Visible Cryptography

Bill Casselman

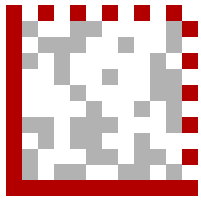
Cryptography literally permeates the air we breathe, since it is part of nearly all transmissions through cellular phones as well as many on the Internet, but there is one form of encryption that is visibly ubiquitous—those little matrices of pixels that occur on the mail of many countries of Europe and North America, inside letters from the Internal Revenue Service, on packaging of many commercial products in the U.S., and (at least in the future) on all pharmaceutical products in Europe. They are classified as bar codes, but in fact they are extremely sophisticated 2D arrays.

Their main advantage over older bar codes is that they allow a great deal of information to be packed into a very small space. All of them require several layers to be unpeeled before they can be interpreted, and in most applications the final layer involves elliptic curve cryptography (ECC). In postal use this provides a certificate of fee payment, among other things, and in pharmaceutical packaging it is intended to prevent counterfeits.

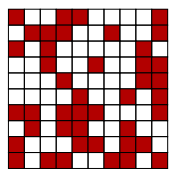
Data matrices come in a wide but fixed range of sizes, from 10×10 to 144×144. What a matrix displays is essentially an array of zeroes and ones. These are assembled into an array of bytes, each one 8 bits in size. These are unpacked according to one of several different modes of interpretation. The last several bytes of each matrix are added to the original message to allow error correction. After possible correction, one might have at hand a readable message, but more likely what one sees at this point is an array to be deciphered only by a secret key. Even if the message is not encrypted for security, it is likely to be somewhat condensed and meant to be interpreted according to some coding scheme particular to the application.

Bill Casselman is professor of mathematics at the University of British Columbia and graphics editor of the Notices. *His email address is* cass@math.ubc.ca.

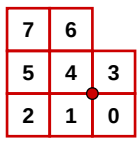
What I'll explain here is how the pixels are to be assembled into bytes, at least in one simple example. Let's look at the symbol on the right above, which was found on a carton of ice cream for sale in a supermarket in western Michigan. Every data matrix symbol is divided up into one or more smaller regions. The Canada postage symbol is divided up into 2×2 regions, while the one at hand is made up of just one. These regions are demarcated by special *peripheral pixels*, which are solid at left and bottom, alternating at right and top. These are used for alignment of bar code readers.



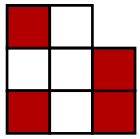
The core matrix is what you get by stripping away the peripheral pixels. Conventionally, its points are given coordinates (r, c) in matrix fashion, so r is row, c is column. Thus $(0, 0)$ is the upper left corner. A pixel is assigned the coordinate of its upper left corner.



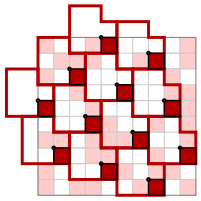
There are 8 pixels in each byte, and the region occupied by a byte (at least for this symbol) is a 3×3 square with a corner taken out. These pixels are ordered right to left, bottom to top.



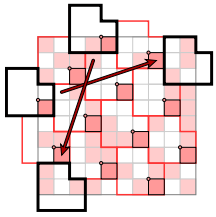
Thus the byte just below is 10001101 = 141.



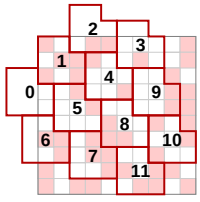
What I call the **origin pixel** is at lower right of the byte. Bytes are arranged so that one byte is packed in tightly at upper left, and then the rest spread out from there on the lattice spanned by vectors $[2,-2]$ and $[3,1]$.



For this particular size of data matrix, bytes are in bijection with their origin pixels inside the symbol. Since $100 = 10 \cdot 10$ is not divisible by 8, 4 pixels are ignored.



Bytes that do not fit inside the symbol wrap around on the other side, with a shift as indicated by arrows in the diagram below.



Bytes are numbered up and down along diagonals, starting at upper left.

What do we do with the bytes we can now read? There are 12 altogether, and the data matrix specification tells us that 7 of them are added at the end according to the ECC200 coding scheme, which is based on a Reed-Solomon code using the Galois field $\mathbb{F}_{256}$. This leaves 5 in the actual message: 84, 157, 171, 130, 129. There are several modes of interpretation of bytes, but this one is in the simplest ASCII mode.

The last byte 129 marks the end of the message (and would begin a padding sequence if the message didn't take up all available space). Bytes in the range $[1,128]$ are ordinary ASCII characters shifted up by 1, so 84 represents **S**. Bytes in the range $[130,229]$ represent integers in the range 0-99, so the sequence 157, 171, 130 represents 27, 41, 0. But now I have to say I have no idea what "S 27 41 0" means. Presumably it is expressed in some code known to packagers and distributors and provides an example of coding for compression rather than security. By contrast, in the Canadian postage stamp roughly 3/4 of the 174 message bytes are devoted to digital keys.

References

The only comprehensive reference in print that I know of is the book

Electronic Postage Systems: Technology, Security, Economics, Gerrit Bleumer, Springer, 2007.

It is mostly concerned with how cryptography is employed in postal systems. As far as low-level reading of data matrix symbols goes, one thing that is presumably essential for professional work is the ISO specification, which you can find by searching for ISO/IEC 16022-2006 at

`http://www.iso.org/iso/`

but it costs real money (224 Swiss francs!), and I have not seen it. The Wikipedia entry for **data matrix** at

`http://en.wikipedia.org/wiki/Data_ matrix`

is helpful, but doesn't have much detail. The webpage

`http://www.bcgen.com/datamatrix-barcode-creator.html`

has an interactive application that will allow you to create data matrix symbols from text you type in. As far as technical details are concerned, links to pages on data matrices can be found on

`http://grandzebu.net/`.

Follow links to the English version of bar codes, then to "data matrix". Another good source of information is

`http://www.libdmtx.org/`

which will allow you to obtain C++ code for reading and writing data matrices. This combined with the `grandzebu` site makes a fairly practical source for programming.

The 12×12 symbol on the ice cream carton has analogues on other groceries, as you can see at

`http://www.flickr.com/photos/nickj365/`

Is there a single company ultimately responsible for this phenomenon?

The Feature Column of the AMS (`http://www.ams.org/featurecolumn`) for February 2010 covered this topic in more detail.