VCK: the Visual Cryptography Kit

Frank Stajano
Olivetti Oracle Research Laboratory
& University of Cambridge Computer Laboratory

The idea of visual cryptography is a fascinating invention by Moni Naor and Adi Shamir (1994). See

http://www.wisdom.weizmann.ac.il/~naor/PAPERS/vis.ps

In its simplest form, it implements an unbreakable (in the information-theoretical sense) cryptosystem, rather similar to the one-time pad, with the additional twist that decryption does not require any computing equipment. James Bond carries a special "key" which is in fact a transparency with lots of random dots on it; when M faxes Bond a secret message, it too consisting of lots of random dots, Bond simply places his transparency over the ciphertext and reads out the plaintext! The stroke of genius here is the technique, explained in the poster, by which the boolean operation of "xor" is implemented by means of a visual "or". I decided to implement the system after seeing a demonstration of it at a brilliant talk by Adi Shamir himself. Since the point here was not so much using the cryptosystem but communicating to others the intellectual excitement of the invention, I wanted to be able to display all the intermediate results in graphical form.

I decided to use the *netpbm* suite of graphical file formats converters and I wrote a collection of small C++ programs, each of which performed a specific atomic operation. The only problem with this was that, to actually generate the slides, the user had to perform half a dozen separate operations, invoking these executables on the command line and giving them filenames as arguments (piping couldn't always be used since some operations required two inputs, and besides it was necessary to go through files to be able to inspect the results). Gluing things together with a batch file or shell script was cumbersome, as well as not portable between Windows and Unix. And it still required to run a separate image viewer to display all these intermediate results.

For a while I thought about writing a VCK extension to Python, so that I could use Python as the glue. But eventually I settled on Fredrik Lundh's PIL (Python Imaging Library) instead. A tiny Python-only module, `vck.py`, provides not only all the atomic operations but also the means to view any intermediate result in its own window with a single function call from within the driving script. Some example scripts and their outputs are shown on the poster. The greyscale variant has also been implemented: unlike in the C++ version, where I had to generate raw PostScript directly, here I could simply draw the figures on a Tk canvas and get the PostScript code for free.

The significance of this work in the context of IPC7 is as another demonstrator of the language's suitability for immediate practical experimentation (*making ideas executable*), which I view as one of Python's greatest strengths. The Python rewrite of VCK took only a few days, including the time spent learning the basics of PIL, and, compared to its C++ predecessor, resulted in a much more versatile and practical system for the end user, firstly because a convenient preview facility was now available and secondly because the VCK primitives could now be easily composed together by scripting. As an additional benefit, the code is now much shorter, to the point that it is perfectly reasonable to show the source to the user in order to explain what the primitives do. With the previous version, this would have exposed lots of irrelevant code to do with low-level image manipulation details. With the building blocks provided by VCK, such as the "bitmap" class and the boolean operations that take bitmaps as arguments, it is a simple matter for anyone to add new primitives in Python to implement any of the many variants of the visual cryptography idea that the cryptological research papers keep bringing out.

This code is totally un-optimised and has been written for the human reader rather than for the machine. The bitmap class exports methods to set and get the value of a pixel, and every other primitive of the toolkit goes through them. Most operations could be implemented much more efficiently at a lower level; but, given the purpose of this work, which is more about communicating ideas and enabling experimentation than providing a tool to perform actual encryption, this solution wins both for the time it took to develop it and for the advantage it offers of exposing the simplest possible source code. VCK is freely downloadable from

http://www.cl.cam.ac.uk/~fms27/vck/