EURASIP Journal on Applied Signal Processing 2004:14, 2214–2223

Secure Multimedia Authoring with Dishonest Collaborators

Nicholas Paul Sheppard
School of Information Technology and Computer Science, The University of Wollongong, NSW 2522, Australia
Email: nps@uow.edu.au

Reihaneh Safavi-Naini
School of Information Technology and Computer Science, The University of Wollongong, NSW 2522, Australia
Email: rei@uow.edu.au

Philip Ogunbona
School of Information Technology and Computer Science, The University of Wollongong, NSW 2522, Australia
Email: philipo@uow.edu.au

Received 31 March 2003; Revised 16 December 2003

Many systems have been proposed for protecting the intellectual property of multimedia authors and owners from the public at large, who have access to the multimedia only after it is published. In this paper, we consider the problem of protecting authors' intellectual property rights from insiders, such as collaborating authors and producers, who interact with the creative process before publication. We describe the weaknesses of standard proof-of-ownership watermarking approaches against dishonest insiders and propose several possible architectures for systems that avoid these weaknesses. We further show how these architectures can be adapted for fingerprinting in the presence of dishonest insiders.

Keywords and phrases: digital watermarking, collaboration, multiple watermarking, proof of ownership, fingerprinting.

1. INTRODUCTION

Multimedia security research has focused on security of published content, and upon protecting the intellectual property of the content owners and creators from malicious end users. These systems, however, do nothing to resolve intellectual property disputes that arise prior to publication, for example, between collaborating authors.

We will consider intellectual property protection in the case where the disputing parties are (or claim to be) involved in the creation stage of the content in dispute. We will specifically consider *proof-of-ownership*, that is, enabling authors to prove to an arbiter that they were involved in the authoring process. We will also consider how our architectures can be adapted to *fingerprinting*, that is, enabling authors to determine the identity of an author who has "leaked" a copy of the work without permission from the other authors.

Watermarking solutions to the above problems have been proposed in the case where the adversary has access only to the published work, that is, is an *outsider*. In Section 2, we will describe the weaknesses in these solutions against an adversary who is part of the authoring process—that is, is an *insider*—who in a naïve protocol may be able to obtain a copy of the unwatermarked original. While some previous algorithms have considered watermarks for representing the collaborative effort of several contributors [1, 2], protocols by which such watermarked objects are created have not been extensively studied.

In Section 3, we will describe several possible protocols for multimedia authoring in the proof-of-ownership setting that avoid the weaknesses in naïve protocols by preventing insiders from obtaining a copy of the unwatermarked original. We will further show how these protocols can be adapted for fingerprinting in Section 4.

2. INTELLECTUAL PROPERTY PROTECTION USING WATERMARKS

A *digital watermark* is a secret signal embedded into a multimedia object that can only be detected or recovered by someone possessing a secret key. Many techniques for embedding watermarks in all manner of multimedia objects have been proposed; a survey is given in [3].

In the watermarking solution to the proof-of-ownership problem, the owner of a multimedia object embeds a watermark into the finished object prior to publication, and

publishes the watermarked object instead of the original version. If, at a later time, an imposter claims to be the originator of the published object, the true owner can prove his or her ownership by demonstrating the existence of the secret watermark to an arbiter.

This solution assumes that the adversary has access only to the published version of the object. Existing watermarking systems generally make an implicit assumption that watermarking is more or less the final step before publication, since they take a finalised object as input and output the object to be published. Without an additional protocol to govern access to the object prior to watermarking and publication, an insider is able to take a copy of the object without a watermark.

Clearly, an adversary in possession of an unwatermarked object can circumvent the protocol described above, since this copy does not contain the legitimate owners' secret watermark. In this paper, we will discuss protocols for authoring multimedia such that no party gains access to an unwatermarked version of the content, thus preserving the integrity of the protocol described above even in the presence of dishonest insiders.

Of course, any attack on a watermarking system that is available to outsiders is also available to insiders. In this paper, however, we will only consider attacks by insiders that are not available to outsiders. Our example watermarks will be chosen for ease of exposition rather than security against conventional outsider attacks.

2.1. Multiple watermarking

We will use multiple watermarking to represent the intellectual property rights of multiple contributors, that is, each contributor will have a personal watermark and the final object will contain the collection of these personal watermarks. An overview of schemes that allow multiple watermarks to be embedded into a single object is given in [2].

We distinguish three classes of multiple watermark:

(i) a *rewatermark* created by watermarking the object with several different watermarks in turn;

(ii) a *segmented watermark* created by dividing the object into pieces and embedding a different watermark into each piece;

(iii) a *composite watermark* created by composing several different watermarks into a single watermark (i.e., the composition is a kind of shared secret) and embedding this composition.

Separability

For our purposes, we assume that all of our multiple watermarks are *separable*, that is, that it is possible to detect each component watermark individually in the watermarked object.

Segmented watermarks are always separable, since each segment (and therefore watermark) is tested independently.

Watermarks produced by rewatermarking are usually separable if the underlying algorithm is robust against rewatermarking. For the applications discussed in this paper, watermarks are required to be robust against rewatermarking since otherwise an attacker can defeat the proof-of-ownership protocol by simply rewatermarking the object.

Composite watermarks may or may not be separable, depending on the way composition is performed. For the examples in this paper, composition is performed by simple vector or matrix addition of independently chosen, randomly distributed watermark patterns. A statistical detector can separate the component watermarks since the watermarking patterns are mutually uncorrelated. Some more exotic methods of composition, such as those suggested by Guo and Georganas [1] may require modified detectors. The specifics of each of our examples will be discussed in Section 3.

Capacity

Obviously, there is a limit to the number of watermarks that any multimedia object can contain. Watermarks formed by composition or rewatermarking gradually degrade the image as each new watermark is added. In a segmented watermark, the number of watermarks that can be embedded is limited by the number of available segments.

In general, it seems reasonable to believe that the watermarking capacity of an object would be commensurate with the number of authors working on it. It does not seem very likely, for example, that a still image would require more than two or three authors to produce. Larger works that may require large teams of authors to produce, such as feature films, have a much greater watermarking capacity.

2.2. Our model

In our collaborative version of the proof-of-ownership problem, our aim is to prevent a dishonest insider from denying the contribution of other insiders. This is not much different from the aim in the conventional proof-of-ownership problem, except that the dishonest outsider in that model is replaced by a dishonest insider here. In both the conventional model and our one, an honest insider desires to produce evidence that proves his or her case against the dishonest party.

We define an *insider* as someone who has access to the multimedia content before publication, such as an author. We will sometimes use the term "author" to mean an actively contributing insider. Each insider is assumed to have some secret information which he or she can use to embed a secret watermark known only to that insider. We will give some examples of how this secret information is used in Section 3.

An *outsider* is anyone who is not an insider. We will not explicitly consider protection from dishonest outsiders in this paper. During the prepublication phase, we assume that the insiders have suitable private channels which cannot be listened to or tampered with by outsiders. (By the letter of the definition, an outsider who could do such things would become an insider).

We are not aware of any method by which a computer system can make artistic decisions about the contributions of authors. We will therefore assume that

(i) all insiders are permitted to make arbitrary changes to the object being authored, whatever their perceived artistic value is;
(ii) all insiders have an equal right to be represented as the owners of the finished object, whatever a human judge might think of their contribution.

It is possible to develop more complex systems that use access control structures to constrain authors to changing only certain regions of the object; give different preassigned weights to different authors' watermarks; eliminate insiders' watermarks if that insider makes no contribution; and so forth, but for simplicity we will not discuss these straightforward extensions here.

Any insider is able to take a copy of the object being authored at any time, and optionally make private changes to it, possibly including "changes" made by ignoring the contributions of other authors. An object created other than by the legitimate publication procedure will be referred to as a *rebel object*. We will not attempt to prevent authors from creating and publishing rebel objects, since such activity is analogous to an outsider who takes a copy of the published object and makes his or her own changes to it, and this cannot be prevented in the general watermarking model. We do, however, demand that rebel objects contain the watermarks of all the contributors to the object, so that the rebel insider cannot deny the other insiders' contribution to any object, whether it is a rebel one or not.

3. ARCHITECTURES FOR SECURE AUTHORING

In this section, we will describe several possible architectures for multimedia authoring systems that provide intellectual property protection against dishonest insiders who participate in the authoring process itself, avoiding the vulnerability of the conventional approaches to dishonest insiders described in Section 2. For ease of exposition, we will describe only proof-of-ownership watermarking in this section. We will show how to adapt the constructions here for fingerprinting in Section 4.

As in the conventional proof-of-ownership case, we cannot appeal to encryption for protection against dishonest parties since all parties must have access to the unencrypted object if they are to make any use of it. Watermarking aims to solve this problem by embedding subliminal information into an unencrypted object that deters illegitimate use by threatening an illegitimate user with detection.

Our general approach is to maintain a version of the work-in-progress that contains a "watermark-in-progress." Changes to the work-in-progress result in corresponding changes to the watermark. The authors, therefore, do not have an opportunity to obtain an unwatermarked version of the object, but are still able to access a usable version of the object. An author making some illegitimate use of the object can then be dealt with in the same way as in the conventional case.

Of course, any form of collaborative authoring system requires some form of concurrency control to prevent mishaps due to two or more authors trying to edit the same thing at the same time. This is a well-known problem with well-known solutions in concurrent programming, and for simplicity we will not explicitly mention them here.

3.1. Authoring with a trusted repository

If the authors have access to a repository which they all trust with their watermark information and the unwatermarked original, it is relatively straightforward to implement a solution to our problem, using an architecture similar to the IETF's WebDAV protocol [4].

Whenever an author wishes to make a change to the object, the repository makes a watermarked version (containing the watermarks of all authors) of its master copy, and transmits this to the editing author. The editing author transmits the changes back to the repository, which incorporates them into its unwatermarked original. In a naïve implementation, the master copy may become degraded due to the repeated addition of watermarks every time the object is checked out; however, we will give an example of how this can be avoided in Section 3.2.2.

3.2. Authoring with a blind repository

By embedding the watermark in an encrypted domain, it is possible to implement a system in which

(i) no party, including the server, has access to the unwatermarked original X;
(ii) the watermark w^i is known only to author i;
(iii) all the authors have access to the watermarked object $\hat{X}$ containing all of the authors' watermarks.

Some techniques for embedding watermarks in encrypted domains are described by Fridrich et al. [5, 6], Yen [7], and Memon and Wong [8]. Memon and Wong's construction, based on a *privacy homomorphism* [9] between the encryption and watermarking functions, is the most convenient for our purposes.

An encryption function $E(X,k)$ is a privacy homomorphism with respect to a function $f(X,Y)$ if and only if

$$E(f(X,Y),k) = f(E(X,k),E(Y,k)) \tag{1}$$

for all plaintexts X and Y, and keys k. For example, RSA [10] is a privacy homomorphism with respect to fixed point multiplication.

Let each author i have a secret watermark w^i, and let k be a global encryption key known to the authors (and no one else, including the server). Let $W(X,w)$ denote watermarking an object X with a watermark w and let $g(X,\delta X)$ be a function that applies the changes δX to X. We require that $g(X,\delta X)$ be invertible, that is, given an object X and another object X', it is possible to compute δX such that

$$X' = g(X,\delta X). \tag{2}$$

Let $E(X,k)$ be an encryption function that is a privacy homomorphism with respect to both $W(X,w)$ and $g(X,\delta X)$.

To initialise the server, each author transmits $E(w^i, k)$ to the server using a private secure channel, and the server records the encrypted watermarks for future use. The server's master copy of the encrypted object can be initialised by having an author choosing a random object X and transmitting $E(X, k)$ to the server. Alternatively, if the encryption function is such that the server can randomly generate a valid ciphertext without knowing the key, it is possible for the server to simply choose its own random "encrypted" object $E(X, k)$.

An author wishing to modify the object X makes a request to the server. Let $W^*(X, w^1, \ldots, w^m)$ denote the object X watermarked with each watermark w^1 up to w^m in turn (by rewatermarking), where m is the number of authors. Note that composition rather than rewatermarking is also possible if the encryption function is a privacy homomorphism with respect to the composition function; we will see an example of this in Section 3.2.2. The server computes

$$W^*(E(X,k), E(w^1,k), \ldots, E(w^m,k)) \tag{3}$$

and transmits this to the author that made the request.

Since $E(X, k)$ is a privacy homomorphism with respect to $W(X, w)$, we can see that

$$W^*(E(X,k), E(w^1,k), \ldots, E(w^m,k)) = E(\hat{X}, k) \tag{4}$$

by applying the homomorphic property m times. Hence, the author receiving $E(\hat{X}, k)$ can decrypt the watermarked object

$$\hat{X} = W^*(X, w^1, \ldots, w^m) \tag{5}$$

and edit this object as normal to produce a new object $\hat{X}'$. The server, however, cannot decrypt the object since it does not know the key k.

The author computes δX such that $\hat{X}' = g(\hat{X}, \delta X)$ and transmits $E(\delta X, k)$ to the server (in practice, the author may just create δX directly by storing the changes he or she makes). The server computes

$$E(g(X, \delta X), k) = g(E(X,k), E(\delta X, k)) \approx E(X', k) \tag{6}$$

and makes this its new master copy of the encrypted object. Some care needs to be taken in the choice of $g(X, \delta X)$ to keep the approximation manageable. For a well-chosen $g(X, \delta X)$, the approximation can be eliminated altogether, and we will give an example of such a choice in Section 3.2.2.

3.2.1. Limitations

Memon and Wong note that this system of embedding watermarks in an encrypted domain prevents the watermarking algorithm from using any perceptual information about the object. An alternative approach that may avoid this problem is the random transform domain technique of Fridrich et al. [5, 6], in which watermarking is performed in a random frequency-like domain. Due to space considerations, we will not explore this alternative further in the present paper.

3.2.2. An example

In their example of a homomorphic watermarker, Memon and Wong use RSA encryption and the watermarking algorithm of Cox et al. [11]. However,

(i) using a nonoblivious watermarking method is inconvenient in our situation, where we have stated that the original should be inaccessible (though a collusion of the server and at least one author could reveal it);

(ii) asymmetric encryption, such as RSA, results in a many-fold expansion in the size of the object when used in the pointwise fashion required for the construction to work;

(iii) pointwise encryption is potentially vulnerable to attacks because of the small number of possible plaintexts;

(iv) applying changes in the transform domain is difficult since human authors work in the spatial domain.

As we do not need asymmetric encryption for our situation, a more convenient choice for the encryption function is permutation in the spatial domain. Since permutation is homomorphic with respect to any pointwise function, we have great flexibility in choosing a watermarking function. Let the watermark of author i be represented by a matrix w^i of the same size as the image to be watermarked, and let watermarking be performed by matrix addition of the watermark to the image. Several simple watermarking algorithms, such as the Patchwork algorithm of Bender et al. [12] and the algorithm of Pitas [13], can be implemented in this way.

A convenient choice for $g(X, \delta X)$ is the function that selectively replaces the elements of a $p \times q$ matrix X with those from another $p \times q$ matrix δX to form a new matrix X' with

$$X'(x,y) = \begin{cases} X(x,y), & \text{if } \delta X(x,y) = -1, \\ \delta X(x,y), & \text{otherwise.} \end{cases} \tag{7}$$

An inverse for any X and X' using this function can be derived from a simple pointwise comparison.

With this choice of $g(X, \delta X)$, watermarked pixels obtained from the server and unmodified by the author are not returned to the server since they are at positions where $\delta X(x, y) = -1$. The only pixels incorporated into the server's master (unwatermarked) copy are the unwatermarked ones created by authors after modifying the image.

Let κ be a permutation on the elements of a $p \times q$ matrix, known only to the authors. Let w^i be a $p \times q$ watermarking pattern known only to author i. Let the image being authored be X, and let the server have $\kappa(X)$ and $\kappa(w^i)$ for all authors i. The procedure for an author i to edit the object is the same as before, except that it is possible to use a composite watermark here.

(1) The server computes a composed permuted watermark pattern $\kappa(w^*) = \sum_{j=1}^{m} \kappa(w^j)$.

(2) The server computes the permuted watermarked object by $\kappa(\hat{X}) = \kappa(X) + \kappa(w^*)[= \kappa(X + w^*)]$, and transmits $\kappa(\hat{X})$ to author i.

(3) Author i uses the inverse permutation to get $\hat{X} = \kappa^{-1}(\kappa(X + w^*)) = X + w^*$.
(4) Author i makes changes δX to $\hat{X}$, where δX is a $p \times q$ matrix with entries of -1 where the pixel at that position was unchanged, or the new pixel value otherwise.
(5) Author i transmits $\kappa(\delta X)$ to the server.
(6) The server computes its new master copy of the permuted original by $\kappa(X') = g(\kappa(X), \kappa(\delta X))[= \kappa(g(X, \delta X))]$.

Using either the Patchwork or Pitas algorithms, the composed watermark w^* can be detected as usual by a conventional detector since it is a valid watermark pattern of itself. It is also possible for a conventional detector to separate the individual watermarks $w^1, \ldots, w^m$ since they are uncorrelated and composition in this fashion is equivalent to rewatermarking in these systems.

3.2.3. An attack

Given a watermarked object and its original version, an attacker can attempt to estimate the watermark signal by comparing the two. This leads to a variety of possible attacks in which a dishonest author submits a specially-constructed object to the server, immediately requests the watermarked version, and uses the two versions to obtain information about the other authors' watermarks.

Suppose, for example, an author creates an object X and submits this to the server. This X could be the initial object given to the server during the initialisation phase, or it could be created by checking out an existing object and overwriting it before resubmission.

If the author immediately requests the object again, the author will obtain the watermarked version,

$$\hat{X} = W(X, w^1, \ldots, w^m). \tag{8}$$

The author now knows both X and its watermarked version, which may allow the author to compute the (composed) watermark. For example, in the permutation example above, the author can compute

$$\begin{aligned} \hat{X} - X &= (X + w^*) - X \\ &= w^*. \end{aligned} \tag{9}$$

Knowledge of w^*, in the example system, allows the author to remove the watermark from any image watermarked by the server by a simple matrix subtraction.

A simple-minded solution might be to disallow all-of-object changes, but a patient author can still build up knowledge of a collective watermark w^* using a sequence of changes that, when taken together, cover the object. Alternatively, an author could be prohibited from accessing the object twice in a row, but a determined author may still be able to piece information together from points that were not changed by intermediate authors.

To defeat this attack and other similar attacks based on examining the output of the server for a specially-constructed input, either

(i) it should not be feasible to compute w^* given X and $W(X, w^*)$, or
(ii) it should not be feasible to compute X given $W(X, w^*)$ and w^*.

Watermarking schemes that satisfy one or the other of these conditions are proposed by Depovere and Kalker [14] and Stern and Tillich [15]. In these schemes, a single detection key σ can be used to generate many different watermark patterns w^* using a one-way function. Each watermarked object is watermarked using the same σ, but a different w^*. This approach prevents an attacker from learning any information about σ even if he or she can learn w^*. Without knowledge of σ, an attacker cannot remove or otherwise tamper with watermarks created by the server. Investigation of how these types of schemes can be implemented in our architectures is a subject of ongoing research.

3.3. Authoring with layers

In this section, we will consider an architecture for authoring that does not require a server, trusted or otherwise. Consider a function $U(X^1, \ldots, X^m)$ that takes a collection of *layers* $X^1, \ldots, X^m$ and merges them into a single object X. A simple example is the function that overlays a collection of line-drawings on transparent backgrounds, producing an object containing every line from every drawing. For a suitable choice of $U(X^1, \ldots, X^m)$, we can arrange for an object $X = U(X^1, \ldots, X^m)$ to be manipulated by a collection of m authors, each making changes to one layer only.

Let each author i own a layer and maintain two versions of this layer: an unwatermarked layer X^i and a watermarked layer $\hat{X}^i = W(X^i, w^i)$, where $W(X, w)$ denotes watermarking an object X with a watermark w. The former is a secret of its author, and the latter is public. Of course the author need not embed his or her watermark in the public layer if he or she does not want to, making the public layer the same as the "private" layer, but this in no way affects the other authors' watermarks. Anyone knowing all of the public layers can compute an object $\hat{X} = U(\hat{X}^1, \ldots, \hat{X}^m)$.

To make a change to the object, an author i first makes the appropriate change to his or her private layer X^i. He or she then computes a new version of the public layer $\hat{X}^i$ corresponding to the new private layer, and publishes the new $\hat{X}^i$. The other authors may then recompute their copy of the merged object.

A rebel author may choose to create a rebel object by ignoring broadcasts from some particular author i. The rebel object thus produced will not contain the watermark w^i, and therefore author i cannot claim any contribution to the rebel object. This is unavoidable in this architecture, and it is debatable as to whether or not author i *should* be able to claim contribution to an object from which his or her contribution has been erased. Eliminating one author, however, does not affect the ability of the other authors to exhibit their watermarks in the rebel object.

Of course, it is not automatic that the watermarks in the $\hat{X}^i$s will survive the merging process for any arbitrary combination of watermarking and merging functions. We will give

an example in which there is a statistical expectation that the watermarks can be detected in the merged object, but we do not know of any way of guaranteeing this while still providing a useful merging function.

3.3.1. An example

We will describe a layered watermarking system for raster images, using the JAWS watermark of Kalker et al. [16], except that for simplicity of exposition we will not use translation invariance. While this watermark's stated purpose is broadcast monitoring rather than proof-of-ownership, it is a convenient example for our purposes. For simplicity, we will assume that the images are grey scale though it is easy to extend the procedure to colour images.

As described above, each author i maintains a private, unwatermarked $p \times q$ image (layer) X^i and a public, watermarked $p \times q$ image $\hat{X}^i$. These are both initialised to zero. Each author i also has a private $p \times q$ watermark pattern w^i with standard normal distribution (i.e., each element of w^i is randomly chosen from a normal distribution with mean zero and standard deviation one), as usual in JAWS.

Each author also maintains a copy of a $p \times q$ matrix Y with entries from $1, \ldots, m$, which is initialised randomly. To compute the merged, watermarked image $\hat{X} = U(\hat{X}^1, \ldots, \hat{X}^m)$, every author can compute

$$\hat{X}(x, y) = \hat{X}^{Y(x,y)}(x, y). \tag{10}$$

The authors do not need to agree on an initial Y since every author's layer is identical in the beginning. Even if the layers are not identical, choosing one is as good as choosing the next.

If an author i wishes to make a change to a set of pixel locations D, he or she makes the appropriate changes in X^i and computes $X' = U(\hat{X}^1, \ldots, X^i, \ldots, \hat{X}^m)$, that is,

$$X'(x, y) = \begin{cases} X^i(x, y), & \text{if } (x, y) \in D, \\ \hat{X}^{Y(x,y)}(x, y), & \text{otherwise.} \end{cases} \tag{11}$$

The author computes the perceptual mask λ of X' as usual in JAWS, that is,

$$\lambda = \frac{1}{9} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} * X', \tag{12}$$

where "$*$" denotes convolution and computes the watermarked values for all the pixel locations $(x, y) \in D$ using the usual JAWS embedding function

$$\hat{X}^i(x, y) = X^i(x, y) + \alpha\lambda(x, y)w^i(x, y), \tag{13}$$

where α is a global scaling parameter. The other pixels of $\hat{X}^i$ are left unchanged.

The other authors are then informed of the change by a broadcast of D by author i. Each author then updates his or her copy of Y by setting

$$Y(x, y) = i \quad \forall (x, y) \in D, \tag{14}$$

leaving other entries in Y unchanged. Note that an author can also choose a rebel Y, thus creating a rebel object, but this object still contains the other authors' watermarks unless one author has been targeted for removal as described in the introduction to this section.

The resulting watermark is a kind of segmented watermark. If the watermark detector has access to Y, it can partially invert the merging function to obtain a set of layers, each containing the pixels watermarked by a particular author (with zeros where the contents of that layer are unknown).

Since the watermark patterns are mutually uncorrelated, however, it is possible for a detector to test for a given watermark pattern without knowledge of Y, using the normal JAWS detection algorithm. To test an image Z for the presence of a watermark w, we filter Z with

$$Z' = \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} * Z, \tag{15}$$

and then compute the correlation of Z' with w. Even though only some of the pixels of $\hat{X}$ come from the layer containing a watermark pattern w^i, the correlation of $\hat{X}$ with w^i is still high, as

$$\begin{aligned} w^i \cdot \hat{X} &= \sum_{x=1}^{p} \sum_{y=1}^{q} w^i(x, y)\left(X(x, y) + \alpha\lambda(x, y)w^{Y(x,y)}(x, y)\right) \\ &= \sum_{x=1}^{p} \sum_{y=1}^{q} w^i(x, y)X(x, y) + \sum_{Y(x,y)=i} \alpha\lambda(x, y)\left(w^i(x, y)\right)^2 \\ &\quad + \sum_{Y(x,y)\neq i} \alpha\lambda(x, y)w^{Y(x,y)}(x, y)w^i(x, y) \\ &\approx \sum_{Y(x,y)=i} \alpha\lambda(x, y)\left(w^i(x, y)\right)^2 \\ &> 0 \end{aligned} \tag{16}$$

since the expected correlation of w^i with the original image and the other watermarks is zero. This is the same idea as used by the asymmetric watermark of Hartung and Girod [17]; in fact, Eggers et al. [18] suggest that Hartung and Girod's method might be more useful as a multiple watermark than as an asymmetric one.

3.3.2. Limitations

This system does not guarantee that an author's watermark will be detectable in the final object, since it is entirely possible that an author's contribution will be obliterated by later authors overwriting that author's contribution. Consider, for example, the case where some director makes a rough sketch of a scene he or she wants drawn, then other artists move in to fill out the details, obliterating the sketch. No watermark can survive a complete redrawing of the image (whether or not the new image is semantically related to the old one), so it is difficult to see how any useful merging function could preserve watermarks in such obliterated contributions.

3.4. Authoring with instructions

A special case of the layered authoring system described in the previous section is the case where the object is created by authors who issue streams of instructions to make changes to the object, such as "draw a line here," "make this pixel blue," and so forth. The final object can be thought of as the interleaving (merging) of the individual instruction streams (layers) of each author. The Network Text Editor of Handley and Crowcroft [19], for example, uses a similar architecture. Clearly, this model is well suited to formats that represent objects by a sequence of rendering primitives, such as text or vector graphics, rather than formats that represent objects by raster data.

The system is initialised by each author creating an empty object. Let each author i have a secret watermark w^i and let $X^i = X_1^i, X_2^i, \ldots$ denote the stream of instructions issued by author i.

To issue an instruction X_j^i to make a change to the object, an author i computes a watermarked version of the instruction $\hat{X}_j^i = W(X_j^i, w^i)$, and broadcasts $\hat{X}_j^i$ to all of the authors, who append this to their local copy of the object. The unwatermarked version X_j^i is discarded (though there is no reason author i could not keep it if he or she wanted to).

As in the layered system, an author can choose to ignore the broadcasts of other authors and create a rebel object with an eliminated author. In this architecture, this is equivalent to an outsider who crops instructions from the final object, which is unavoidable in the general watermarking model.

3.4.1. On instruction complexity

Depending on the complexity of the instructions used, it may or may not be possible to embed an entire watermark into a single instruction. Solachidis et al., for example, propose a watermark for polylines [20] that could be used to embed a whole watermark into an instruction to draw a polyline or similar complex shape.

However, multimedia languages typically make use of many much simpler instructions such as "put text here" or "draw a line" that have only one or two points available for embedding watermark information. In this case, the watermark information needs to be distributed over many instructions. Let the watermark pattern w^i of a participant i be made up of a sequence of n components $w_1^i, \ldots, w_n^i$, and let $f(X_j, w_l^i)$ be a function for embedding a watermark component w_l^i into an instruction X_j. Let $\tau(\cdot)$ be some mapping of instructions to the integers $1, \ldots, n$. Then an author i can embed a watermark component in each instruction X_j by

$$\hat{X}_j = f\left(X_j, w^i_{\tau(X_j)}\right). \tag{17}$$

A simple choice for $\tau(\cdot)$ would be to number instructions according to the order in which they were issued, that is,

$$\tau(X_j) = j \bmod n + 1. \tag{18}$$

However, this is a poor choice since the instructions may, in general, be reordered without affecting the way the object is rendered. A more robust choice is to determine $\tau(X_j)$ by some property of X_j that cannot be changed so easily, such as its position in the drawing space. We will give an example of such a function in Section 3.4.3.

3.4.2. On the output format

The raw instruction streams issued by authors are unlikely to make an attractive format for distribution. We can expect that the raw instruction streams will contain many instructions that make corrections to earlier instructions. Distributing such redundant instructions is not only inefficient, but may also be unimplementable on output devices, such as printers, that cannot alter the effect of any instructions once they have been carried out.

We can therefore expect some degree of postprocessing on the instruction stream to put it into an acceptable format for distribution. This may mean removing redundant instructions, or combining a series of corrective instructions into a single instruction, or radical format conversions, such as rasterisation. It is inevitable that watermark information will be lost in the process, and possibly whole contributions obliterated as in the layered case. A radical format conversion may destroy the watermark completely; this is true of any watermark, not just ones created by instruction streams.

3.4.3. An example

We will describe a system for authoring two-dimensional vector graphics where authors may draw lines, circles, polygons, and so forth. We will use a very simple watermark similar to the one suggested by Koh and Chen [21], but ours will be robust against reordering of drawing elements. We assume that every drawing primitive is associated with one or more points in the plane, such as the end-points of a line, the centre of a circle, the vertices of a polygon, and so forth, and consider each point v_j individually.

We will assume that all points lie in the first quadrant of the Cartesian plane, that is, that the origin is at the bottom-left of the drawing space. We associate a point v_j with a bin $b_{\tau(v_j)}$ by dividing the drawing space into n sectors using n radial lines emanating from the origin at equally-spaced angles, that is, let $(r(v_j), \theta(v_j))$ denote the polar coordinates of a point v_j and set

$$\tau(v_j) = \left\lfloor \frac{2n}{\pi}\theta(v_j) \right\rfloor + 1. \tag{19}$$

Let $w^i = w_1^i, \ldots, w_n^i$ denote the watermark of author i, where each w_j^i is drawn from a standard normal distribution. We compute the watermarked version $\hat{v}_j$ of a point v_j by

$$\begin{aligned} r(\hat{v}_j) &= r(v_j) + \alpha w^i_{\tau(v_j)}, \\ \theta(\hat{v}_j) &= \theta(v_j) \end{aligned} \tag{20}$$

for some agreed global scaling parameter α, that is, the point is moved further away from or closer to the origin by an amount proportional to $w^i_{\tau(v_j)}$.

As for the general layered watermark, the watermark resulting from a collection of collaborating authors is a segmented watermark and can be detected by breaking the instruction stream into the streams contributed by each author.

However, it is possible, and more convenient, to detect the individual watermark as if the object contained a composite watermark as in the layered example. This can be done for a watermark w^i using the correlation of the points distances from the origin

$$r = r(v_1), r(v_2), \ldots, r(v_t) \tag{21}$$

with the vector of corresponding watermark components

$$\tilde{w}^i = w^i_{\tau(v_1)}, w^i_{\tau(v_2)}, \ldots, w^i_{\tau(v_t)}. \tag{22}$$

If the correlation is high, we report that the watermark is present, otherwise we report that it is not.

4. FINGERPRINTING

For simplicity, in this section we will assume that proof-of-ownership is not an issue. Suppose, for example, that the authors are employees of a company and do not own the intellectual property in their work. However, leaking a copy of their work prior to the official company publication may compromise the company's intellectual property, and the company might be interested in learning who made the leak.

In the watermarking solution to this problem, each legitimate copy of the object is embedded with a distinct watermark, called a *fingerprint*, that identifies the owner of that copy. If one of the legitimate owners makes an illegitimate copy, and this copy is found by investigators, this copy can be traced to the owner using the fingerprint in it.

As in the proof-of-ownership case, it is easy to see that a dishonest insider in possession of the unwatermarked original can circumvent the tracing protocol. In this section, we will consider how the architectures described in Section 3 can be adapted to solve the fingerprinting problem in the presence of dishonest insiders.

In order to implement fingerprinting, there are two basic changes that need to be made to the proof-of-ownership systems described in the previous sections:

(i) watermarks (i.e., fingerprints) are not known by the owner of that watermark;
(ii) each author should have a distinct (fingerprinted) version of the object.

In general, fingerprints may be chosen to have various useful properties, such as collusion security. For simplicity and due to space considerations, we will not consider such properties here. We require only that each author receives a version of the work containing a distinct watermark.

4.1. With a server

Implementing fingerprinting is straightforward using a server. The server simply chooses a distinct watermark w^i for each author i known only to the server, and embeds w^i (only) into any objects that are transmitted to author i. If author i leaks a copy of the object, the author can be traced by the presence of w^i in the leaked copy.

4.2. Without a server

Without a server, it is necessary for every author i to choose a distinct fingerprint $w^{i,j}$ for every coauthor j. When making a change to the object, author i must generate a version of the change for each fingerprint $w^{i,j}$ and transmit this version to author j over a private channel instead of using the broadcast channel as before. In this way, each author i has a copy of the object containing a collection of $m-1$ fingerprints $w^{1,i}, w^{2,i}, \ldots$, and so forth, uniquely identifying that author's copy. Assuming that the watermark in use is separable, any author j who leaks a copy can be traced by the presence in the leaked copy of any one of $w^{i,j}$ for some other author i.

Since each fingerprint $w^{i,j}$ is known by author i, it may be possible for author i to attempt to frame author j by leaking a copy of the object containing $w^{i,j}$. A simple solution would be to use majority voting in the tracing algorithm, and require that the majority of fingerprints found in a leaked copy correspond to the accused author. Since a dishonest author i's object also contains the $m-1$ fingerprints assigned to i by the other authors, this test would correctly identify i as the leaker. However, it is still possible for a majority of authors acting in collusion to frame an author in the minority.

A more robust, but more complicated, solution is to use *asymmetric fingerprinting* [22] (also known as a *buyer-seller protocol* [8]). In these protocols, the fingerprinter (author i in the above) and the fingerprintee (author j) interact during the fingerprinting process in such a way that the fingerprinter cannot obtain a copy of the fingerprinted object. Every time author i makes a change to the object, he or she must execute the asymmetric fingerprinting protocol with every other author j, using fingerprint $w^{i,j}$.

5. DISCUSSION

5.1. Security

Our systems permit authors to access only watermarked versions of the object they are working on, and hence an insider wishing to deny the contribution of the other authors, or leak an illegitimate copy of the object, would ideally be in the same position as an outsider attempting to do the same. The systems described above do not quite meet this ideal, since

(i) insiders see many different objects (being different versions of the object-in-progress) containing the same watermark, potentially giving insiders greater opportunity for attacks that attempt to estimate the watermark;
(ii) insiders generally know the source of any change, and therefore which pixels or instructions are watermarked by which author, and can use this knowledge to target a particular watermark.

Of course, if the watermark being used was perfectly secure (in the sense that it is unremovable without unacceptably degrading the object), this extra knowledge should not matter, but on current watermarking technology, this seems a little optimistic.

5.2. Collusions

A group of dishonest insiders may pool their information in an attempt to defeat the watermarks of insiders from outside the colluding group. This sort of attack is commonly considered in fingerprinting systems, where the colluders are a collection of outsiders. Here, such colluders may be insiders as well, but as we have observed in the previous section, inside colluders are in the same position as outside colluders since the insiders have access only to a fingerprinted version of the object. Hence we expect that fingerprinting algorithms that are secure against outsider collusions should also be secure against insider collusions.

In the proof-of-ownership case, all authors have exactly the same information about the original object and about other authors' watermarks (which, ideally, is no information at all). Hence a collusion will not reveal any information to the colluders other than the colluders' own watermarks, and what they already knew by virtue of their being insiders. Since all the watermarks are independently chosen and embedded, the colluders have not improved their chances of defeating the noncolluders' watermarks over an insider acting alone.

6. CONCLUSION

We have introduced the problem of protecting the intellectual property rights of multimedia content owners where potentially malicious insiders have access to the content before publication. Conventional watermarking solutions to the proof-of-ownership problem cannot resolve intellectual property disputes that arise prior to publication, and conventional fingerprinting solutions cannot trace leakers who leak prepublication versions of content, since the adversary in such situations has access to an unwatermarked version of the content.

We have proposed several possible architectures for watermarking with dishonest insiders, in which insiders have access only to a watermarked version of the object that they are working on. Hence, an insider is in not much better a position to defeat the watermark than an outsider. If watermarks had perfect security, insiders would not be in a better position at all.

Our systems cannot be guaranteed to successfully resolve any particular intellectual property dispute in a collaborative environment, and we do not think that any currently known (or even foreseen) computer system can, since

(i) computers cannot make artistic judgements on the worth of any particular contribution;

(ii) realistic authors will generally use out-of-band communications such as face-to-face meetings to exchange ideas;

(iii) we cannot watermark the semantics of multimedia content.

However, the architectures proposed in this paper provide a basis for the development of systems that can assist in resolving intellectual property disputes between collaborators by providing at least some evidence of what happened prior to publication, and we are hopeful that further research can overcome at least some of the limitations we have noted.

REFERENCES

[1] H. Guo and N. D. Georganas, "A novel approach to digital image watermarking based on a generalized secret sharing scheme," *Multimedia Systems*, vol. 9, no. 3, pp. 249–260, 2003.

[2] N. P. Sheppard, R. Safavi-Naini, and P. Ogunbona, "On multiple watermarking," in *Workshop on Security and Multimedia at ACM Multimedia*, pp. 3–6, Ottawa, Ont, Canada, 2001.

[3] G. C. Langelaar, I. Setyawan, and R. L. Lagendijk, "Watermarking digital image and video data. A state-of-the-art overview," *IEEE Signal Processing Magazine*, vol. 17, no. 5, pp. 20–46, 2000.

[4] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen, *HTTP extensions for distributed authoring – WEBDAV*, RFC 2518, Internet Society, 1999.

[5] J. Fridrich, A. C. Baldoza, and R. Simard, "Robust digital watermarking based on key-dependent basis functions," in *Proc. 2nd Information Hiding Workshop*, pp. 143–157, Portland,Ore, USA, April 1998.

[6] J. Fridrich, "Key-dependent random image transforms and their applications in image watermarking," in *International Conference on Imaging Science, Systems, and Technology*, pp. 237–243, Las Vegas, Nev, USA, June 1999.

[7] J.-C. Yen, "Watermarks embedded in the permuted image," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 53–56, Sydney, Australia, May 2001.

[8] N. Memon and P. W. Wong, "A buyer-seller watermarking protocol," *IEEE Trans. Image Processing*, vol. 10, no. 4, pp. 643–649, 2001.

[9] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, R. A. DeMillo, D. Dobkin, A. Jones, and R. Lipton, Eds., pp. 169–179, Academic Press, New York, NY, USA, 1978.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[11] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "A secure, robust watermark for multimedia," in *Proc. 1st International Workshop on Information Hiding*, pp. 185–206, Cambridge, UK, 1996.

[12] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313–336, 1996.

[13] I. Pitas, "A method for signature casting on digital images," in *Proc. IEEE International Conference on Image Processing*, pp. 215–218, Lausanne, Switzerland, September 1996.

[14] G. Depovere and T. Kalker, "Secret key watermarking with changing keys," in *IEEE International Conference on Image Processing*, pp. 427–429, Vancouver, BC, Canada, September 2000.

[15] J. Stern and J.-P. Tillich, "Automatic detection of a watermarked document using a private key," in *Proc. 4th International Workshop on Information Hiding*, pp. 258–272, Pittsburgh, Pa, USA, April 2001.

[16] 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*, pp. 103–122, San Jose, Calif, USA, January 1999.

[17] F. Hartung and B. Girod, "Fast public-key watermarking of compressed video," in *IEEE International Conference on Image Processing*, pp. 528–531, Santa Barbara, Calif, USA, October 1997.

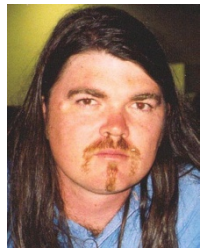
[18] J. J. Eggers, J. K. Su, and B. Girod, "Asymmetric watermarking schemes," in *Sicherheit in Netzen und Medienströmen*, M. Schumacher and R. Steinmetz, Eds., pp. 124–133, Berlin, Germany, 2000.

[19] M. Handley and J. Crowcroft, "Network Text Editor (NTE): a scalable shared text editor for the MBone," in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 197–208, Cannes, France, September 1997.

[20] V. Solachidis, N. Nikolaidis, and I. Pitas, "Fourier descriptors watermarking of vector graphics images," in *IEEE International Conference on Image Processing*, pp. 9–12, Vancouver, BC, Canada, September 2000.

[21] B. Koh and T. Chen, "Progressive browsing of 3D models," in *Proc. IEEE 3rd Workshop on Multimedia Signal Processing*, pp. 71–76, Copenhagen, Denmark, September 1999.

[22] B. Pfitzmann and M. Schunter, "Asymmetric fingerprinting," in *EUROCRYPT '96*, pp. 84–95, Springer-Verlag, Berlin, Germany, 1996.



Nicholas Paul Sheppard received Bachelor's degrees in computer systems engineering and pure mathematics from the University of Queensland in 1996, and a Ph.D. in computer science from the University of Sydney in 2001. He is currently a Research Fellow in multimedia security at the University of Wollongong.

Reihaneh Safavi-Naini is a Professor of computer science at the University of Wollongong. She holds a Ph.D. in electrical and computer engineering from University of Waterloo in Canada. Her research interests include cryptography, computer and communication security, multimedia security, and digital right management.

Philip Ogunbona received the B.S. (with honors) in electronic and electrical engineering from the University of Ife, Nigeria, and the Ph.D. in electrical engineering from Imperial College of Science, Technology and Medicine, University of London. From 1990 to 1998, he was on the academic staff of the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong. In 1998, he joined Motorola Australian Research Centre, where he was responsible for developing imaging algorithms. He became Manager of the Digital Media Collection and Management Lab, and directed research into multimedia content management for mobile systems and the home. He is now a Professor in the School of Information Technology and Computer Science, University of Wollongong. His research interests include multimedia signal processing, multimedia content management, multimedia security, video surveillance, and colour processing.