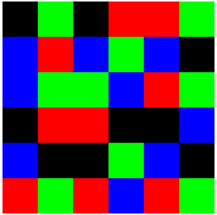
Each tile is formed of random pixels R, G, B, K.

BTW, it does not have to be 6×6 (not used here)

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

- Therefore, the first share is (pseudo)random by definition – it is formed by drawing values $\{1, 2, 3, 4\} \equiv \{R, G, B, K\}$ from a (pseudo)random number generator
- The second share is formed from the first one, but it is modified to code the secret image
- Qn: does this modification make the second share "less random"?
- Randomness in discrete sets means *no structure can be detected*
- Structure $\rightarrow$ not random; otherwise $\rightarrow$ no evidence of randomness
- Randomness cannot be *detected*; its lack can, so as many tests as possible are needed
- We shall return to this after considering the problem of errors

A totally random color tile



Each tile is formed of random pixels R, G, B, K.

BTW, it does not have to be 6×6 (not used here)

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

- Therefore, the first share is (pseudo)random by definition – it is formed by drawing values $\{1, 2, 3, 4\} \equiv \{R, G, B, K\}$ from a (pseudo)random number generator
- The second share is formed from the first one, but it is modified to code the secret image
- Qn: does this modification make the second share "less random"?
- Randomness in discrete sets means *no structure can be detected*
- Structure $\rightarrow$ not random; otherwise $\rightarrow$ no evidence of randomness
- Randomness cannot be *detected*; its lack can, so as many tests as possible are needed
- We shall return to this after considering the problem of errors

Two methods

- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Two methods

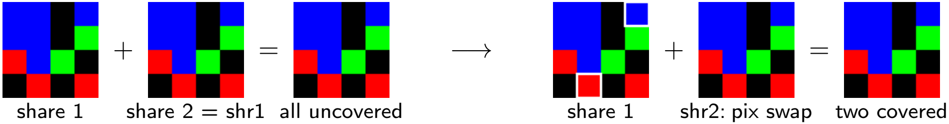
- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Two methods

- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

***Hiding* method: initially shares equal → hiding by swapping**



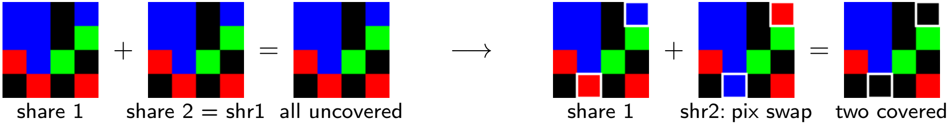
It is probable that some pixels remain unhidden

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Two methods

- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

Hiding **method: initially shares equal → hiding by swapping**



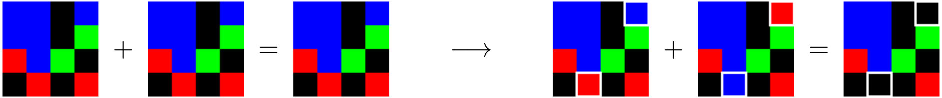
It is probable that some pixels remain unhidden

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Two methods

- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

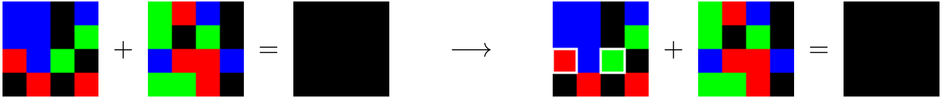
***Hiding* method: initially shares equal → hiding by swapping**



share 1 + share 2 = shr1 = all uncovered → share 1 + shr2: pix swap = two covered

It is probable that some pixels remain unhidden

***Unhiding* method: initially shares opposite (covering) → unhiding by swapping**



share 1 + opposite to shr1 = all covered → share 1 + shr2: pix swap = one uncovered

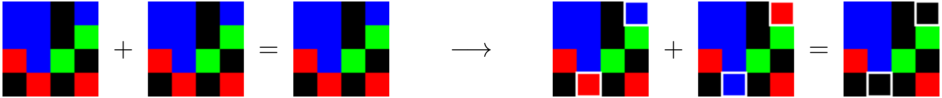
Surely no pixels can remain unhidden

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Two methods

- Now, segment 1 contains randomly drawn R, G, B and K pixels – **randomness!**
- Only operation on a segment in share 2 is swapping the pixels: **randomness** hopefully maintained

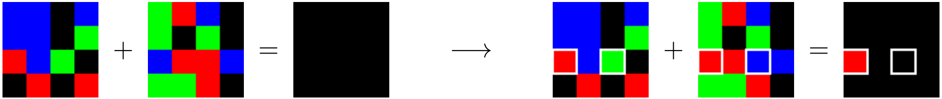
***Hiding* method: initially shares equal → hiding by swapping**



share 1 + share 2 = shr1 = all uncovered → share 1 + shr2: pix swap = two covered

It is probable that some pixels remain unhidden

***Unhiding* method: initially shares opposite (covering) → unhiding by swapping**



share 1 + opposite to shr1 = all covered → share 1 + shr2: pix swap = one uncovered

Surely no pixels can remain unhidden

Source: (Chmielewski, Nieniewski, and Orłowski 2021a; Chmielewski, Nieniewski, and Orłowski 2022b)

Algorithm – Truly random coding in one segment

Algorithm (simplified):

- **Input:** random `seg1 = seg2` (***hiding***) or `seg1` $\not\equiv$ `seg2` (***unhiding***)
  Input: numbers of R, G, B pixels `planned to be color` according to dithering
- For each segment in the image:
  - **While** planned (from dithering) number of uncovered color pixels not attained and there are pixels to choose from
    1. Choose 2 pixels `pix1` and `pix2` at random
    2. **If** `pix1` or `pix2` is `fixed` **then** goto 1
    3. **If** swapping `pix1` with `pix2` in `seg2` would `be profitable` **then**
       // condition `be profitable` means:
       // **hiding**: 1 or 2 pixels covered, none uncovered
       // **unhiding**: 1 or 2 pixels uncovered, within planned numbers
       Swap `pix1` with `pix2` in `seg2`
- **Output:** seg2 with only `planned` pixels uncovered, if possible

Comments:

- Randomness is the goal, not efficiency, hence minimum protection against repetitions
- Segments are processed independently, so parallelization would be natural

Algorithm – Truly random coding in one segment

Algorithm (simplified):

- **Input:** random `seg1 = seg2` (***hiding***) or `seg1` $\not\equiv$ `seg2` (***unhiding***)
  Input: numbers of R, G, B pixels `planned to be color` according to dithering
- For each segment in the image:
  - **While** planned (from dithering) number of uncovered color pixels not attained and
    there are pixels to choose from
    1. Choose 2 pixels `pix1` and `pix2` at random
    2. **If** `pix1` or `pix2` is `fixed` **then** goto 1
    3. **If** swapping `pix1` with `pix2` in `seg2` would `be profitable` **then**
       // condition `be profitable` means:
       // **hiding**: 1 or 2 pixels covered, none uncovered
       // **unhiding**: 1 or 2 pixels uncovered, within planned numbers
       Swap `pix1` with `pix2` in `seg2`
- **Output:** seg2 with only `planned` pixels uncovered, if possible

Comments:

- Randomness is the goal, not efficiency, hence minimum protection against repetitions
- Segments are processed independently, so parallelization would be natural

Two types of errors

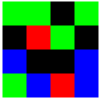
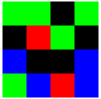
Assume the ***hiding*** method, initially equal shares, all pixels unhidden.

- Pixels in a segment are **random**: not always there are enough pixels in any color $\rightsquigarrow$ **missing color** error

Two types of errors

Assume the ***hiding*** method, initially equal shares, all pixels unhidden.

- Pixels in a segment are **random**: not always there are enough pixels in any color
  $\rightsquigarrow$ **missing color** error

We need **4 R** pixels, but random shares are  +  $\Longrightarrow$ No way, **missing color**

Two types of errors

Assume the ***hiding*** method, initially equal shares, all pixels unhidden.

- Pixels in a segment are **random**: not always there are enough pixels in any color
  $\rightsquigarrow$ **missing color** error

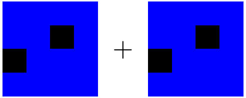
- Not always any number of pixels can be covered
  $\rightsquigarrow$ **hiding failure** error

Two types of errors

Assume the ***hiding*** method, initially equal shares, all pixels unhidden.

- Pixels in a segment are **random**: not always there are enough pixels in any color
  $\rightsquigarrow$ **missing color** error

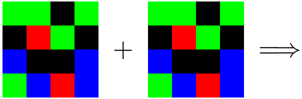
- Not always any number of pixels can be covered
  $\rightsquigarrow$ **hiding failure** error

We need **3 B** pixels, but random shares are  $\Longrightarrow$ No way, **hiding failure**

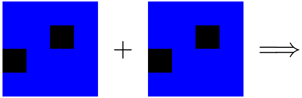
Two types of errors

Assume the ***hiding*** method, initially equal shares, all pixels unhidden.

- Pixels in a segment are **random**: not always there are enough pixels in any color
  ⇝ **missing color** error

We need **4 R** pixels, but random shares are  No way, **missing color**

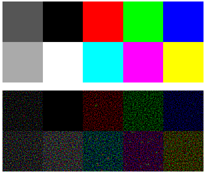
- Not always any number of pixels can be covered
  ⇝ **hiding failure** error

We need **3 B** pixels, but random shares are  No way, **hiding failure**

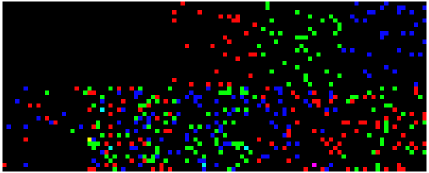
In the ***unhiding*** method, where all pixels are initially hidden, and while swapping the unhiding of unwanted color is not allowed, the hiding failure error cannot appear

Hiding method: Two types of errors

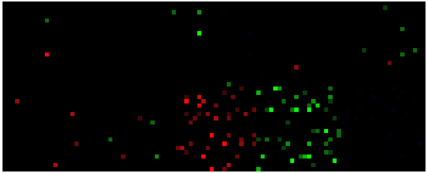
1/2: `test100`



decoded



missing color errors

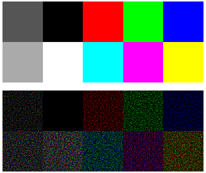


hiding failure errors

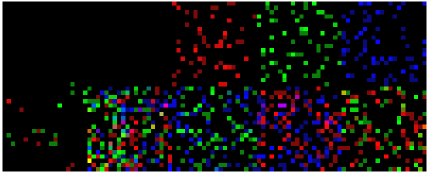
- Note the relation of error density to color: more white more errors
- Number of *hiding failures* is generally smaller than that of *missing colors*

Unhiding method: One type of errors

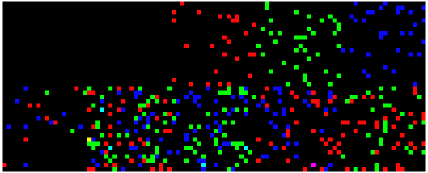
1/2: `test100`



decoded



missing color errors

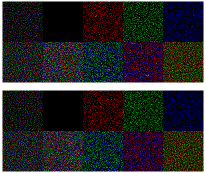


For comparison: *missing color* from *hiding*

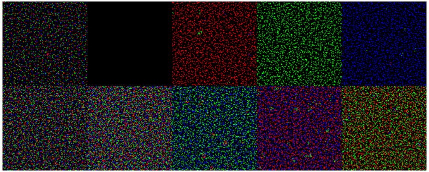
- Note the relation of error density to color: more white more errors
- Number of *hiding failures* is generally larger than that in the *hiding* method

Two methods: restored images

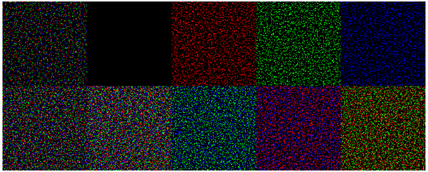
1/2: `test100`



decoded: ↑ *bas* ↓ *no hid*



decoded from *hiding*

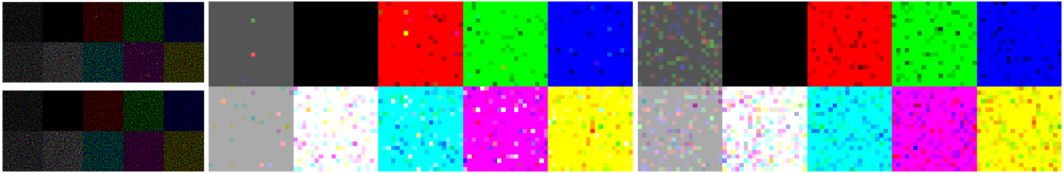


decoded from *unhiding*

- In the *hiding* method pixels can be too bright, there are spikes
- In the *unhiding* method pixels can be too dark, there is granularity

Two methods: restored images

1/2: `test100`



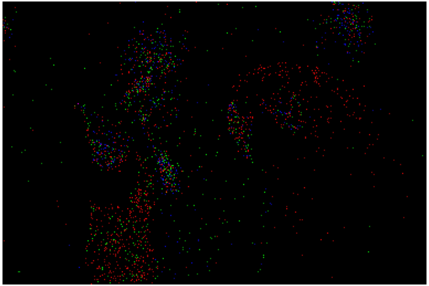
decoded: $\uparrow$ *bas* $\downarrow$ *no hid* restored from *hiding* restored from *unhiding*

- In the *hiding* method pixels can be too bright, there are spikes
- In the *unhiding* method pixels can be too dark, there is granularity

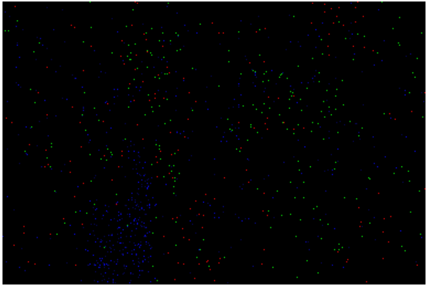
Hidig method: Two types of errors

2/2: `parrots`

decoded



missing color errors



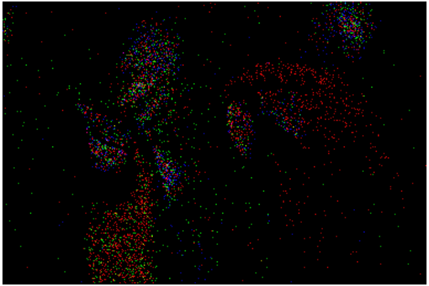
hiding failure errors

- Note the relation of error density to color: more white more errors
- Number of *hiding failures* is generally smaller than that of *missing colors*

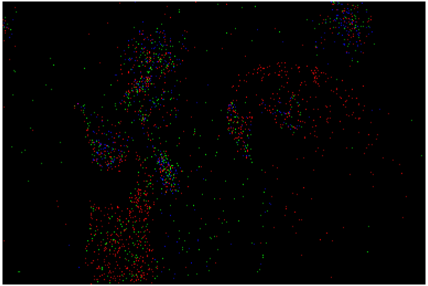
Unhiding method: One type of errors

2/2: parrots

decoded



missing color errors



For comparison: *missing color* from *hiding*

- Note the relation of error density to color: more white more errors
- Number of *hiding failures* is generally larger than that in the *hiding* method

Two methods: restored images

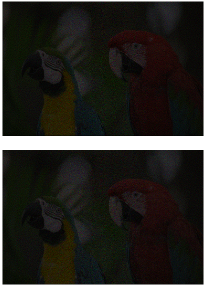
2/2: `parrots`

decoded: ↑ *bas* ↓ *no hid* | decoded from *hiding* | decoded from *unhiding*

- In *hiding* method there are spikes
- In *unhiding* there is granularity

Two methods: restored images

2/2: `parrots`



decoded: $\uparrow$ *bas* $\downarrow$ *no hid*

restored from *hiding*

restored from *unhiding*

- In *hiding* method there are spikes
- In *unhiding* there is granularity

Testing the randomness

Simulation of a series of experiments

- We used the NIST randomness tests (Bassham, Rukhin, Soto, Nechvatal, Smid, Leigh, Levenson, Vangel, Heckert, and Banks 2010): a battery of 15 advanced tests, some with many subtests (188 tests together).
- Tests were performed for six known test images (`parrots`, `peppers`, `Lena`, ...). For each, 100 realizations of coding were simulated.
  - both shares were analyzed $\longrightarrow$ 2 cases,
  - pixels were read by rows and by columns $\longrightarrow \times 2$ cases $= 4$ cases per image,
  - pixels in R, G, B, K were represented by 00, 01, 10, 11 (arbitrary choice).
- p-values for each of 4×100 realizations $\times 188$ tests, for benchmark images, were recorded
- These 75 200 data per image were presented in a compact form, in one page of graphs (Chmielewski, Nieniewski, and Orłowski 2022a)
- The $6 \times 188 \times 4 = 4512$ histograms (6 images, 188 tests, 2 shares, 2 directions) were tested for representing a random process
- Qn: Did the modification of share 2 introduced loss of randomness?

Source: (Chmielewski, Nieniewski, and Orłowski 2021b), probably the first extensive randomness tests.

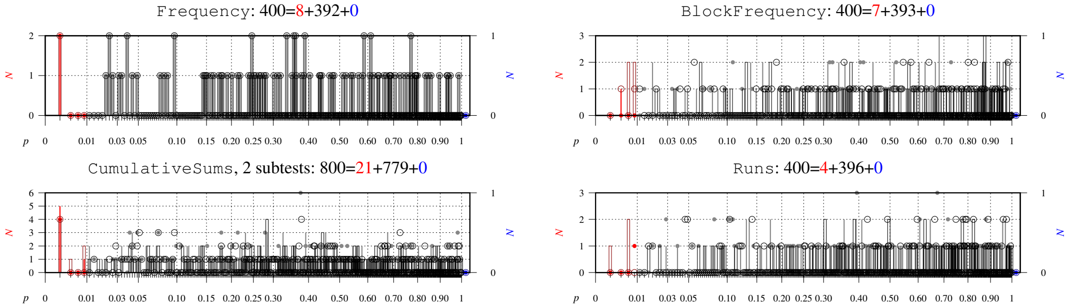
Statistical evidence for randomness

- In any test, the hypothesis of randomness can be rejected.
  But randomness cannot be proven.
  Hence, the more tests the better, but never too many.
- We have already used the results of NIST randomness tests to analyze *p*-values and to show graphical evidence of randomness (Chmielewski, Nieniewski, and Orłowski 2022a). Now we shall test it statistically.
- We analyze these 4512 sets of *p*-value vectors, 100 elements each, in two ways:
  - by counting how many fall below a fixed rejection threshold ($\alpha = 0.01$),
  - by testing their histogram for deviations from uniformity using the Kolmogorov–Smirnov and chi-squared tests. This second-order statistical analysis enables a more rigorous evaluation of the method's ability to preserve randomness.
- **Shortly: a sequence of bits is random, if the histogram of *p*-values in a test is flat.**
  We shall spare you the trouble of looking at all the histograms.

Statistical evidence for randomness

- In any test, the hypothesis of randomness can be rejected.
  But randomness cannot be proven.
  Hence, the more tests the better, but never too many.
- We have already used the results of NIST randomness tests to analyze p-values and to show graphical evidence of randomness (Chmielewski, Nieniewski, and Orłowski 2022a). Now we shall test it statistically.
- We analyze these 4512 sets of p-value vectors, 100 elements each, in two ways:
  - by counting how many fall below a fixed rejection threshold ($\alpha = 0.01$),
  - by testing their histogram for deviations from uniformity using the Kolmogorov–Smirnov and chi-squared tests. This second-order statistical analysis enables a more rigorous evaluation of the method's ability to preserve randomness.
- **Shortly: a sequence of bits is random, if the histogram of p-values in a test is flat.**
  We shall spare you the trouble of looking at all the histograms.

Histograms of p-values for parrots



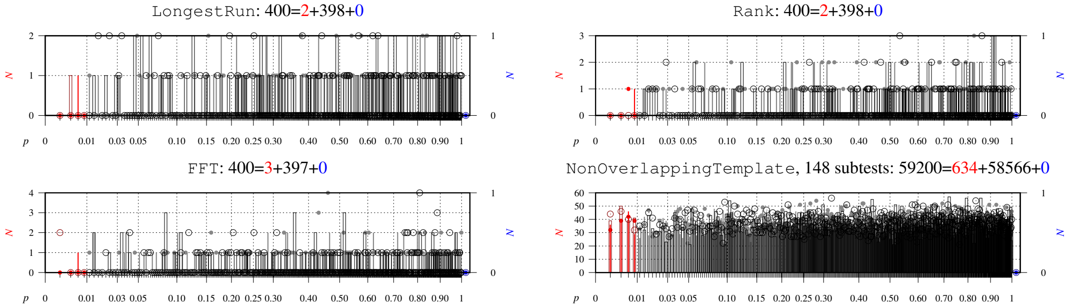
1/4

- Histograms are flat and exhibit expected numbers of failures ($p \leq \alpha, \alpha = 0.01$)
- No reason to reject the hypothesis of randomness, so success

key for graphs

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

Histograms of *p*-values for parrots



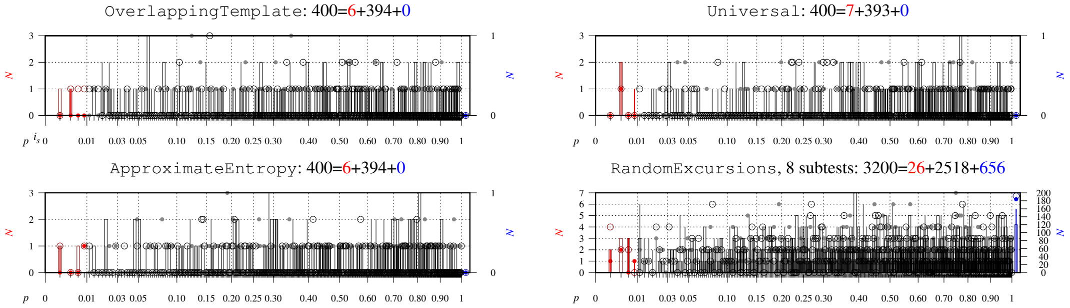
2/4

- Histograms are flat and exhibit expected numbers of failures ($p \leq \alpha, \alpha = 0.01$)
- No reason to reject the hypothesis of randomness, so success

key for graphs

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

Histograms of p-values for parrots



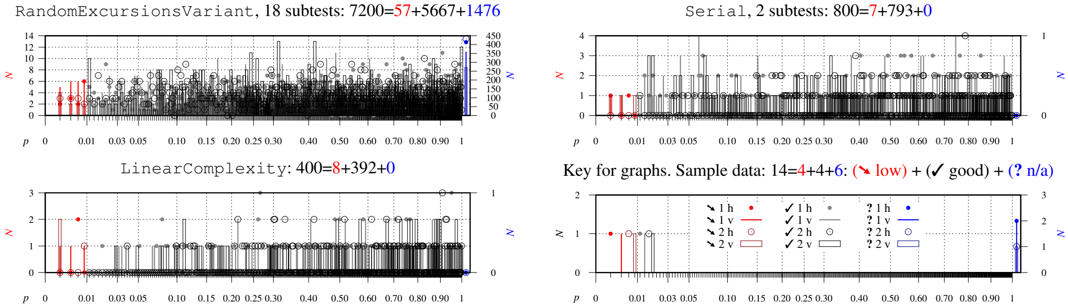
3/4

- Histograms are flat and exhibit expected numbers of failures ($p \leq \alpha, \alpha = 0.01$)
- No reason to reject the hypothesis of randomness, so success

▸ key for graphs

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

Histograms of *p*-values for parrots

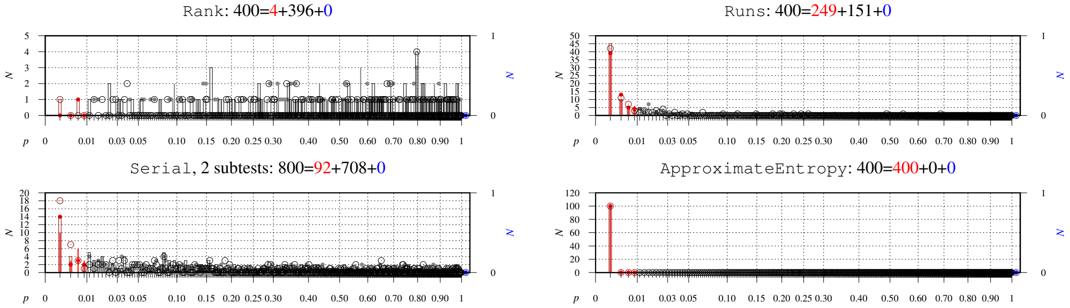


4/4

- Histograms are flat and exhibit expected numbers of failures ($p \leq \alpha, \alpha = 0.01$)
- No reason to reject the hypothesis of randomness, so success

Source: (Chmielewski, Nieniewski, and Orłowski 2021a).

Negative examples for one of previous methods, for parrots



To see ↑ how the negative evidence looks like

- Some histograms are flat, but some are clearly not; too many failures (red)
- There are reasons to reject the hypothesis of randomness, so failure

key for graphs

Source: (Chmielewski, Nieniewski, and Orłowski 2021c).

By test

Table: Rejections by Test

Test	Share 1	Share 2	Rows	Cols	Fails/Samps
Hiding method					
p-v	43	36	45	34	79/4512
K-S	16	29	16	29	46/4512
χ^2	19	27	23	23	46/4512
Unhiding method					
p-v	49	51	49	51	100/4512
K-S	16	20	21	15	36/4512
χ^2	24	25	28	21	49/4512

p-v – counting small *p*-values; K-S – Kolmogorov–Smirnov test; χ^2 – chi-squared test

By share/direction

Table: Rejections by Shares/Directions

Case	Counts/4512 Samples		
Hiding method			
By shares	Share 1: 69	Share 2: 73	Both: 8
By directions	Rows: 74	Columns: 68	Both: 6
Unhiding method			
By shares	Share 1: 77	Share 2: 84	Both: 7
By directions	Rows: 83	Columns: 78	Both: 9

See the paper for more comparisons.

Discussion

- Numbers of rejections were small with respect to the number of samples – close to 1%.
- Rejections in the modified share 2 were not evidently larger, and in some cases smaller than those in the readily generated share 1.
- The cases of rejecting the randomness by all three test or pairs of tests were not a rule.
- Evidence for non-randomness of color distributions in both shares is similarly small.
- NIST tests detected rejections of randomness with very different frequency.
  - `OverlappingTemplate` detected over ten rejections in both methods – *hiding* and *unhiding*.
  - A number of tests detected from 1 to 3 rejections.
  - A number of tests appeared not to detect any lack of randomness in any sample, in at least one method: `ApproximateEntropy`, `CumulativeSums 2`, `LinearComplexity`, `Rank`, `Runs`, `Universal`, and some subtests of `NonOverlappingTemplate`, `RandomExcursions` and `RandomExcursionsVariant`.

Discussion

- Numbers of rejections were small with respect to the number of samples – close to 1%.
- Rejections in the modified share 2 were not evidently larger, and in some cases smaller than those in the readily generated share 1.
- The cases of rejecting the randomness by all three test or pairs of tests were not a rule.
- Evidence for non-randomness of color distributions in both shares is similarly small.
- NIST tests detected rejections of randomness with very different frequency.
  - `OverlappingTemplate` detected over ten rejections in both methods – *hiding* and *unhiding*.
  - A number of tests detected from 1 to 3 rejections.
  - A number of tests appeared not to detect any lack of randomness in any sample, in at least one method: `ApproximateEntropy`, `CumulativeSums 2`, `LinearComplexity`, `Rank`, `Runs`, `Universal`, and some subtests of `NonOverlappingTemplate`, `RandomExcursions` and `RandomExcursionsVariant`.

Discussion

- Numbers of rejections were small with respect to the number of samples – close to 1%.
- Rejections in the modified share 2 were not evidently larger, and in some cases smaller than those in the readily generated share 1.
- The cases of rejecting the randomness by all three test or pairs of tests were not a rule.
- Evidence for non-randomness of color distributions in both shares is similarly small.
- NIST tests detected rejections of randomness with very different frequency.
  - `OverlappingTemplate` detected over ten rejections in both methods – *hiding* and *unhiding*.
  - A number of tests detected from 1 to 3 rejections.
  - A number of tests appeared not to detect any lack of randomness in any sample, in at least one method: `ApproximateEntropy`, `CumulativeSums 2`, `LinearComplexity`, `Rank`, `Runs`, `Universal`, and some subtests of `NonOverlappingTemplate`, `RandomExcursions` and `RandomExcursionsVariant`.

Conclusion

- We analyzed whether the generation of the second share in visual cryptography compromises its statistical randomness.
- We found no evidence of such degradation.
- Standard randomness tests and second-order evaluations of p-value distributions were used.
- Our method for constructing the second share produces outputs that remain statistically indistinguishable from noise.
- This confirms its suitability for secure visual encryption.
- This also demonstrates that controlled determinism can coexist with apparent randomness.
- More broadly, this shows that meta-analysis of test outputs provides a powerful tool for validating the integrity of cryptographic structures under transformation.

Thank you

fav. sl.

Conclusion

- We analyzed whether the generation of the second share in visual cryptography compromises its statistical randomness.
- We found no evidence of such degradation.
- Standard randomness tests and second-order evaluations of p-value distributions were used.

- Our method for constructing the second share produces outputs that remain statistically indistinguishable from noise.
- This confirms its suitability for secure visual encryption.
- This also demonstrates that controlled determinism can coexist with apparent randomness.

- More broadly, this shows that meta-analysis of test outputs provides a powerful tool for validating the integrity of cryptographic structures under transformation.

Thank you

fav. sl.

Conclusion

- We analyzed whether the generation of the second share in visual cryptography compromises its statistical randomness.
- We found no evidence of such degradation.
- Standard randomness tests and second-order evaluations of p-value distributions were used.

- Our method for constructing the second share produces outputs that remain statistically indistinguishable from noise.
- This confirms its suitability for secure visual encryption.
- This also demonstrates that controlled determinism can coexist with apparent randomness.

- More broadly, this shows that meta-analysis of test outputs provides a powerful tool for validating the integrity of cryptographic structures under transformation.

Thank you

fav. sl.

References I

Ateniese, G., C. Blundo, A. D. Santis, and D. R. Stinson (2001).
Extended capabilities for visual cryptography.
Theoretical Computer Science 250(1), 143–161.

Bassham, L. E., A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks (2010, 16 Sep).
A statistical test suite for random and pseudorandom number generators for cryptographic applications.
Technical report, National Institute of Standards and Technology, Gaithersburg, MD, USA.
Series: Special Publication (NIST SP), Rep. No. 800-22 Rev 1a.

Chmielewski, L. J., G. Gawdzik, and A. Orłowski (2019, 16-18 Oct).
Towards color visual cryptography with completely random shares.
In *Proc. Conf. PP-RAI 2019 – Konf. Polskiego Porozumienia na rzecz Rozwoju Sztucznej Inteligencji*, Wrocław, Poland, pp. 150–155. Wrocław University of Science and Technology.

Chmielewski, L. J., M. Nieniewski, and A. Orłowski (2021a, 28-30 Jun).
Can color visual cryptography be truly random?
In M. Choraś, R. S. Choraś, M. Kurzyński, et al. (Eds.), *Progress in Image Processing, Pattern Recognition and Communication Systems – Proc. Int. Conf. CORES, IP&C, ACS 2021*, Volume 255 of *LNNS*, Bydgoszcz, Poland, pp. 72–86. Springer, 2022.

References II

Chmielewski, L. J., M. Nieniewski, and A. Orłowski (2021b).
Testing the randomness of shares in color visual cryptography.
Pattern Analysis & Applications 24(4), 1475–1487.
doi:10.1007/s10044-021-00999-5.

Chmielewski, L. J., M. Nieniewski, and A. Orłowski (2021c).
Testing the randomness of shares in color visual cryptography.
Pattern Analysis & Applications 24(4), 1475–1487.

Chmielewski, L. J., M. Nieniewski, and A. Orłowski (2022a, 19-21 Sep).
Error analysis and graphical evidence of randomness in two methods of color visual cryptography.
In L. J. Chmielewski and A. Orłowski (Eds.), *Computer Vision and Graphics: Proc. ICCVG 2022*, Volume 598 of *LNNS*, Warsaw, Poland, pp. 237–267. Springer, Cham, 2023.

Chmielewski, L. J., M. Nieniewski, and A. Orłowski (2022b, 25-27 Apr).
Truly random color visual cryptography without surplus color spikes.
In P. Jędrzejowicz and I. Czarnowski (Eds.), *Proc. 3rd Polish Conference on Artificial Intelligence PP-RAI'2022*, Gdynia, Poland, pp. 53–56. Publishing House of Gdynia Maritime University.

Dhiman, K. and S. S. Kasana (2018).
Extended visual cryptography techniques for true color images.
Computers & Electrical Engineering 70, 647–658.

References III

Naor, M. and A. Shamir (1995, 9-12 May).
Visual cryptography.
In A. De Santis (Ed.), *Advances in Cryptology — EUROCRYPT'94. Proc. Workshop on the Theory and Application of Cryptographic Techniques*, Volume 950 of *LNCS*, Perugia, Italy, pp. 1–12. Springer.

Naor, M. and A. Shamir (1997, 10-12 Apr).
Visual cryptography II: Improving the contrast via the cover base.
In M. Lomas (Ed.), *Security Protocols. Proc. Int. Workshop on Security Protocols*, Volume 1189 of *LNCS*, Cambridge, United Kingdom, pp. 197–202. Springer.

Orłowski, A. and L. J. Chmielewski (2019a, 7-9 Jan).
Generalized visual cryptography scheme with completely random shares.
In N. Petkov, N. Strisciuglio, and C. M. Travieso (Eds.), *Proc. 2nd Int. Conf. Applications of Intelligent Systems APPIS 2019*, Las Palmas de Gran Canaria, Spain, pp. 33:1–33:6. Association for Computing Machinery.

Orłowski, A. and L. J. Chmielewski (2019b, 7-9 Jan).
Generalized visual cryptography scheme with completely random shares.
In N. Petkov, N. Strisciuglio, and C. M. Travieso (Eds.), *Proc. 2nd Int. Conf. Applications of Intelligent Systems APPIS 2019*, Las Palmas de Gran Canaria, Spain, pp. 33:1–33:6. Association for Computing Machinery.

References IV

Orłowski, A. and L. J. Chmielewski (2019c, 16-20 Jun).
Randomness of shares versus quality of secret reconstruction in black-and-white visual cryptography.
In L. Rutkowski et al. (Eds.), *Proc. Int. Conf. on Artificial Intelligence and Soft Computing ICAISC 2019*, Volume 11509 of *LNAI*, Zakopane, Poland, pp. 58–69. Springer.

Wikipedia contributors (2021).
Dither — Wikipedia, the free encyclopedia.
url:https://en.wikipedia.org/w/index.php?title=Dither&oldid=1028207456 [Accessed: 25 Jun 2021].

Key for histograms of *p*-values

Key for graphs. Sample data: 14=4+4+6: (↘ low) + (✓ good) + (? n/a)