Enabling Hierarchical and Bulk-Distribution for Watermarked Content

Germano Caronni and Christoph Schuba
Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303, USA
gec@acm.org, Christoph.Schuba@Sun.COM

Abstract

One of the solutions to deter copyright violations consists of embedding hard to detect watermarks in digital media. Current developments have focused on how to embed watermarks, and on one-to-one exchanges on how to securely convey tagged data to the end consumer. Assuming a large customer base or predistributed media, it may become prohibitively expensive or time consuming to tag each separate copy of data individually before it is delivered to the customer.

We present two mechanisms that allow the preparation and distribution of tagged data in a more scalable way than existing approaches. The first one, called hierarchical tagging, is preferable for on-line multi-level distribution, where producer and consumer are not in direct contact, but intermediate agents provide distribution channels and sales platforms. The second method is well suited to pre-produced bulk-media distribution (such as through CDROMs or DVDs), with only small amounts of on-line information being transferred to each consumer. We call it bulk-tagging.

1 Introduction

The science of inserting hidden marks in digital data has advanced from steganographic uses [Kah96] and first primitive watermarking approaches [TNM90], [Car95] to a field of much research interest. Many ways have been proposed on how to insert tags (aka watermarks) into digital media such as audio and video streams, digital images, text of various formats, source code [ST98], and even program binaries [SHKQ99]. Distribution channels were usually limited to the one-to-one scenario, with few notable exceptions such as Watercasting [BPC99]), where watermarking becomes a feature of multicast communications. Delivery and dispute resolution protocols with different security properties were designed, such as described in [PW97].

There exists, however, a fundamental problem. If each individual copy needs to be tagged separately and with a unique watermark, the distribution of tagged media is limited by the computational capacity of the watermark creator. Watermarks can either be pre-computed (and stored at the distributor-side) or done on the fly, right before data is to be distributed. Both approaches do not scale well to larger customer bases because a different watermark has to be computed and inserted into the data for each new deliverable copy. Because the insertion of watermarks is not an inherently sequential process, a solution to this problem is to allow a set of distributors to insert watermarks independently from each other. In this approach, all distributors are trusted to correctly report their sales proceedings to the owner of the data. Similarly, pre-produced bulk-media could be created in a parallel fashion, but in both cases the trust model is hard to accept. The former DivX product pursued a different approach, where devices were inserting watermarks into the video stream during actual playback. Relying on trusted hardware on the consumer side, this approach does not work in a software-only environment. There, programs executing the tagging algorithm are under control of the end-user on whose computer the operation is being performed.

In the next few sections, the paper describes two schemes that allow the process of creating and distributing tagged media to scale to large quantities. In the first approach, named *hierarchical tagging*, the online distribution scenario is extended by allowing a hierarchy of distributors to provide media to end customers. Here, the media is tagged at each level in the hierarchy, introducing multiple layers of tags in the same data.

The second approach (*bulk tagging*) targets both the off-line distribution of tagged media (such as via CDs or DVDs) and the online bulk distribution (e.g., via mirror FTP servers). In this second approach, the same set of preprocessed media is distributed to every customer and the receiver subsequently needs to acquire a set of cryptographic keys (online, or packaged together with the media) to access the uniquely tagged data.

Published in Proceedings of the 17^{th} Annual Computer Security Applications Conference (ACSAC), New Orleans, December 2001

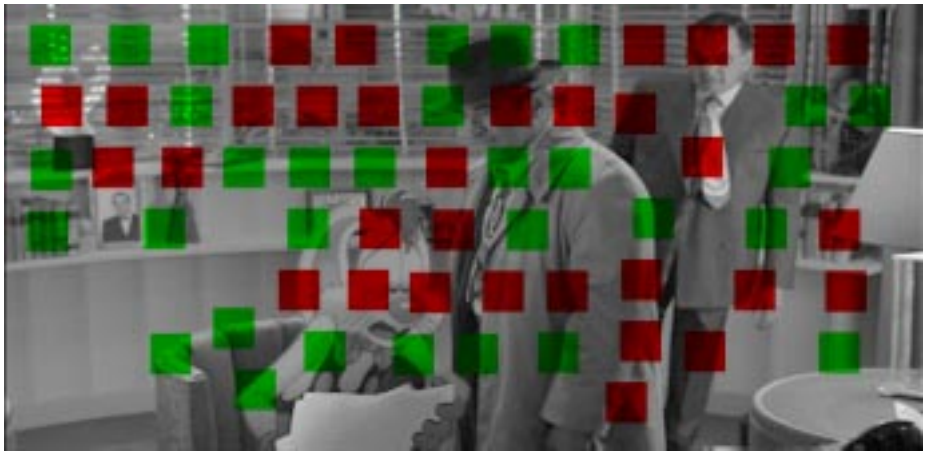

Figure 1. Image with all potential tags superimposed, tag size is 16x16.

Figure 2. Tagged image (with exaggerated markings).

2 Background Material

Before pursuing the two approaches to scale tagging to larger distributions, we need to shortly revisit an actual watermarking mechanism, and mention other fundamentals. While more sophisticated tagging approaches exist, we will use an old scheme of image tagging [Car95] for our examples. It is easy to understand and to implement, and thus facilitated the creation of a prototype to test and validate our ideas.

The tagging scheme consists of choosing a series of rectangular areas in an image, and then changing the brightness of some of those areas. The choice whether a given location is selected to carry a tag depends on the variance of the brightness around that location, and on some pseudo-random factors. Effectively, a different random bit string of a few hundred bits in length is encoded in each image. Each bit string is specific to one consumer. Figure 1 shows potential tagging locations, and Figure 2 an exaggeratedly tagged image. By making the location and nature of the individual tags depend on keying information, the scheme tries to avoid the problems of achieving security through obscurity.

Interestingly enough, this scheme is quite resistant to contemporary attacks such as the use of StirMark [PAK98], where a geometric, color space, and dithering distortion still yields acceptable tag recovery. The following Table 1 outlines[1] this for the sample image depicted above. It does

[1]Tag Mode indicates both tag size in pixels and the marking intensity; PSNR is the signal to noise ratio (in dB) of the luminance. Correlation is done the same way as in [Car95]. StirMark #1 used the following parameters: -i2 -o.7 -d3 -n6 -r1, and StirMark #2 used -i1.5 -o.5 -d3 -n6 -r1, resulting in about 20% less geometrical distortion, see `http://www.cl.cam.ac.uk/~mgk25/stirmark` for code.

however succumb to other attacks, and should thus not be considered secure.

The exemplary scheme can be used for hierarchical and bulk-tagging applications. For hierarchical tagging, it is necessary that different layers of watermarks do not (or only minimally) interfere with each other. This is usually a basic property of watermarking for copyright protection purposes because watermarks are by design supposed to be hard to detect and hard to interfere with. For a watermarking scheme that is to be used in a bulk-tagging environment, the tags or customer-dependent distinct markings must be separable into building blocks such that the whole can be composed of independent parts. The sections below will provide more details for both these requirements.

3 Hierarchical Tagging

Watermarks for digital media are supposed to be hard to detect and resilient to modification. Their modification may originate from random reproduction errors (e.g., subsampling of the color space or reshaping and resizing of the picture). Alterations may also be caused by malicious users who try to erase the watermarks by introducing noise or by performing any of a number of geometric or content transformations on the data. As seen in the previous section, an excellent example for this is StirMark [PAK98], a tool to modify images with the purpose of making the recovery of tags difficult.

As a direct consequence of abovementioned desirable properties, one may contemplate mixing multiple layers of watermarks into the same document. Not too surprisingly, this works quite well, mainly because of the redundancy and non-interference that are some of the design goals of good watermarking schemes. One can draw a direct analogy to direct sequence spread spectrum radios where the same frequency range is used for multiple transmissions (potentially below the ambient noise level) at the same time. As long as watermarks are placed in sufficiently randomized data portions and as long as the interference of multiple layers of watermarks in the same data portion does not overly reduce the media quality, one can use this approach to enable a hierarchy for data distribution. See also [CKLS96] for an application of spread spectrum techniques to watermarking.

In hierarchical tagging, the situation is not quite the same as when independent tags are introduced into media. For one, the upper-level distributors need to store less information in the data than the sub-distributors, because there usually exist markedly fewer distributors than end customers. Secondly, the upper layer distributor can take into account the expected tagging behavior of lower layer distributors, and use this knowledge to make his watermark even more resistant to modification. He may also advise the sub-distributors on how they should place their marks, to optimize recoverability.

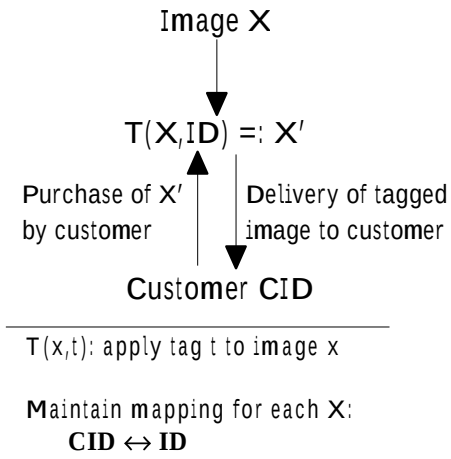

Figure 3. Traditional approach to watermarking. Image X is tagged with mark ID. The tagged image X' is acquired by customer CID.

As an example, consider an artist who produces high-quality digital artwork in the form of pictures. Figure 3 illustrates the traditional approach to watermarking the picture. The artwork (Image X) is tagged with a watermark that encodes an identifier ID: $T(X, ID)$. We call the resulting image X'. When a customer (identified by customer id CID) acquires the rights to the image, he receives the tagged image X'. The distributor records the mapping between CID and ID. If bootlegged copies of X' are found, the watermark ID can be retrieved. The mapping $CID \leftrightarrow ID$ then reveals whose copy was bootlegged.

In hierarchical tagging, however, the artist may choose to offer his creations to customers through several distributors, but not until after inserting a different watermark in each of his pictures. The distributors could in turn insert watermarks for each sub-distributor they have. Sub-distributors could tag the image before delivery to individual customers or further distributors. The process is illustrated in Figure 4. Image X is tagged with the identifiers of three distributors D_1, D_2, and D_3. Distributor D_1 uses three sub-distributors (D_4, D_5, and D_6.) Eventually, one of the distributors, say D_n interacts with the final customer. The final distributor again maintains the mapping $ID \leftrightarrow CID$. The image X''' that the customer receives contains all watermarks that were added for the distributors on the path from the creator of the image X to the customer himself. If bootlegged copies of X''' are found, all of the watermarks pertaining to D_1, D_5, ..., D_n, ID can be retrieved. This data reveals whose copy was bootlegged.

			Tagged Only		StirMark #1				StirMark #2			
Tag	Mode	#Tags	PSNR	Correl.	PSNR	Correl.	loss(%)		PSNR	Correl.	loss(%)	
16x16	1.4%	63	47.38	.9989	31.15	.9605	4	6.3%	32.88	.9736	2	3.1%
16x16	1.0%	63	49.09	.9992	31.18	.9608	7	11.1%	32.94	.9741	6	9.5%
12x12	1.4%	106	47.32	.9989	31.14	.9604	14	13.2%	32.87	.9737	13	12.2%
12x12	1.0%	106	49.05	.9992	31.17	.9608	19	17.9%	32.93	.9741	15	14.1%
10x10	1.4%	152	47.73	.9990	31.16	.9606	28	18.4%	32.90	.9739	19	12.5%
10x10	1.0%	152	49.93	.9993	31.19	.9609	34	22.3%	32.95	.9741	28	18.4%
8x8	1.4%	209	48.50	.9991	31.17	.9608	52	24.8%	32.92	.9740	39	18.6%
8x8	1.0%	209	50.00	.9994	31.20	.9610	60	28.7%	32.96	.9743	52	24.8%
6x6	1.4%	324	49.00	.9992	31.17	.9608	74	22.8%	32.93	.9740	58	17.9%
6x6	1.0%	324	50.39	.9994	31.19	.9610	87	26.8%	32.96	.9742	71	21.9%
4x4	1.4%	537	49.63	.9993	31.18	.9609	175	32.5%	32.94	.9741	155	28.8%
4x4	1.0%	537	50.88	.9995	31.20	.9610	192	35.7%	32.97	.9743	167	31.0%

Table 1. Resilience of chosen watermark algorithm to image modification via StirMark

Figure 5. Image with three layers of tags in different sizes.

Figure 5 contains a picture that has been (exaggeratedly) marked three times. Even when normal markings are introduced, the quality of the image is reduced. Table 2 identifies how much tagging information is lost through subsequent taggings and other transforms.

Figure 6 (left half) shows an enlarged spot in the image, where the placement of all three layers of tags is visible. One can clearly make out that different sizes of tags have been applied (16x16 for the first layer, 12x12 for the second layer, and 8x8 for the third layer). The intensity of the markings also increases from layer to layer, with 1.0% on the first, 1.2% on the second, and 1.4% on the third layer. Each layer also adds 0.5% of noise. Subjective measurement of the quality degradation is hard to express: The images become progressively more grainy, and the jpeg images appear slightly blocky. The quality of all images appears to be very good, or at least good. The right side of Figure 6 shows the same spot (in the normally marked image) as a difference from the original image.

While in principle this hierarchical process can be repeated many times, in practice more than a few layers of watermarks tend to reduce the data quality to a state that their presence can interfere with the purported use of the data. Experiments similar to the example above show that more than a few layers of watermarks tend to make digital images grainy. Some experiments assuming the knowledge of the algorithm and all its parameters except, e.g., their location, have shown a quality degradation of about 0.0003 to 0.0009 per tagging iteration. The degradation is measured as the difference of the correlation coefficient of the image tagged once versus the image tagged twice, as compared to the original image. A loss of about 3-4% of the original tags is observed per iteration. After about the fifth iteration, an observer's subjective image quality begins to suffer.

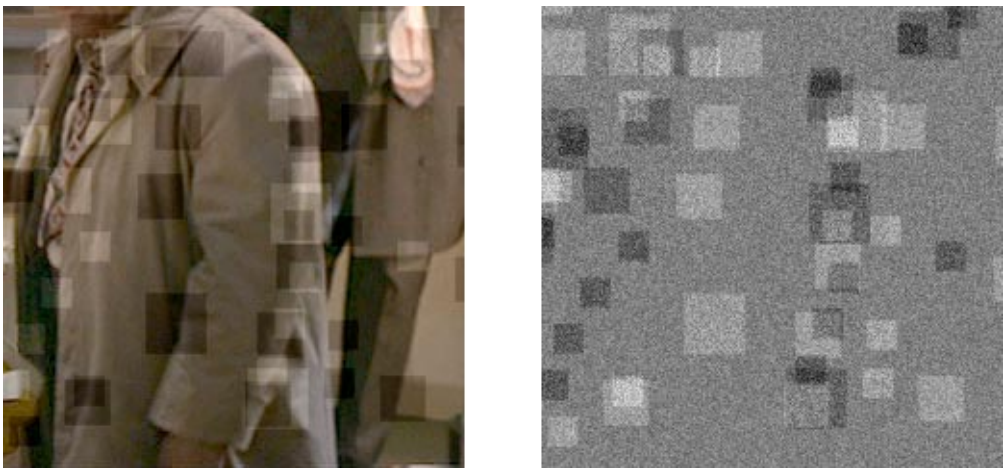

Figure 6. Extract of exaggerated marked image, and difference of normally marked image against the original.

	# Tags	Compared to Original PSNR	Correl.	Tags lost Level 1		Level 2		Level 3	
JPG-compressed original (Q=45)		39.88	.9954						
L1: Level 1 tags introduced	63	49.09	.9992						
L2: L1 and level 2 tags	107	45.98	.9985	4	6.3%				
L3: L1, L2 and level 3 tags	214	43.98	.9976	3	4.8%	6	5.6%		
JPG-compressed image L3 (Q=45)		38.86	.9938	2	3.2%	6	5.6%	1	0.5%
StirMark #1 on image L3		31.02	.9593	9	14.2%	20	18.6%	41	19.2%

Table 2. Tagging information loss through multiple layers

If any one of the distributors in the hierarchy (e.g., distributors or sub-distributors in the previous example) give away images without reporting their distribution, it can be shown that the images were those handed to the suspected distributor, and not to another one. It is expected that there is usually only a small fan-out of sub-distributors at each level, thus tags can easily distinguish between them. The problem of differentiating between various tagged images becomes exponentially more difficult if the number of separately tagged copies increases. An additional concern is that a distributor can frame one of their sub-distributors, unless special delivery protocols (such as [PW97]) are used.

While multi-layer tagging has the nice property that the computing power necessary to tag data can be delegated to distributors and sub-distributors, the requirement for online channels for the media distribution remains. Furthermore, every distributor must be able to create and insert a new set of watermarks, which may not be desirable or practical. Bulk-tagging, as explained in the following section, removes this requirement by shifting some of the computational requirements to the final customer.

4 Bulk-Tagging

The idea behind bulk tagging is to have the distributor create a single copy of data that can be given to a large set of customers without loosing the ability to assign individual tags to different customers. We will illustrate one such scheme, using the tagging mechanism presented in Section 2.

Up to this point, a distributor had to tag every copy of the data separately. Each single copy had then to be delivered to the customer, over a secure[2] channel. There was no requirement for the customer to preprocess the data once it was received. With bulk-tagging, the distributor creates multiple, tagged versions of the data. He then hides their contents through cryptographic techniques, and distributes all of them encoded as a single dataset. Each customer receives the same data set, but has to obtain an individual set of keys. Before the customer can access the tagged im-

[2]Otherwise somebody might intercept the data, abuse it, and the original customer would be blamed instead.

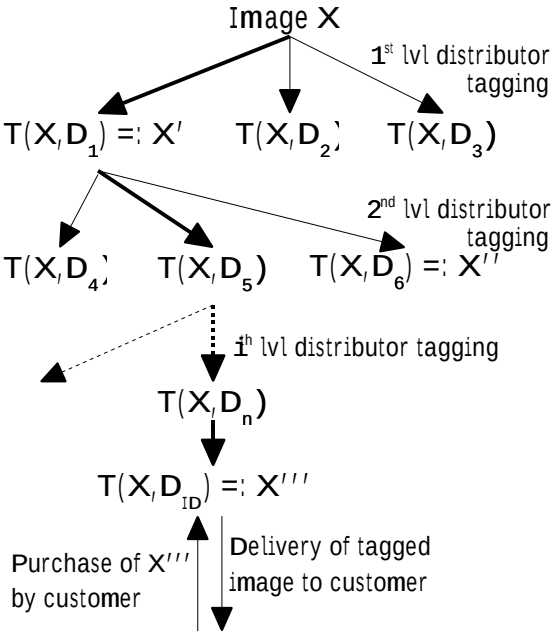

Figure 4. Distributed tagging. Image X is tagged repeatedly and differently before distribution to either sub-distributors, or to end customers.

age, he must perform some preprocessing. Thus he retrieves only the tagged data that is meant for him.

The key differences that distinguish bulk-tagging from previous approaches are that the distributor can create the tags and the distributable complete image exactly once, and that the customer is required to perform a precomputation before the data is intelligible. The preprocessing step at the client side is often an acceptable requirement which in many cases the customer won't even know about. That is because the computation can be hidden in the customer equipment that is used for further processing the data, e.g, viewing the picture on a screen.

Such a bulk-tagging scheme must offer certain features, to be a viable solution. First of all, a bulk-tagged data set, decoded for a particular customer, must offer similar content-quality as if tagged and delivered individually. As an example, a decoded bulk-tagged image should have the same quality[3] as a traditionally tagged image. Secondly, the feature of bulk-tagging must be well adapted to the tagging mechanism such that data volume is not overly expanded (it is OK to triple the size for some image that can now be transferred via a CD, but an increase in size of a factor 1000 is unlikely to be acceptable). The bulk-tagging process should also avoid to introduce additional weaknesses to the tagging scheme. Receiving two bulk-tagged data sets should give an attacker no more opportunities as he would have with two differently tagged data sets. Lastly, bulk-tagging should also support the use of compression, to allow a reasonable use of available bandwidth or storage capacity. This relates to the data expansion requirement above, but is different in that it depends more on the bulk-tagged media format than on the method of tag introduction.

For example, with bulk tagging a digital image of size 5MB no longer needs to be individually tagged and distributed, but with some precomputation arrives at a size of 20MB. Every customer then receives the same data set of 20MB, either in an online or off-line fashion, e.g., over a multicast channel, a ftp mirror, or through the mail on a CDROM. The 20MB image data, however, is not viewable by any customer until he has talked to the distributor online, conveyed his registration information, and in turn has received a set of cryptographic keys (e.g., 1000 of them, representing about 16KB of data) that can then be used to decrypt enough of the distributed data set to arrive at a single, individually tagged image.

It should be made clear, that this does not mean that the program the user runs would in any way insert tags into the data. It only takes the keys the customer receives from the distributor, and uses those to decrypt specific instances of parts of the image. Depending on which keys he got, differently tagged instances become decryptable. Each customer receives a different set of decryption keys and therefore ends up with an image with a different set of tags. The effect is that every customer has an individually tagged image.

This scheme establishes an equivalence between keys and differently tagged versions of an image. If somebody were to give away his keys to somebody else, it is the same as if he were to give away the decoded image. No new vulnerabilities are introduced through this equivalence, i.e., bulk tagged images are in the end not different from traditional tagged images in that an attacker must be handled the same way. It can be considered disadvantageous though that copies of the image can now be distributed with less communication overhead, because transmitting the keys is sufficient.

A naive scheme to implement this approach would be to have one base image where all N spots containing tags are blacked out, together with a lists of data (one for the tag being present, and one for the tag being absent) detailing the position information and decryption key identifier. This would allow to produce an individually tagged image, while at most doubling the communication overhead, and maintaining $2 * N$ different keys. Because the decryption and recombination software runs under the auspices of cus-

[3] In fact, it can be of even better quality, as online processing requirements (available CPU and bandwidth) are no longer limiting factors.

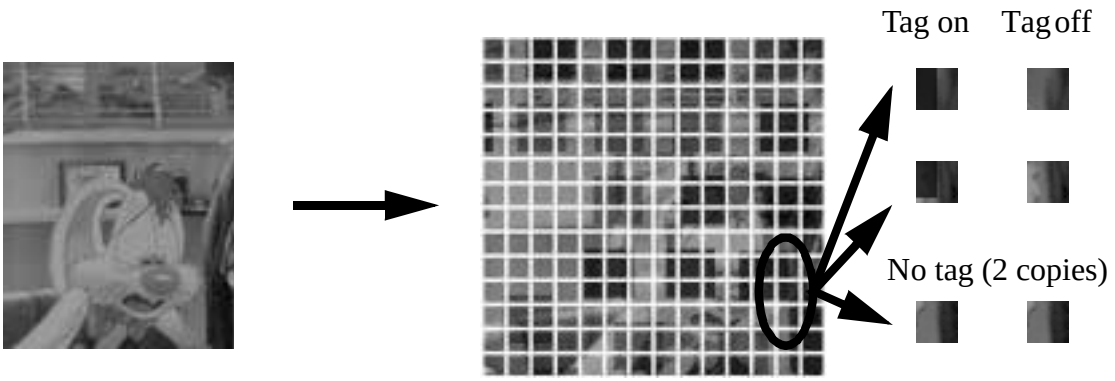

Figure 7. Process of segmenting an image, and producing different versions of cells. Cells containing the same tag are grouped together logically, but still encrypted with different keys to hide their relatedness.

tomers, they can observe the position information (disregarding the actual encoding of the watermarks, i.e., spatial or frequency domain). This would facilitate the removal of tags altogether, defying the original purpose of tagging.

However, less obvious forms of recombination can be imagined. Analogous to the k out of n share key threshold schemes [Sim92], customers can be required to decrypt multiple shares for the same image, followed by a mixing step to obtain usable pictures. Each share may contain several disabling or enabling watermark spots, and depending on which shares get mixed, the net result is a differently tagged picture. It would also be interesting to find a solution to this problem that works analogous to the one depicted in [Dro96]. The authors presented a way to create different messages depending on which k out of n slides you combined and looked through. So far, no way has been found to apply these two ideas to bulk tagging.

After these musings, there is however a simple approach to make bulk-tagged data delivery (at least in the case of images) feasible. Take an image with all its potential tag locations, as depicted in Figure 1. Independent of the tag locations, overlay a grid, favorably aligned to the cell boundary of a JPEG compressor, i.e. a 16x16 grid. For each cell in the grid, determine how many potential tag locations are contained therein. In our example, the minimum spacing of tag positions is 16 pixels, the complete tag size is 32x32 pixels, thus one cell will contain at most one potential tag, and one tag will be split over at most nine cells. Finally, save (in this case) two version of each cell (one where the contained tag, if any, is enabled, and one where it is not). The saved cells optionally are JPEG compressed, and the output of each version of each cell is encrypted with a different key. Figure 7 illustrates this process.

When a consumer registers with the producer, and asks for the keys to decrypt his individually tagged copy, the producer generates (depending on the customer ID) a random bit string. Here, each bit stands for one tag position, and indicates whether the tag is to be enabled or not. For each tag, the cells holding it are determined, and depending on the state of the tag (on or off) the keys for one of the versions of the cells is delivered. Thus, the consumer receives one key for each cell. In the case where no tag is contained in a cell, the version of the cell is chosen randomly. One should note that if there were only one version of cells containing no tags, potential attackers would have help in determining which cells hold tags. As it is, they can reconstruct the image, and then only run the same attacks that they could have run otherwise. This includes mixing different versions of images or parts thereof.

5 Related Work

As mentioned in Section 1, the creation of watermarked content is computationally expensive, and the secure delivery of tagged media to each recipient usually requires a separate transmission. Naturally this is only true when the watermarks are introduced to help enforcing ownership rights. A different kind of watermarking has the purpose to convey means of verifying the source or authenticity of the media [MW98]. In this context, watermarks depend on the source, and are the same for each recipient – scaling the distribution of watermarked media is not a problem, and hierarchical and bulk-tagging do not apply.

When watermarks are used to assure ownership rights, the media in question can either be distributed on-line or off-line. For off-line distribution, we are not aware of any existing solution. Bulk-tagging appears to be a first step in that direction. Watercasting [BPC99] is an interesting

scheme to tie the introduction of watermarks to the on-line distribution process of media itself. Intermediate entities (and transmission errors) in a multicast environment introduce destination-dependent data loss, in effect distributing the tagging effort and making it part of the data delivery process. The quality of the media the customer receives can vary, and network infrastructure components must interact with the distributor of the media to inform him of whom has received what subset of data.

Interestingly enough, Schneier [Sch00] claims that digital watermarking (used to assure ownership rights) "just won't work". One point he makes is that a customer can acquire watermarked data under a false identity, and then distribute the tagged media without fear of repercussion. This is a very valid concern. Another point he makes is that "the mechanisms for watermarking will eventually become public, and when they do, they can be reverse engineered and removed from the image". One might reply to this that the use of a keyed watermarking mechanism changes the process of introducing and later detecting watermarks to a process similar the encryption of data. Even though the algorithm is known, it may be hard to detect and remove the watermarks without knowing the keying information. Time will tell...

6 Conclusions and Future Work

Watermarking has long been accepted as one mechanism to protect intellectual property. This paper presented two novel methods that can be employed to increase the value of watermarking techniques when a wider (but still limited and traceable) distribution of the media is intended. The first, called *hierarchical watermarking*, specifies the iterative application of watermarking techniques in a hierarchical fashion. It enables a group of distributors to detect leakage of tagged data in a fashion similar to how spoofed certificates can be detected in a public key hierarchy. It thus results in the ability to use watermarking in a scalable and trustworthy manner in distributing systems.

Because watermarking is a technique that allows to transform the same data into individually tagged copies that can be distinguished only by a select few, it was generally accepted such tagged data need to be created and distributed individually. Our design of a technique named *bulk-tagging* shows how it is possible to create a single data set once but still have it be tagged individually on the consumer side. While a small amount of user participation is required, we argue that this is in many scenarios an acceptable tradeoff. Data marked through bulk-tagging is equivalent to data marked and delivered individually, and does not create significant additional security risks.

There is no reason why media producers and (sub-)distributors should not be allowed to combine the approaches of bulk- and hierarchical watermarking, leading to even more flexible usage schemes.

While we presented one proof of concept for how bulk-tagging works, we see a lot more work that needs to be done. It would be especially interesting to study which watermarking techniques would fare well with bulk-tagging for different classes of data, such as movies, images, program object and source code, or audio. It would also be interesting to study the applicability to hierarchical tagging for more structured data sets (such as program object code), and to find other (stronger) segmentation methods than the naive splitting of data into independent shares.

Acknowledgments

Many thanks to Radia Perlman and Amit Gupta, for input and discussions during the formative stages of hierarchical tagging.

References

[BPC99] Ian Brown, Colin Perkins, and Jon Crowcroft. Watercasting: Distributed watermarking of multicast multimedia. In *First International Workshop on Networked Group Communication (NGC99)*, 1999.

[Car95] Germano Caronni. Assuring ownership rights for digital images. In H. H. Brüggemann and W. Gerhardt-Häckl, editors, *Proceedings of Reliable IT Systems VIS '95*, Germany, 1995. Vieweg Publishing Company. "http://www.olymp.org/~ caronni/WWW-new/work/papers/givis-final.pdf".

[CKLS96] Ingemar J. Cox, Joe Killian, Tom Leighton, and Talal Shamoon. A secure, robust watermark for multimedia. In *Workshop on Information Hiding*. Newton Institute, University of Cambridge, May 1996.

[Dro96] Stefan Droste. New results on visual cryptography. In *Crypto '96, Lecture Notes in Computer Science vol. 1109*, pages 401–415. Springer-Verlag, 1996.

[Kah96] David Kahn. *The Codebreakers*. Scribner, New York, New York, revised edition, 1996.

[MW98] N. Memon and P. Wong. Protecting digital media content. In *Communications of the ACM,Vol. 41, No. 7*, July 1998.

[PAK98] Fabien Petitcolas, Ross Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Proceedings of the Second Workshop on Information Hiding, vol. 1525 of Lecture Notes in Computer Science, Springer Verlag. pp. 219-239.*, April 1998.

[PW97] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *4th ACM Conference on Computer and Communication Security*, 1997.

[Sch00] Bruce Schneier. Security tricks – digital watermarking. In *Secrets and Lies*, pages 248–250. Wiley Computer Publishing, 2000.

[SHKQ99] Julien P. Stern, Gael Hachez, François Koeune, and Jean-Jacques Quisquater. Robust object watermarking: Application to code. In *Information Hiding Workshop '99*. Springer-Verlag, 1999.

[Sim92] G.J. Simmons. An introduction to shared secret and/or shared control schemes and their application. In *Contemporary Cryptology, The Science of Information Integrity*, pages 441–497. IEEE Press, 1992.

[ST98] T. Sander and C. F. Tschudin. On software protection via function hiding. *Lecture Notes in Computer Science*, 1525:111–123, 1998.

[TNM90] K. Tanaka, Y. Nakamura, and K. Masui. Embedding secret information into a dithered multilevel image. In *Proceedings of the 1990 IEEE Military Communications Conference, pp. 216-220*, September 1990.