On Multiple Watermarking

Nicholas Paul Sheppard
School of IT and CS
University of Wollongong
NSW 2522
Australia
nps@uow.edu.au

Reihaneh Safavi-Naini
School of IT and CS
University of Wollongong
NSW 2522
Australia
rei@uow.edu.au

Philip Ogunbona
Motorola Australia Research Centre
12 Lord St
Botany NSW 2019
Australia
pogunbon@arc.corp.mot.com

ABSTRACT

Mintzer and Braudaway [6] once asked: *If one watermark is good, are more better?* In this paper, we discuss some techniques for embedding multiple watermarks into a single multimedia object and report some observations on implementations of these techniques.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Digital Watermarking*

General Terms

Algorithms,Performance,Security

Keywords

Digital watermarking

1. INTRODUCTION

Many digital watermarking algorithms have been proposed for a variety of different tasks on multimedia creations. These purposes are not mutually exclusive and any one multimedia creation might be subjected to several different watermarking applications, or several repetitions of the same application. For example, in proof-of-ownership applications, it is common for large artistic works such as feature films and music recordings to have several creators, and several layers of distributors and retailers. Some more sophisticated protocols, such as that in [5], are implemented using two or more watermarks in a single object.

Mintzer and Braudaway [6] give a limited discussion of some specific examples of combinations of watermarking applications. Their paper discusses only one solution to the multiple watermarking problem, here described as the re-watermarking solution. In this paper, we discuss several methods (including re-watermarking) and make observations on their relative strengths and weaknesses.

All of the algorithms described here are, in general, applicable to any type of multimedia object for which we have a basic watermark embedder. Our implementation was for grey-scale images. We implemented the algorithms for each of two basic watermark embedders, [2] and [4], which we chose as typical examples of frequency domain and spatial domain embedders, respectively.

2. TERMINOLOGY

For convenient exposition of our methods, we assume that we have a method for embedding a single watermark in a multimedia object which we will refer to as the *underlying algorithm*, and denote the watermarked object it returns by $W(X, k)$ for a host object X and key k. We assume that this algorithm returns a detection score $\bar{W}(X, k)$ on an object X, indicating the presence or otherwise of a key k. Using different types of watermarking functions is similar.

We will assume that we have a set of m keys $\mathcal{K} = \{k_1, \ldots, k_m\}$ to be embedded into an object.

We say that a multiple watermark is *separable* if it is possible to detect each component key k_i individually in the final watermarked object. We say that a multiple watermark is *completable* if it is possible for the detector to distinguish between $\mathcal{K}$ and a smaller set of keys; that is, the detector knows when the full set of keys is not present.

We distinguish three types of protocols for embedding multiple watermarks: a *trusted embedder* that accepts keys or watermark patterns and embeds them in an internal copy of the original; a *combiner* that accepts watermarked objects and combines them to form the final watermarked object; and a *serverless embedding* in which the watermarked object is produced by the key-holders alone. In general, any multiple watermarking algorithm may be implemented by a variety of protocols but not always all of them.

3. ALGORITHMS FOR MULTIPLE WATERMARKING

3.1 Re-watermarking

The most obvious and straightforward method of embedding multiple watermarks is to add them one after the other, that is, compute $\hat{X}_1 = W(X, k_1)$, then $\hat{X}_2 = W(\hat{X}_1, k_2)$, etc., until we arrive at the final object $\hat{X} = \hat{X}_m$. This method

lends itself to simple serverless embedding. This technique has also been used as an attack, e.g. in [2] and [3].

If the underlying algorithm is robust against re-watermarking, watermarks created this way are typically separable by simply computing $\bar{W}(\hat{X}, k_i)$. This is true, for example, of [4].

In our implementation of [2], however, we found that only k_1 could be detected in $\hat{X}$ using X as the original image. A subsequent key k_i could only be detected by using $\hat{X}_{i-1}$ as the original image. This seems to be because embedding the first watermark re-arranges the order of the 1000 highest DCT coefficients, causing a second watermark pattern to be scrambled relative to the original image. [2], on the other hand, describes a re-watermarking experiment in which they claim to have recovered all five watermarks that they embedded. We do not know why our results differ from theirs.

This sort of behaviour, while undoubtedly problematic in some applications, might also be useful in applications where retrieval of one watermark should depend on the retrieval of another. For example, it allows us to determine the order in which watermarks were embedded.

Inevitably, the object becomes more degraded with every new watermark inserted into it, both in terms of PSNR and perceived quality. In our experiments, both the detection scores and PSNR dropped fairly rapidly with each new watermark added. For the method of [2], in particular, it is straightforward to show mathematically that the expected value of $\bar{W}(\hat{X}, k)$ drops by a factor of $\sqrt{2}$ for every watermark added.

3.2 Segmented Watermarking

Instead of placing watermarks on top of each other, we can divide up the space available for watermarking and allocate each division to a different watermark. In general, let $Q = \{q_1, \ldots, q_r\}$ be a (public) partition of the object with $r \geq m$. Our implementation, for example, simply partitions an image into square blocks. Let ϕ be a mapping of Q onto $\mathcal{K}$. For each $q_i \in Q$ we compute $\hat{X}_i = W(q_i, \phi(q_i))$, and assemble the final watermarked object $\hat{X}$ in the obvious way using a combiner.

Watermarks can be detected by partitioning the object according to Q and testing the resulting pieces. The algorithm is obviously separable, and is also completable since the detector can test for a "covering" of the object. If all keys are present, the detector can find a watermark in every segment, otherwise it cannot.

Clearly, r limits the number of watermarks that can be embedded using this method. Up until Q has been exhausted, adding more watermarks does not degrade the image any more than having a single watermark. As r grows larger, the segments grow smaller and it becomes more difficult to embed a watermark into the segments, which places a limit on r and hence on m.

3.2.1 Interleaved Watermarking

Rather than divide the object into contiguous blocks, we can sub-sample it at some interval, with a different offset for each key. For example, one watermark can be assigned the odd-numbered columns and another watermark assigned the even-numbered columns. Each sub-sampled object is watermarked independently, and the watermarked objects are interleaved together to form a watermarked version of the original object.

This method tends to cause a large degree of degradation, particularly for watermarks inserted in the frequency domain. Using the method of [2], we found that the final image tended to have a rippled look corresponding to the interleaving pattern, which we intepret as an interference pattern between the watermarks. While the spatial domain watermark of [4] fared much better, the image quality was still quite poor compared to most of the other techniques we examined. Hence, the interleaved method seems suitable only for very small values of m, probably no greater than 3.

3.3 Composite Watermarking

Rather than repeatedly watermarking an object, for many underlying algorithms it is possible to build a single composite watermark from a collection of watermarks. The composite watermark is then embedded into the host object in the usual way by a trusted embedder. This method is used in [4] for embedding information in the relative arrangments of several watermarks.

For this method, we assume that the underlying algorithm has a function $w(k)$ that takes a key k and produces a watermark pattern, for example, by using a pseudo-random number generator. The watermark pattern may be a vector, as in [2], or a matrix, as in [4], or something else entirely. We further assume that there is some binary operation $\odot$ for combining patterns, e.g. we used vector addition for [2] and matrix addition for [4]. We can then compute

$$w_0 = w(k_1) \odot \cdots \odot w(k_m)$$

and embed w_0 into the host object.

The composite watermark will be separable if the watermarking patterns are orthogonal (or uncorrelated) in some sense relevent to the watermark detection. Most watermarking algorithms have this property.

In particular, both of our underlying algorithms use correlation-type detection, and different watermark patterns are uncorrelated. Hence the expected value of the inner product $w(k_i).w(k_j)$ of two watermark patterns is zero and

$$\begin{aligned} w(k_i).(w(k_i) + w(k_j)) &= w(k_i).w(k_i) + w(k_i).w(k_j) \\ &\approx w(k_i).w(k_i) \end{aligned}$$

so we expect $w(k_i)$ to have high correlation with the embedded pattern $w(k_i) + w(k_j)$.

As for re-watermarking, it is simple to show that the expected value of $\bar{W}(\hat{X}, k)$ for [2] drops by a factor of $\sqrt{2}$ for every watermark added. However, our experiments suggested that the detection scores and PSNRs tend be a little better than for the re-watermarking algorithm. The segmented algorithm gave much better image quality, but the composite

algorithm gave better detection performance, particularly for [4].

3.3.1 Averaged Watermarking

Rather than compose the watermark patterns before embedding, we could instead perform the composition step on the watermarked objects, taking advantage of most watermarking schemes' resistance to collusion attacks.

An obvious way to compose a set of objects is to average them together using a combiner. In our experiments, each pixel of the output image was simply the average of the corresponding pixels in the input images; using Fourier, DCT, etc. coefficients instead is also possible and may seem more appropriate if the watermark is inserted in those domains. The averaging process moderates the strength of the watermark signals compared to the host signal, so this method weakens detection performance but improves the image quality compared to other methods.

4. SECURITY ISSUES

4.1 Outsider Attacks

Other than for segmented watermarks, which are subject to some special attacks described below, it seems that an outsider (that is, someone without a key from $\mathcal{K}$) would need to perform the same steps to break a multiple watermark (in a cryptographic sense) as he or she would to break the underlying algorithm, and hence the multiple watermark seems to be as secure as the underlying algorithm with respect to outsiders.

4.1.1 Robust Watermarks

It seems reasonable to expect a multiple watermark to have the same robustness qualities as the underlying algorithm used. It is obviously unreasonable to expect a multiple watermark to have better robustness than the underlying algorithm. Hence, we did not test the multiple watermarks for robustness against attacks such as rotation, scaling, etc., that are known to defeat our underlying algorithms.

We tested each of the algorithms described, with both underlying algorithms, for robustness against JPEG compression, Gaussian noise addition and the FMLR attack [1] on two different images with between one and five different keys. In all cases, the multiple watermark performed very similarly to the the underlying algorithm when subjected to the same attack – the best we could reasonably have hoped for.

4.1.1.1 Segmented Watermarks

Any single watermark may be subject to a cropping attack if the segment to which it has been assigned forms an insignificant part of the object. One solution is to use an interleaved watermark. Another solution, if $r \gg m$, is to assign each key to many segments so that each key gets a turn at being in a significant part of the object.

The latter solution gives rise to another potential problem in that an attacker now has access to several different objects (segments) containing the same watermark pattern. The attacker can average these segments together in in the hope that the host components average out to zero while the watermark components will re-inforce themselves, thus allowing the attacker to recover the watermark pattern. We tried this attack against the standard version of [4] (in which a single watermark pattern is tiled over the whole image) and found it to be fairly ineffective at removing the watermark. However, the possibility of this attack can be eliminated entirely by associating each key with a sequence of watermark patterns so that each segment belonging to one key contains a different watermark pattern.

Segmented watermarks are also subject to de-synchronisation attacks that attempt to prevent the detector from correctly recovering Q. This is a common problem in watermarking and can be solved by comparison with the original object or by embedding synchronisation information into the object.

4.1.2 Authentication Watermarks

In a segmented authentication watermark, the authentication watermark can only guard the segment that it is in and an attacker may be able to make undetected meaningful modifications by avoided segments containing authentication data. Analogously to the cropping attack on a robust watermark, the solution is to distribute the watermark throughout the object by inserting it in many (closely-packed) segments, or by using interleaving, so that it is extremely difficult for an attacker to make meaningful modifications without disturbing a segment containing the authentication data.

4.2 Insider Attacks Against Ownership Watermarks

Proof-of-ownership applications introduce a number of potential attacks by insiders acting maliciously against other insiders, for example, to wrongfully exclude an author.

In many applications, we might reasonably assume that the insiders have a copy of the original object – after all, they may have been the authors in a proof-of-authorship scenario. Obviously no watermark can resist removal by an opponent possessing the original, and an insider is free to leak a copy of the object containing some watermark other than the correct one. The solution to these problems must lie in an infrastructure outside watermarking that is not covered here.

Since all of the algorithms described above have an independent, uncorrelated watermark pattern for each participant, any insider without the original is in exactly the same position as an outsider if he or she is attempting to remove another insider's watermark (assuming that the protocol used to collect and embed the watermarks is secure).

An insider in possession of the original can compare the original with the watermarked object to recover the watermark pattern by a subtractive process. This pattern can then be copied into another object, for example, to frame the other authors. This attack could be prevented by making watermark patterns depend on the object as well as the watermarking keys, for example, by combining a hash of the original with the keys.

If a multiple watermark is non-completable, it is possible

for a subset of the watermark owners to claim to a detector that they are the sole owners of the object. If the other watermark owners do not or cannot act to assert their rights, a completable watermark can still protect their interests.

For some algorithms, it is possible for one watermark to "over-power" the other watermarks by embedding it with an abnormally high strength. If one watermarker generates a watermark pattern with, say, five times the average amplitude as that used by the other watermarks, the strong watermark can obscure the normal watermarks. This is particularly effective against detectors that use normalisation, such as in [2], since the detector will normalise to the strength of the strong watermark and make the normal watermarks seem very weak.

One solution to this problem is for the embedding machinery to perform a normalisation step before embedding; for example, it is simple to normalise all of the patterns input to a composite watermarker before the composite watermark is created. A more punitive solution would be for the embedder to compute pair-wise distances (PSNR, Euclidean, etc.) between its inputs and reject any outliers.

5. CONCLUSION

We have discussed several straightforward methods for embedding multiple watermarks into a single multimedia object, and reported observations based on our implementation of these methods. We have also introduced some interesting properties for multiple watermarks, and described some potential security problems in multiple watermarking applications that are not applicable for single watermark applications.

We do not claim that the methods described here exhaust the possibilities for multiple watermarks, or for attacks on the ones we have suggested. Future work also includes devising and evaluating protocols for collecting and embedding watermarks, and for performing detection.

6. ACKNOWLEDGEMENTS

This work was partially funded by the Motorola Australia Research Centre.

7. REFERENCES

[1] R. Barnett and D. E. Pearson. Frequency mode LR attack operator for digitally watermarked images. *Electronics Letters*, 19:1837–1838, 1998.

[2] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. A secure, robust watermark for multimedia. In *Information Hiding: First International Workshop*, pages 185–206, Berlin, Germany, 1996. Springer.

[3] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung. Resolving rightful ownerships with invisible watermarking techniques. *IEEE Journal on Selected Areas in Communications*, 16:573–586, 1998.

[4] T. Kalker, G. Depovere, J. Haitsma, and M. Maes. A video watermarking system for broadcast monitoring. In *IS&T/SPIE Conference on Security and Watermarking of Multimedia Contents*, pages 103–122, San Jose, California, 1999. SPIE.

[5] N. Memon and P. W. Wong. A buyer-seller watermarking protocol. *IEEE Transactions on Image Processing*, 10:643–649, 2001.

[6] F. Mintzer and G. W. Braudaway. If one watermark is good, are more better? In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2067–2069. IEEE, 1999.