Matryoshkryption

Lena DAVID

March 4, 2018

Abstract

Steganography can be defined as the process of concealing a piece of information or data inside another one, so that a third party cannot become aware of the presence of the embedded data. Such process can be useful in manifold situations, as the fact that it has been used since Ancient Greece shows.

Nowadays though, the techniques commonly used are far different from these of Ancient times, and steganography mostly applies to digital content. In particular, something quite interesting consists in combining it with cryptography, as this provides a means to benefit from the best of both worlds: assuming you use cryptography correctly, the data you conceal is confidential and thus safe from unwanted eyes, and what is more, the very fact that there is encrypted data in the cover file used is undetectable.

The present challenge aims to provide whomever may tackle it with some insights on classical steganographic techniques for different kinds of files, as well as on how combining steganography with cryptography can be implemented practically.

The following section describes two steganographic techniques that are at the core of the realization of the challenge. The last section describes in more detail what common methods are used in each step, and how each step leads to the next one.

Please note that this challenge was designed with pedagogical purposes and should thus not be considered as an illustration of a realistic practical implementation.

Matryoshkryption – Core principles

This challenge relies on two core steganographic techniques, which are combined with more common ones – the latter being described in the next section. Both

of these core techniques use the way some file formats are specified to embed additional content within them without making the original file unreadable.

The first technique is called *Angecryption*[1]. Its concept is simple and yet powerful: it consists in using AES-128 CBC to encrypt a file in one format to a valid file in another format, by choosing the IV so that the first block of the target (second) format - considered as a ciphertext block - decrypts into a plaintext block corresponding to the beginning of a file of the first format. The original algorithm was written in Python 2 and covered file formats such as PNG and PDF; in the framework of this project, we ported it to Python 3 and added support for raw MP3 files as well as for MP3 files with metadata.

The second core technique used to build this challenge consists in embedding a VeraCrypt volume in a MP4 file, so that both the encrypted volume and the original video are still readable. More specifically, VeraCrypt volumes can be created in way that allow plausible deniablility: you can create a first, outer volume, and create a second, hidden volume within the first one. You then put content in the outer volume that can seem interesting to a third party but contains nothing critical, and put the actual data you want to conceal in the inner volume. Thus, in a situation where the existence of a VeraCrypt volume is discovered by a adversarial third party who asks you to mount and decrypt the volume, you can provide the passphrase of the outer volume, displaying only the data you do not really care about without arousing the suspicion of the adversarial third party, who cannot prove there is more data than you are claiming there is.

The structure of a VeraCrypt container makes it possible to write the headers of a MP4 file on the part of the file that originally contains the headers of the outer volume. With some more tweaks, one can get a file that is a valid MP4 video file and also contains the hidden Veracrypt volume. The outer volume is lost in that hybridization process.

Previous works had been carried out on MP4-TrueCrypt hybridization. In the context of this project, we implemented a solution that allows MP4-VeraCrypt hybridization.

In the next section, we will describe more specifically what steganographic techniques are at stake in each step of the challenge, and how the different step are connected to each other.

Matryoshkryption – Detailed description

The challenge consists of four steps and is divided in two parts, each corresponding to two steps of the whole challenge. At each step, the challenger gets a file of a new type, and has to find cryptographic material concealed in it to proceed to the next step via Angecryption.

[1] See github.com/indrora/corkami/blob/master/src/angecryption/slides/AngeCryption.pdf for more details

The challenger initially gets a `.tar.gz` archive, which once decompressed reveals a PNG file. The steganographic technique used in that step to hide cryptographic material as well as an additional element is that of the LSB (**L**east **S**ignificant **B**it), where the least significant bit of the bytes that make each component (red, green, blue) of a pixel is altered to embed part of the data to conceal. In our case, the symmetric key is hidden in the plan made of the LSBs of the green component, the IV in that of the blue component. Both key and IV are hidden through "visual" LSB: one has to look at the black and white picture resulting of the extraction of the LSB plan to see the hidden message. Besides, the LSB plan of the red component holds an dictionary that will be needed by the challenger later on.

The challenger must then write a script that "de-angecrypts" the file using the found key/IV pair. The process of decryption reveals a PDF file.

The second step consists in a PDF document, which displays a music score. The dictionary found in the previous step must be used to decode the notes into a text message, which provides the challenger with a new key/IV pair, and with an intermediate flag. Additionally, the copyright line of the music score holds a pasphrase that will be of use later on. With the new key/IV pair, the challenger can proceed to the next step.

After decrypting the PDF document, one gets an audio file in a format which is non-standard but close to the MP3 file format. The cryptographic material for this step has been embedded in two different ways: the symmetric key is simply morse-encoded, and the IV is hidden in the spectrogram of the file – basically, a picture showing the IV is used to generate an audio file whose spectrography has the aspect of that picture. The audio file does not hold any additional content. Once again, decrypting the current file lets the challenger proceed to the next one, which appears to be a MP4 video file.

The last step consists of what first seems to be a mere MP4 file. However, the file is actually the result of the hybridization of a MP4 file and a VeraCrypt volume, as described in the previous section. The passphrase the challenger got from the PDF document leads to that information, and can then be used to decrypt the volume and mount it. The volume contains a single file, whose content is the final flag of the challenge.

A graph showing the different steps and their nesting can be found in Fig.1.

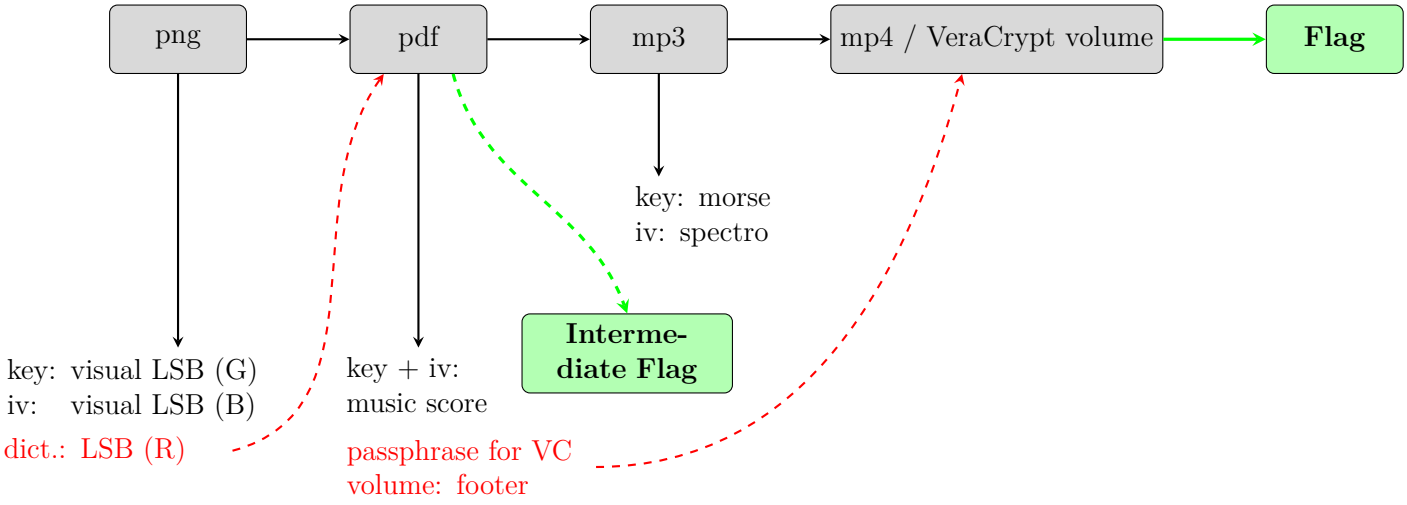

Figure 1: Challenge Overview