Task 1

Security of Additive Spread Spectrum Watermarking: PCA and ICA Attacks

Christian Riess (christian.riess@fau.de)

Let us drive a concrete attack on additive spread spectrum watermarks, to better understand security aspects of watermark embedding. To this end, let us work with two papers on watermarking security:

1. Gwenaël Doër and Jean-Luc Dugelay reported that a watermark can be removed via Principal Components Analysis (PCA) if multiple watermarks are embedded with the same secret carriers [DD04]. Note that this also works if the watermarks themselves are different, because the attack directly removes the subspace spanned by the carriers.

2. Patrick Bas and François Cayre reported that the actual carriers can be obtained via Independent Component Analysis (ICA) after application of the PCA attack [BC06]. If this attack succeeds, the attacker is obtains even stronger information about the watermark, which could for example be used to inject false watermarks.

In principle, we could attack our basic multi-symbol additive embeddings. However, in order to be closer to these two works, let us consider a slight variation of our multisymbol embedding called "Code Division Multiple Access" (CDMA). As embedding images, you can use images from the UCID database in `/proj/ciptmp/sichries/`.

1 CDMA Additive Spread-Spectrum Embedding

Code Division Multiple Access (CDMA) embeddings use a clever mathematical trick: If you aim to embed k bits, then choose k WM carrier vectors. Choose these vectors such that they are (perfectly or at least approximately) orthogonal. This makes it possible to embed these k carriers *on top* of each other, i.e., on the same pixels: since they are orthogonal, the correlation of one carrier does not interfere with the correlation of another.

Thus, if the host signal $\mathbf{x}$ (i.e., the image) has N pixels, we can choose k orthogonal watermark carriers $\mathbf{w}_i$, and perform the embedding as

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{k} \alpha \cdot \text{sign}(m_i) \cdot \mathbf{w}_i \ , \tag{1}$$

where $\text{sign}(m_i)$ is chosen as $+1$ if message bit m_i is 1, and -1 otherwise.

How can we get carriers that are orthogonal? If N is sufficiently large, then randomly drawn carriers $\mathbf{w}_i$ will with extremely high probability be orthogonal.

1. We aim to embed a 3-bit message (the numbers $0 \ldots 7$). Hence, $k = 3$. Draw 3 carriers and verify that they are (almost) orthogonal.

2. Perform the watermark embedding on some images from the UCID database in `/proj/ciptmp/sichries/`.

   Make the message $\mathbf{m}$ a parameter that you can easily vary. We will need that in the next task of the exercise.

   Make the embedding strength α a parameter that you can easily change. It might be useful to increase this parameter later if our signals are too weak.

   Keep the code sufficiently flexible that you can also work with only a subregion of an image instead of the whole image. This may help to prevent memory issues in the later stage of this exercise.

   Take care of a numerical detail: bytes may overflow/underflow if the embedding exceeds the value range of $[0; 255]$. My recommendation is to truncate such values to 0 or 255.

   Save your image in a lossless format (PNG) to keep things simple. Look at your image with embedded watermark. Is the visual appearance (fidelity) still OK?

3. Load your image, and detect the watermark message via correlation. Are you able to recover your 3 bit message?

2 Attacking the Watermark via PCA

Add different watermark messages with *the same carriers* $\mathbf{w}_i$ to multiple images using your implementation of task 1.

1. Load the watermarked images. Arrange them in a matrix for PCA decomposition, such that each watermarked image is a "observation vector" in the PCA sense.

   Let us try to choose the number of images and the length of the embedded watermarks in a way that we can use python's off-the-shelf PCA implementation. If this succeeds, we can omit implementation of the iterative PCA computation in the paper by Doërr and Dugelay (which is not complicated, but still a little bit of extra work).

2. Locate the PCA components that contain the 3-dimensional watermark subspace. The PCA components are orthogonal projections sorted by the variance of the observations. Thus, I would expect the watermarking subspace to reside either in the largest or smallest components.

3. Remove the watermark by cancelling these components (what is the least-effort programming solution to achieve this?). Check with your watermark detector that the signal is indeed gone.

3 Recovering the Watermark Carriers

Bas and Cayre present a blind source separation attack to recover the watermark carriers in Sec. 2.3 of their work [BC06]. With the PCA already available, the attack consists solely of the application of an independent component analysis to separate the orthogonal watermarking carriers.

Validate your attack by correlating the original carriers with the ICA-calculated carriers!

4 Totally Voluntarily: JPEG Compression

If you had fun so far: our watermark is not super robust, but it should have at least some robustness. Slightly increase the embedding strength α, apply a high-quality JPEG compression, and redo the experiments. Does watermark detection still work? Does watermark cancellation and carrier estimation still work?

References

[BC06] Patrick Bas and François Cayre. Natural Watermarking: a secure spread spectrum technique for WOA. In *International Workshop on Information Hiding*, pages 1–14, 2006.

[DD04] Gwenaël Doërr and Jean-Luc Dugelay. Danger of Low-Dimensional Watermarking Subspaces. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages III–93 – III–96, 2004.