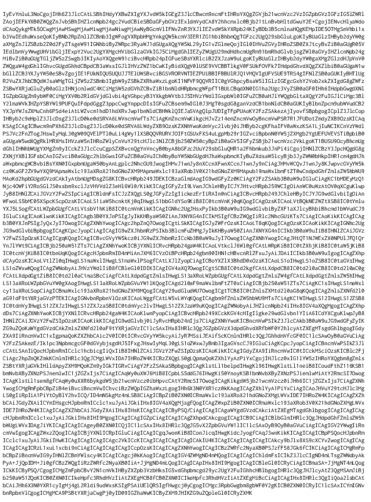
How to store data on paper?

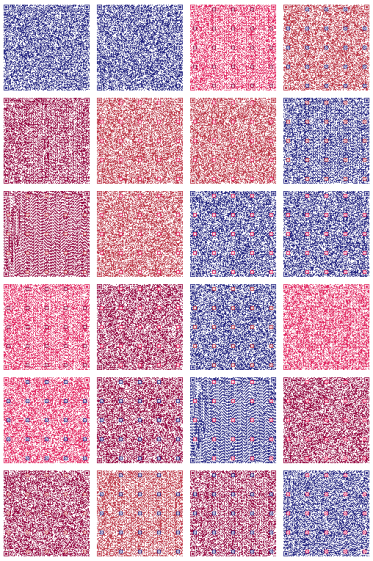
by Martin Monperrus

How to print my voice? How to put music on paper? How to print an executable program? All this boils down to storing digital data on paper. This is also called paper data storage. Arrived in this domain from poetry, I've been investigating the area for some time. Here is what I found.

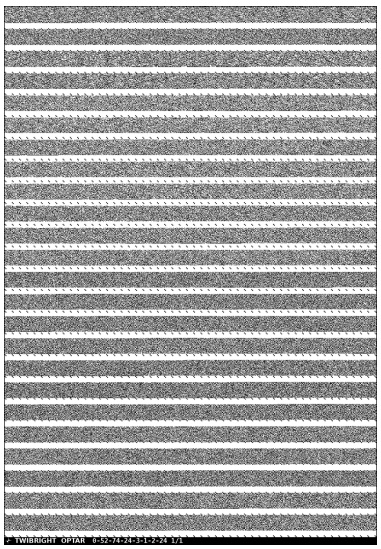
What to print on paper?



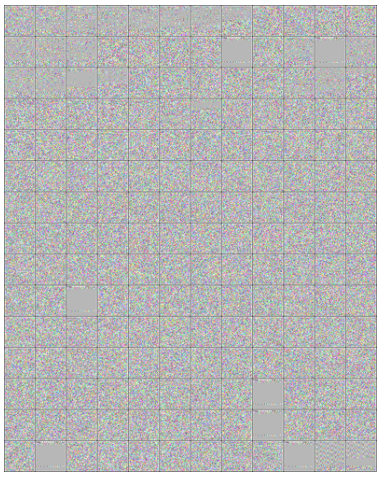
Printing an executable program (encoded in base64)



Printing a secret message (encrypted with GPG-AES256, encoded as stacked qrcode).



Printing a 21 seconds soundscape of a rugby match (in 64kpbs MP3 encoded with Optar)



Printing a 7-page scientific paper on a single page (a PDF file encoded with colorsafe)

Maximum information capacity on a single A4 page

How to transform digital data to a printable format? You need a format that transforms a sequence of bytes into an image with a specific encoding. Then, you need two pieces of software, one encoder (data->image) and one decoder (image->decoder).

In this context, information capacity refers to the number of bytes one can put per a certain area. In this document, I focus on the maximum number of kilobytes per A4 page.

Character-based encodings

TL;DR; OCR digital data is a very hard problem, one can get 2.5 kilobytes per A4 page that is decodable afterwards, see "How to get perfect OCR for digital data?".

With character-based encoding, the idea is to use a binary-to-text encoding and then to use optical character recognition (OCR). The biggest advantage of character-based encodings is that they can be decoded by humans (as opposed to dot-based encodings), which means that you don't need a camera or a scanner to recover the data. This is the most robust assumption for extreme conditions.

The information capacity primarily depends on the size of the alphabet and the font size (plus some variations due to the font being used). Yet, those numbers are theoretical in the sense that they assume perfect OCR.

base16/hexadecimal, I can get 3.5 kilobytes per A4 page (font size 12pt, font Inconsolata) and OCR it with gocr at 400DPI. With smaller fonts, we have for example 4.6 kilobytes per A4 page (font size 9pt, font Inconsolata), but I cannot OCR it well (confusion between "7" and "1"), see "How to get perfect OCR for digital data?".

Printing an encrypted GPG file encoded in base16

base32, I can get 5.8 kilobytes per A4 page (font size 9pt, font Inconsolata), but it is impossible for me to OCR it later without errors, see "How to get perfect OCR for digital data?".

base64: 9.3 kilobytes per A4 page at font size 8t, even 17 kilobytes per A4 page at font size 6pt (confirming this post, but this is theoretical, no OCR can handle it).

Printing an executable program encoded in base64

bip39, I can get 1kb per A4 page (font size 12pt, font Inconsolata), and decode it, see "How to get perfect OCR for digital data?".

Printing binary data encoded in bip39

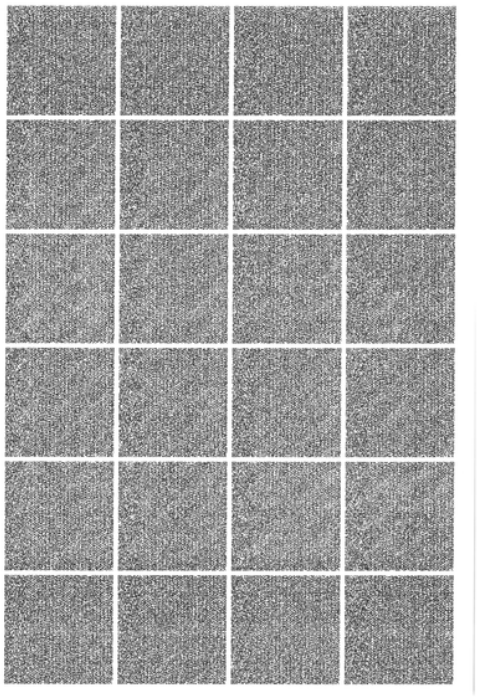
Black-and-white Dot Encodings

TL;DR; Up to 70k-100K per A4 page. Stacked QRCodes work well with off-the-shelf software. Optar is OK if one decreases the default density (XCROSSES/XCROSSES).

In a dot-based based encoding, the information capacity is a function of how small the dots are. In other words, it depends on your printer and your scanner (or camera if you take a picture of it). More technically, it means that the information capacity heavily depends on the conjunction of dots per inch (DPI) resolution at printing time and at scanning time.

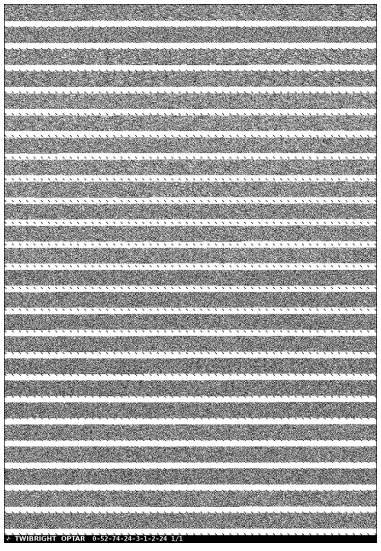
With a 600 DPI scanning resolution (all those experiments are done on a Konica Minolta c550)

QRCode (wikipedia): The maximum size of QR-Code is 2953 bytes, but you can stack several QRCodes on a single page (up to 24 Symbol-40 in my experiments). I decode them using zbar-tools. With 24 QRcode and UUencoding, I can put 47k of binary data. With direct UTF8 text, I can put 70k of text.



24 QrCodes on a page, with text, totaling 9549 words over 71 kilobytes, successfully decoded

Optar origin (unofficial Github repository) works well, with which I can get 100kilobytes per A4 page, with the following tweaks: 1) XCROSSES=45 and YCROSSES=65 2) using TIF scanner output instead of JPG 3) by decreasing the scanning DPI resolution (from 600 to 400 counter-intuitive)



90 kilobytes of MP3 in Optar(52x74) decoded with 0.68% irreparable bits, resulting in a few audible glitches, can go up to 118k per page at 52x74.

I could also get some reasonable results with stacked DataMatrix Same idea as QRCode, we have a maximum of 3116 bytes per matrix and one can pack several datamatrices on a page. However, the default decoder software on Linux (dmtxread) is not as effective as zbarimg (I succeeded to only pack 9 datamatrix per A4 page).

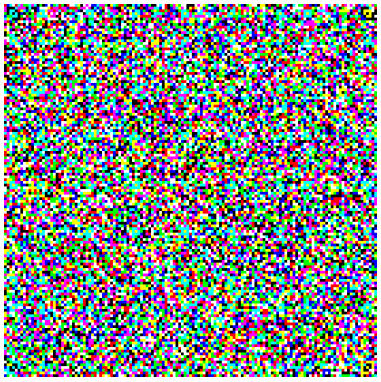
Other formats (failed or untested):

paperback (Windows version) (linux version) I tested the linux version with a 600 DPI scanner and it fails (issue). Note that their 500kb-per-page claim is not reproducible.

Color Dot Encodings

TL;DR; beautiful, but I haven't thoroughly investigated them yet, only got some small success with jabcode.

Adding color allows on to code more information per dot (3x more with three colors). The drawback is that colorized dot encodings requires color printers, which are more expensive.



Example of Jabcode

Colorized paper formats (failed or untested):

- jabcode is done by a German government organization, still maintained.
- colorsafe is full Python, can be installed through pip, claims up to 224kB of data per page. (fails, decoding fails after full-cycle)
- MrYakobo/tk is for fun
- elm-hccb for High Capacity Color Barcode (HCCB) (does not contain a decoder)
- blots (not tested, does not contain a decoder)

Theoretical information density maxima

The theoretical maxima is when you assume perfect scanning, alignment, etc. It depends on the scanning density and the color scale. At 300 DPI and black white, a page is 2480x3508 pixels, so it contains 8.7M bits which is 1100kB (1.1MB).

Redundancy and error correction

In a full cycle printing-transport-scanning there is a real risk that some dots in the image become corrupted. If the file format and use case allow for corruption (eg music in MP3), that's fine. If the exact bit-per-bit content is required, this risk can be mitigated by error correction.

For instance, QRCodes use Reed-Solomon encoding. In QRcode, there are four levels of error correction, low, medium, quartile and high, where the latter allows for up to 30% of the information to be lost.

It is always possible to use off-the-shelf error correction before encoding such as rsbep or PAR2.

How to transport paper-stored data?

You can take the sheets with you, send them by post, or even attach them to homing pigeons

See also

- Papers
  - In Robust and Fast Decoding of High-Capacity Color QR Codes for Mobile Applications (2017) Yang et al. claim to put 7.7kB on 38×38 mm2, which means ~300kb per A4 page. For experimenting, their tool is on Github.
  - In Paper encryption technology (2009), Anan and colleagues do the same thing as GPG plus paper encoding, as presented above.
- The SKOR codex contains 306 pages of printed images and printed sound based on a homegrown encoding.
- Wikinaut's notes is a gold mine with tons of links, in particular related to paper-storage of GPG and bitcoin cryptographic keys.
- Wikipedia:
  - Paper data storage
  - 2D bar code
- Tools
  - Paperback Paper backup generator suitable for long-term storage of bitcoin keys.

Created on April 15, 2020