

Ornament and crime

Description

default watermark

The Austrian architect Adolf Loos (1870-1933) wrote *Ornament and Crime*. It is a celebrated polemic against the superfluity of ornament in the modern industrial age: â??not only is ornament produced by criminals but also a crime is committed through the fact that ornament inflicts serious injury on peopleâ??s health, on the national budget and hence on cultural evolutionâ?• (21).

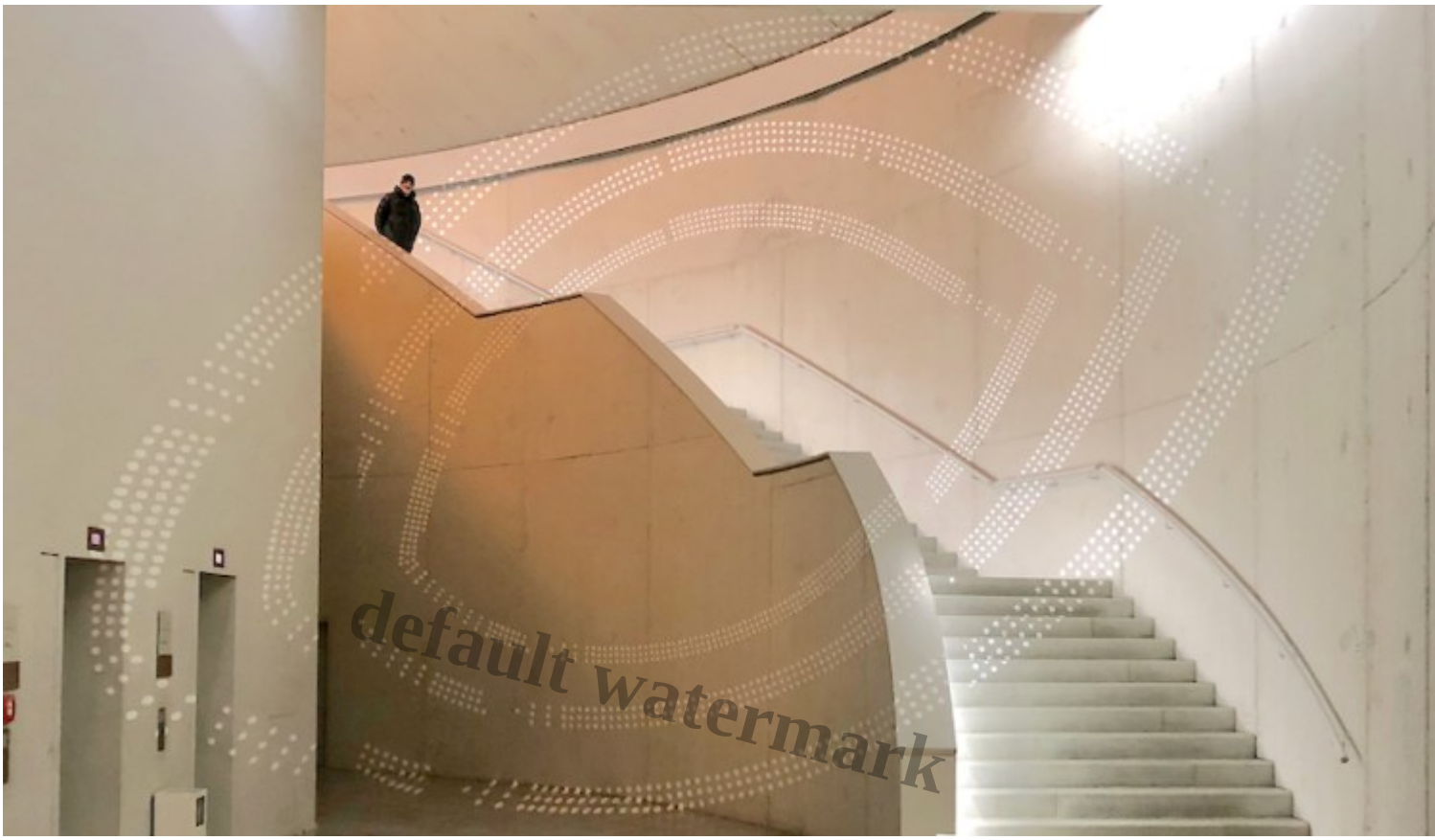
The cover of the Penguin edition shows a grey, lightly textured hand painted rectangle. It could be by Kazimir Malevich, but I canâ??t confirm until I get hold of a paper copy of the edition.

The least significant bits

The superfluity of ornament is a common theme in art, design and architecture. When it comes to structure, form, substance, and function the ornamentation is the least significant *bit*. Itâ??s the part that can be adjusted or even removed without jeopardising the integrity of the whole. Yet it is capable of delivering substantial significance on behalf of the structure.

Iâ??m thinking of traditional stone masonry, where the masons were given reign to shape and adorn those minor parts of the building. Brackets, bosses, capstones, and rainwater spouts could be carved as flowers, crests, tools, portraits, animals and grotesques. We might say these adornments introduced a layer of coded, and even secret, meanings. Thatâ??s about as far as I dare take the analogy with what follows here.

Hereâ??s a photograph taken inside the concert hall in Luxembourg Philharmonic Hall by architect Christian de Portzamparc. The eye shape is not a watermark on the photograph, but a pattern painted onto the walls, floor and stairs.



Least significant bit

Computer files are made up of bit strings. A byte is 8 bits, and in a text file a byte represents a number that in turn converts to a text symbol. The Unicode standard provides a set of unique numbers for 143,696 characters in many languages plus 163 format symbols. The first 128 numbers cover the basic integers, alphabetic characters and format characters such as line breaks and spaces (called the ASCII set). Hereâ??s the words â??Ornamentationâ?• in 8 bit binary. The spaces arenâ??t necessary, but make the 8 bit units (bytes) clearer. Thereâ??s a website (RapidTables) that provides this conversion. I also worked it out with an Excel spreadsheet.

> 01001111 01110010 01101110 01100001 01101101 01100101 01101110 01110100 01100001 01101100 00100000 00100000

If you change just one bit in a bit string then that can render the original unintelligible. Here I changed the last bit in each 8 bit string to its opposite, i.e. changed 1s to 0s and 0s to 1s to produce â??Nso`ldou`m!!â?•. The result is just as meaningless if I change the first bit of each bit string, producing â??Å?Ã?Ã?Â·Ã?Ã?Ã?Â·Ã•â? â? â?•. Text characters are highly intolerant to noise, i.e. random adjustments to individual bits.

The same applies to numbers as characters. The Unicode 8-bit string for the character â??9â?• is 00111001. Converted to decimals, thatâ??s 57. â??9â?• is the 57th character in the Unicode table. Changing the last bit in that bit string produces the character â??8â?•. So that change can result in a significant error in a sentence like: â??In an emergency call 999.â?•

Bit strings are numbers. A graphic data file will consist of bit strings that indicate pixel colour values as numbers, not characters. Here, the order of the bit string is important. Changes to bits on the left side alter the number significantly. Those on the right less so. Left bits are therefore more significant than those on the right.

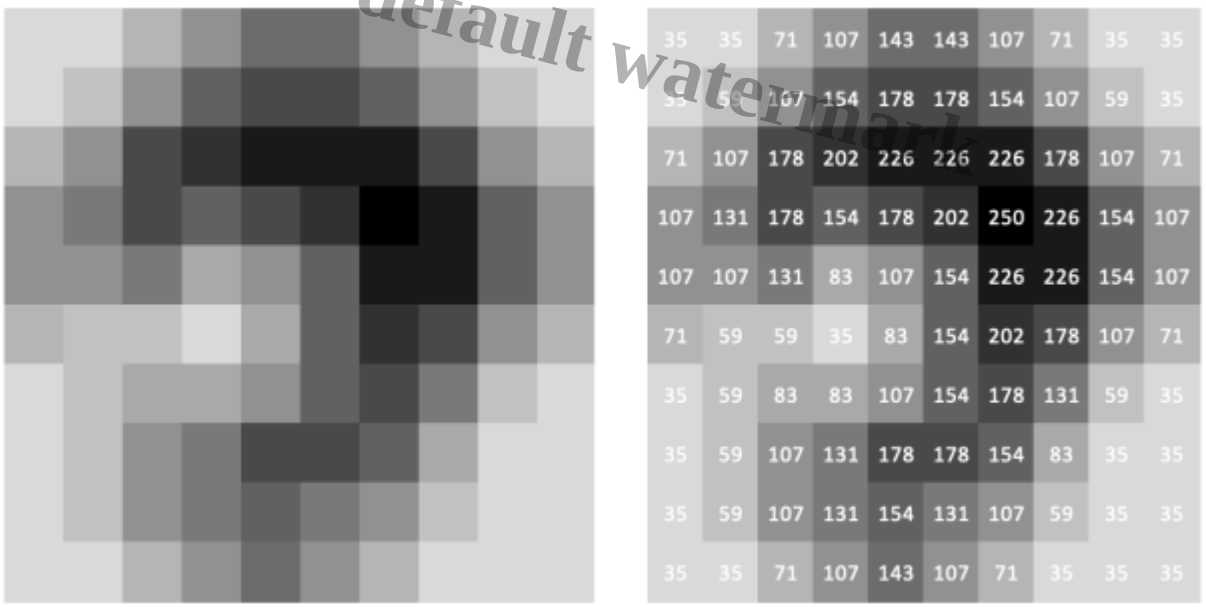
1111111 = 255
1111111 = 254
01111111 = 127

If these are the values of a pixel in a grey scale (or colour) image, then a discrepancy in the last bit on the right will go unnoticed when the pixels are drawn on the screen.

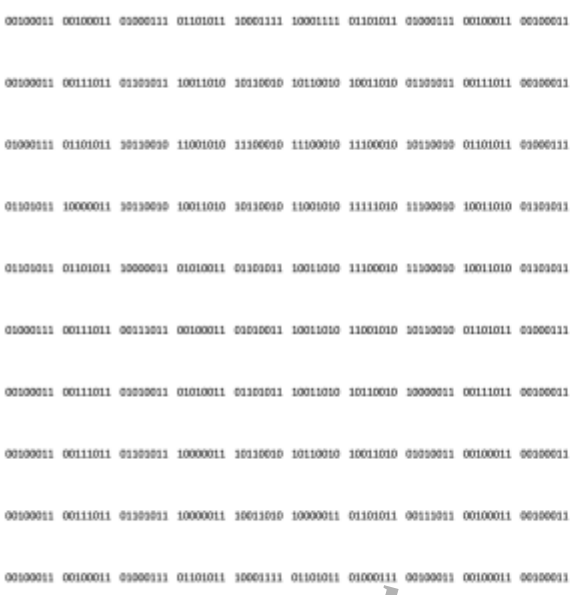
Bit-crime 101

Hereâ??s how I surreptitiously inserted the word â??Ornamentalâ?• in binary in a greyish rectangle. Hereâ??s a fragment of a grey scale image. The grey scale colour value 1-255 appears in each cell.

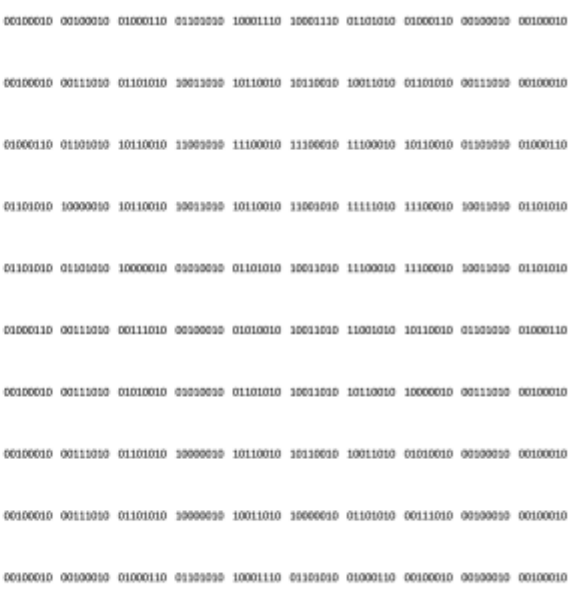
default watermark



Here are the pixel values in binary (on the left). The right version shows the same values with the right most bit set to 0. That slight change produces an imperceptible change to the colours in the image.



Carrier array; numbers 1-255 in binary



Carrier array with last bit set to 0

default watermark

Here are the binary bits of the message â??Ornamentalâ?• distributed across the same 10Ã?10 array.

max 12 character message

Ornamental

01001111011100100110111001100001011011010110010101101
11001110100011000010110110000100000000100000

0	1	0	0	1	1	1	1	0	1
1	1	0	0	1	0	0	1	1	0
1	1	1	0	0	1	1	0	0	0
0	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	0	1	0	1
1	0	1	1	1	0	0	1	1	1
0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

Each bit of binary coding of message allocation to an array cell

Next, these values are added to the grey image binary values. (Click to zoom in.) This is a the data that you would transmit as an image file. It contains the secret message.

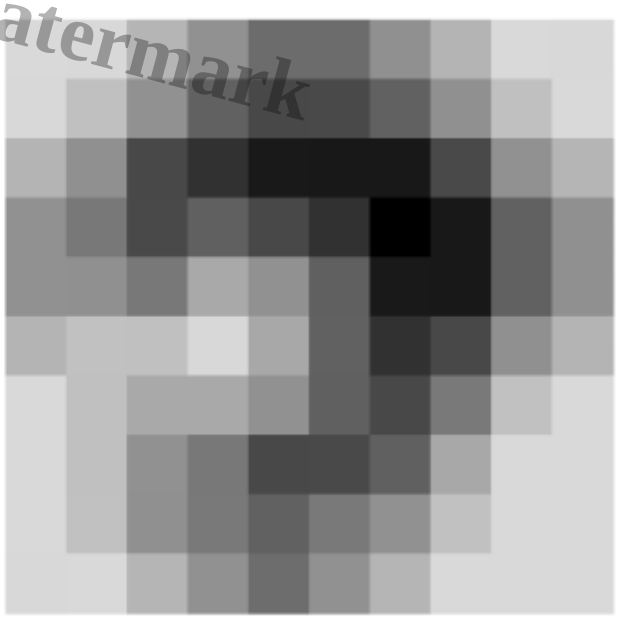
00100010	00100011	01000110	01101010	10001111	10001111	01101011	01000111	00100010	00100011
00100011	00111011	01101010	10011010	10110011	10110010	10011010	01101011	00111011	00100010
01000111	01101011	10110011	11001010	11100010	11100011	11100011	10110010	01101010	01000110
01101010	10000011	10110010	10011011	10110011	11001010	11111011	11100011	10011010	01101011
01101010	01101011	10000011	01010010	01101010	10011011	11100010	11100011	10011010	01101011
01000111	00111010	00113011	00100011	01010011	10011010	11001010	10110011	01101011	01000111
00100010	00111011	01010010	01010010	01101010	10011011	10110011	10000010	00111010	00100010
00100010	00111011	01100010	10000011	10110011	10110010	10011011	01010011	00100010	00100010
00100010	00111010	01100011	10000010	10011010	10000010	01101010	00111010	00100010	00100010
00100011	00100010	01000110	01101010	10001110	01101010	01000110	00100010	00100010	00100010

Binary values of message added to carrier array. This is what gets transmitted

default watermark

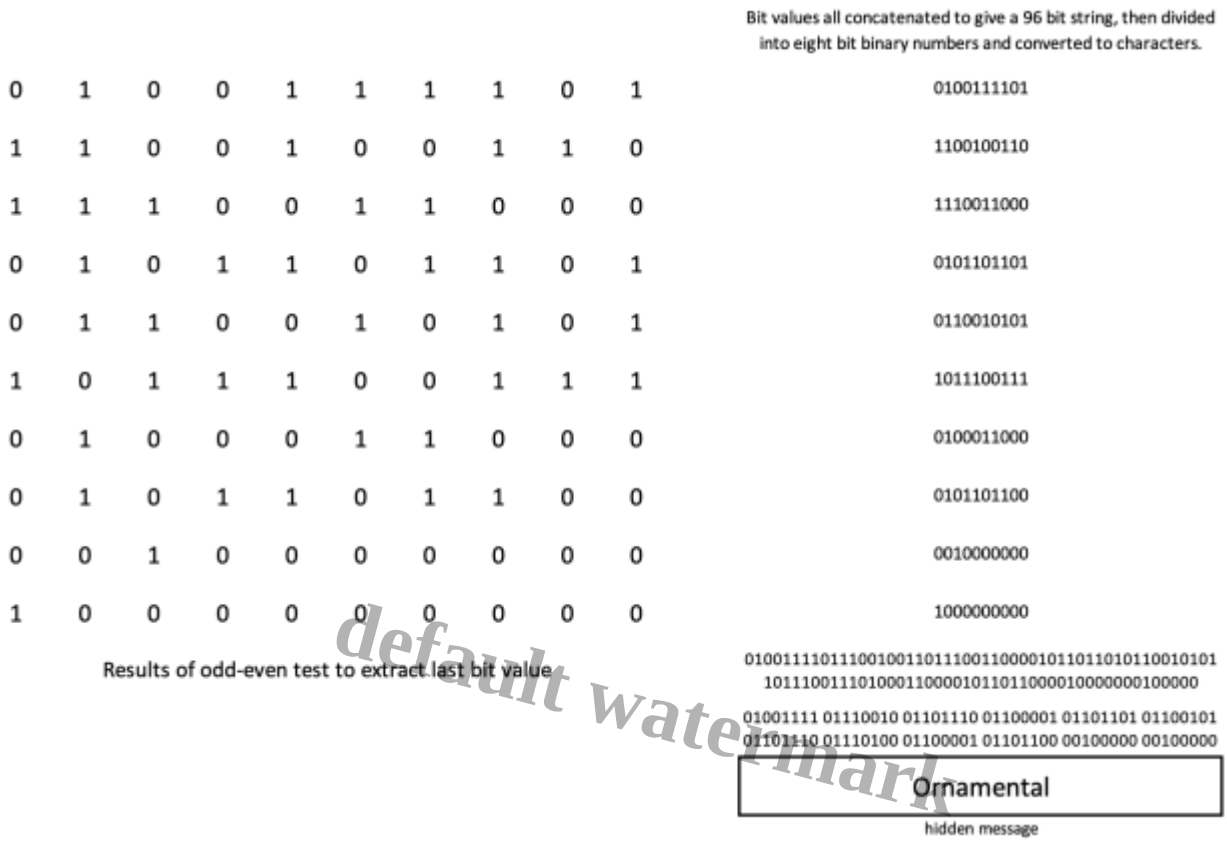
34	35	70	106	143	143	107	71	34	35
35	59	106	154	179	178	154	107	59	34
71	107	179	202	226	227	227	178	106	70
106	131	178	155	179	202	251	227	154	107
106	107	131	82	106	155	226	227	154	107
71	58	59	35	83	154	202	179	107	71
34	59	82	82	106	155	179	130	58	34
34	59	106	131	179	178	155	83	34	34
34	58	107	130	154	130	106	58	34	34
35	34	70	106	142	106	70	34	34	34

Integer values of message added to carrier array.



This is how the array with the hidden message looks on screen.

If a binary string ends in 0 it is even; if it ends in 1 then itâ??s odd. The test for that is easy to automate in a spreadsheet, and that simple test will recover the hidden binary sequence. To proved the message is in there I reversed the process to extract the original message.



default watermark

Thatâ??s one way that a steganographic procedure can conceal a text message in an image. The word â??Ornamentalâ?• is innocuous, but not if the hidden word is a password to a bank account, a decryption key for a file, a watermark, or a trigger to activate some malware.

The hidden message could be any arbitrary string of code, including a hash code. Here the carrier image is just 10Ã?10 pixels concealing 12 characters, but by my reckoning, a raw picture file of 3000Ã?3000 pixels with 0-255 shades of RGB could hide a file of over 3 million characters, i.e. 3 books â?? less if you allow for error checking and compression.

References

- Anon. 2020. Binary to text translator. *RapidTables*. Available online: https://www.rapidtables.com/convert/number/binary-to-ascii.html (accessed 20 June 2020).
- Loos, Adolf. 1908. Ornament and Crime (PDF). *Innsbruck, reprint Vienna, 1930*, Archived from the original on 2015-04-03. Available online: https://web.archive.org/web/20150403175309/http://www2.gwu.edu/~art/Temporary_SL/177/pdfs/Lo (accessed 20 June 2020).
- Loos, Adolf. 2019. *Ornament and Crime*. London: Penguin

Category

1. Architecture

Tags

1. encryption
2. LSB
3. steganography

Date Created
July 11, 2020
Author
rcoyne99

default watermark