COPYRIGHT PROTECTION PROTOCOLS BASED ON ASYMMETRIC WATERMARKING

The Ticket Concept

Scott Craver
Department of Electrical Engineering
Princeton University
sacraver@ee.princeton.edu

Stefan Katzenbeisser
Vienna University of Technology
skatzenbeisser@acm.org

Abstract Traditional watermarking systems require the complete disclosure of a watermarking key in the watermark verification process. In most systems, an attacker is able to remove the watermark completely once the key is known. We propose the use of public-key watermarking systems, systems in which a watermark is inserted using a private key but checked with a public key. A construction of such a scheme is given, which uses scrambled documents and so-called *ownership tickets* in the watermark verification process.

Keywords: Copyright protection, public key watermarking, asymmetric watermarking, watermarking protocols.

Introduction

With the increasing availability and distribution of media in digital form, the protection of intellectual property faces new challenges. The ability to reproduce content easily, cheaply and without loss of quality is undermining the film, music and entertainment industries. As a consequence, technologies which prevent illegal copying are of increasing importance. However, techniques which limit the access to copyrighted

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35413-2_36
R. Steinmetz et al. (eds.), *Communications and Multimedia Security Issues of the New Century*

material or inhibit the copy process itself seem to be difficult to implement in open systems.

Since copy protection cannot be achieved in many situations, the rightful owner of a digital document may want the power to detect or prove its misappropriation. One particularly promising copyright protection method is the use of digital watermarking techniques, which embed information identifying the copyright owner's identity within the content. Whenever the copyright of a digital document is in question, this copyright information can be extracted to identify the rightful owner [6].

However, we argue that it is highly questionable whether traditional *private* watermarking systems can actually be used for proof of ownership, since the verification of a watermark by another party requires its complete disclosure. Once this information is known to the public, subsequent attackers are able to remove the watermark and thereby defeat the goal of copyright protection. With the watermarked media already in distribution, there is no possibility to mark it a second time. In short, traditional private watermarking schemes work only once as proof of ownership, thereafter offering no protection to the content.

To cope with this problem, we propose the use of asymmetric watermarking systems, similar to public key cryptography, in which the watermark is embedded using a private key. However, the watermark extraction process relies on so-called *ownership tickets*, which contain enough information to successfully prove the presence of a watermark but do not contain enough information to remove it.

1. PRIVATE WATERMARKING SYSTEMS

Basically a *private* watermarking scheme consists of two algorithms, one embedding and one extraction algorithm. The embedding algorithm inserts a watermark into digital media using a secret (symmetric) key, thereby generating the watermarked media.

Depending on the nature of the extraction algorithm, two types of watermarking schemes can be identified. The extraction process of *private watermarking* systems takes the watermarked media, the original media, the watermark and the secret key and outputs TRUE if the watermark is actually present. In the case of *blind watermarking* systems, the extractor extracts the watermark given only the watermarked media and the key. Blind watermarking is preferable to non-blind marking, because publication of the original, unmarked media allows subsequent piracy.

Watermark extraction should also be possible in case small modifications have been applied to the marked media. Such modifications can

be the result of intentional attacks in order to remove the mark or the result of coding schemes (e.g. lossy compression) and errors during the transmission [6]. Schemes which are able to retrieve a watermark from a distorted media are called *robust.*

1.1. AN EXAMPLE

Hartung and Girod [5] developed a technique to watermark digital video based on spread spectrum systems, which will serve as an example in this paper. Let $a_j \in \{-1, 1\}$, be the watermark (encoded as strings of 1 and -1) to be hidden in a linearized video stream v_i. The sequence a_j is upsampled by a factor cr, called chip-rate, to obtain a sequence $b_i = a_j$ for $j\,cr \leq i < (j+1)\,cr$. The new sequence b_i is modulated by a pseudo-noise signal p_i ($p_i \in \{-1, 1\}$), scaled by a constant α and added to the video stream to be watermarked: $\hat{v}_i = v_i + \alpha b_i p_i$. Here, $\hat{v}_i$ denotes the watermarked video stream. Due to the noisy appearance of p_i, the watermark $\alpha b_i p_i$ is also noise-like and therefore difficult to detect and remove.

In order to verify the mark, the signal p_i used in the embedding process must be known; for more details, refer to [5]. If a different sequence is used, the recovered watermark bits are random. It seems that this scheme is relatively robust to modifications of the video stream.

1.2. ARE WATERMARKS PRIVATE?

Unfortunately, the watermark verification process requires the complete disclosure of the secret watermarking key (in the previous example the sequence p_i). Once the key is known to the public, an attacker is able to remove the watermark completely. Thus, the watermarking system may be secure as long as there is no need to verify the watermark; once the mark is used as an evidence of copyright ownership, the mark can be removed. If several digital objects were watermarked with the same mark and key, all other watermarks are insecure too. In the case of the watermarking system outlined in the previous section, an attacker can simply subtract the sequence $p_i \alpha b_i$ from the watermarked video signal.

Another issue is the usability of private-key based schemes. Only those who know the key (i.e., the copyright holder) are able to determine if a watermark is present. This prevents the mark from being used for detection by third parties, say if a potential customer wishes to determine the owner of an unlabeled image or piece of music. In general, the usefulness of a watermark revolves around its publication.

Such problems could be avoided if there existed a watermarking algorithm analog to public key cryptography. Each user would have a

private key to embed a watermark which everybody could verify using the corresponding public key. Unfortunately, such schemes seem to be difficult to engineer, as the following example illustrates.

Hartung and Girod [4] presented an extension to their watermarking system, in which a mark is inserted by a private key but where the presence of the watermark can be checked using a different (public) key. By making only parts of the sequence p_i public and replacing all other bits by a random sequence, they obtain a "public" key p_i^p. Using this public key, the watermark can be extracted in the same manner as indicated in the previous section. Due to the redundant embedding of the watermark bits, the watermark can be successfully retrieved.

Unfortunately, their scheme is susceptible to a similar attack: the public portion of the watermark can be successfully destroyed. Although the watermark could still be retrieved using the private sequence p_i, the benefits of the public key are lost. On the other hand, the watermark owner could construct a new public key using sequence elements not yet revealed. If too many such marks are constructed, however, the whole mark could be eliminated.

There is also a possibility of a more subtle attack. Any attacker can construct a purported watermark sequence which could be detected in the media at any given strength level. Further, an attacker can take the public sequence p_i^p and insert a fake watermark into a different video (which could also be verified with the public key p_i^p). These attacks, known as "protocol" or "ambiguity" attacks are normally averted by requiring watermarks to be generated in a standard, one-way manner, e.g. using a one-way hash of a description of the marked content as the seed to a secure pseudo-random number generator; so that an attacker's arbitrarily constructed mark is not likely to be a "legal" one, and marks are not transferable to other objects. However, this approach is no longer possible if the original mark is kept private, as the watermark owner can not reveal the mark to prove its legality. This is not a fault specific to this scheme, but an issue with asymmetric watermarking schemes in general. Ambiguity attacks, long prevented with the simple prescription of a one-way function, are considerably harder to prevent when the watermark data can not be disclosed.

Several asymmetric watermarking systems were proposed (that use e.g. properties of Legendre sequences or one-way signal processing operations). Unfortunately, none of these schemes is sufficiently robust against malicious attacks (see [3] for a discussion).

2. ASYMMETRIC WATERMARKING

Instead of designing a "pure" asymmetric watermarking scheme, one could also design a cryptographic protocol around the watermark verification process which makes a traditional symmetric (blind) watermarking algorithm asymmetric. We will construct such a scheme consisting of an insertion algorithm (called *Emb*) and a probabilistic protocol *Ver* for watermark verification.

The embedding algorithm inserts a watermark W into a digital object O by using a private key. In the verification process, the protocol *Ver* has access to a watermarked object $\overline{O}$ and the corresponding public key; *Ver* proves the presence of W in $\overline{O}$, without revealing the exact location and nature of the watermark (specified by the private key).

The system must satisfy the following requirements in order to be useful for copyright protection:

- The embedding process should be robust; i.e. it should not be possible to remove a watermark without rendering the data useless.
- Knowledge of the public key does not enable an attacker to remove a watermark; more specifically, the public key must not reveal the location of the watermark in the cover.
- Both *Emb* and *Ver* must be computationally feasible.
- It must be computationally infeasible to deduce the private key from the public key.
- It must not be possible to use the public key to forge a watermark on a different digital object.

2.1. WATERMARK VERIFICATION

The main idea of the protocol is to prove the presence of a watermark in a scrambled version of the watermarked document. The scrambling is determined by a permutation τ which is kept secret from the verifier (and will form the secret key of the copyright holder).

Fix any subgroup S of the permutation group S_n on n elements. Alice chooses any permutation $\tau \in S$, generates a large random graph with n nodes and computes the permuted graph $\tau(G)$. The public key of Alice consists of both G and $\tau(G)$, whereas her private key is the permutation τ. Note that deducing the private key from the public information is likely to be infeasible, as it would require finding an isomorphism between two large graphs. We will assume that Alice publishes her public watermarking key cryptographically signed in a public database.

The media object O to be marked is treated as an array of n samples (if the object is larger, chunks of size n will be marked independently). Alice

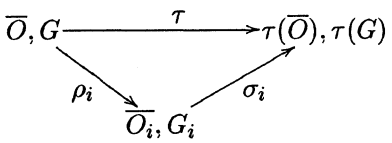
inserts her watermark W into the media object by using a traditional symmetric watermarking algorithm, yielding a watermarked object $\overline{O}$. We will first describe a protocol that allows watermark detection in the unmodified watermarked object *only* and give extensions in section 3.

Alice sets up a public directory containing all her watermarked digital documents together with information needed in the verification step. Denote with $\overline{O}$ a watermarked media object and with $\tau(\overline{O})$ its scrambled version. Alice publishes for each object $\overline{O}$ the scrambled object $\tau(\overline{O})$ together with the scrambled watermark $\tau(W)$. Note that W is kept secret, as it would allow subsequent removal of the watermark in $\overline{O}$.

The main tool in the construction of the verification protocol is the use of "commitments," cryptographic primitives that allow Alice to commit to a sequence of bits and to keep them secret until some time later [8, p. 86]. Symmetric cryptography can be used to construct such a protocol: Bob sends a random number R to Alice; Alice encrypts R together with the bitstring b she wants to commit to using a random symmetric key K: $E_K(R, b)$ and sends the result to Bob. When she wants to disclose the secret, she sends Bob the session key K. Bob decrypts the information and checks whether the first bits of the plaintext equal his random number R.

In case Alice wants to prove the presence of her watermark W in a digital object $\overline{O}$, Alice and Bob engage in the following interactive protocol. By using the protocol, Alice proves Bob that she actually knows the secret τ and that her watermark is present in $\overline{O}$.

- Alice generates two permutations σ_i and ρ_i with the property that $\sigma_i \circ \rho_i = \tau$, computes scrambled versions of her graph $G_i = \rho_i(G)$ and the watermarked image $\overline{O_i} = \rho_i(\overline{O})$:



- Alice generates two commitments (containing the permutations σ_i, ρ_i) and a hash of both $\overline{O_i}$ and G_i:

$$OT = \langle C_1(\sigma_i),\ C_2(\rho_i),\ H(\overline{O_i}),\ H(G_i) \rangle .$$

  OT will be called an *ownership ticket.* Alice sends OT back to Bob. To prevent tampering with the data, Alice signs the ticket.

- Bob flips a coin and asks Alice either (i) to open commitment C_1 or (ii) to open commitment C_2. Alice complies.

- In case (i), Bob computes $\overline{O_i} = \sigma_i^{-1}(\tau(\overline{O}))$ and $\overline{G_i} = \sigma_i^{-1}(\tau(G))$ from the knowledge of σ_i contained in the first commitment. He hashes both $\overline{O_i}$ and G_i and checks whether their values agree with the corresponding hash bits in the ownership ticket. Then, Bob verifies the presence of the watermark $W_i = \sigma_i^{-1}(\tau(W))$ in the scrambled document $\overline{O_i}$.
- In case (ii), Bob computes $\overline{O_i} = \rho_i(\overline{O})$ and $G_i = \rho_i(G)$ from the knowledge of ρ_i contained in the second commitment; he hashes both $\overline{O_i}$ and G_i and checks whether these hash values agree with the hash bits contained in the ownership ticket.
- Alice and Bob perform these steps k times. If all tests passed, Bob assumes Alice's watermark to be present in $\overline{O}$.

If Bob asks Alice to perform case (i), he will be able to verify the scrambled mark in the scrambled image; case (ii) is necessary to prove knowledge of τ. In section 2.3 it will be shown that Alice can fool Bob in each round with probability of at most $1/2$. By repeating the process, the probability that Alice is cheating can be decreased below any threshold. Note also that in case both commitments C_1 and C_2 are opened in one round, τ could be computed, which would allow subsequent piracy.

It is also possible to remove the interactive nature of this protocol by taking a cryptographically secure hash function as a source of unbiased random bits. Instead of asking Bob to flip a coin, Alice can build a hash of all ownership tickets constructed in the second step and take the first bits of the output as random coin flips. Thus, no input from Bob is needed. However, a non-interactive protocol requires more rounds than an interactive one to be valid, since Alice could discard several rounds of the protocol if the output of the hash function does not satisfy her needs [8].

One apparent drawback of the scheme is the amount of information needed to be transmitted in the verification process, as each ticket contains commitments of two large permutations. Furthermore, the verifier needs to know the public watermarking key and both the scrambled document and watermark. With large permutations the size of all ownership tickets to be transmitted might actually be larger than the size of the document in which the watermark must be verified.

2.2. SIGNAL PRECONDITIONING

Crucial to the security of the overall scheme is that the verifier cannot gain any information on the secret permutation τ by comparing the scrambled and original documents. Knowledge of both $\overline{O}$ and $\tau(\overline{O})$

might reveal some information on τ, as e.g. it could be easy to detect atypical samples in both the watermarked and scrambled image. In order to prohibit this flow of information, it is necessary to do some preprocessing on the documents before watermark insertion.

Two opposing forces make the preconditioning problem difficult. The first is that a single sample of multimedia data is likely to vary over a wide range. This will result in a great deal of unique values, and from a permuted array of those values an attacker can derive a great deal of extra information about the permutation used.

The second problem is preservation of robustness. One could, for instance, imagine a solution to the first problem in which each sample is hashed to a value in a much smaller range $[0, \ldots, M]$, and the watermark embedded and detected in this domain. The smaller range prevents unique or rare sample values. However, if this operation is not continuous, in the sense that a small change in the original sample can result in a significant change in the hashed value, then the watermark is no longer robust.

A solution is the use of the Hilbert space-filling curve. In the discrete case, there is a Hilbert traversal $h_{n,m}\colon [0, \ldots, 2^{mn} - 1] \to [0, \ldots, 2^m - 1]^n$ mapping any nm-bit number to an n-dimensional vector of m-bit numbers. If the binary numbers are treated as fixed-point values in $[0, 1)$, then as the precision m increases for a fixed dimension n, the sequence of functions $\{h_{n,m}\}$ limits to a continuous function $h_n\colon [0, 1] \to [0, 1]^n$. This continuity property in the limit case generally confers comparable properties in the discrete case, which has been exploited for a number of applications [2]. In our application, a Hilbert traversal $h_{n,m}$ can be used to chop an array of samples, each varying over a large range, into an array of n times as many samples, but whose values vary over an exponentially smaller range. Further, the continuity of the operation preserves the robustness of the watermark, as small changes to the samples translate into small changes in the conditioned array.

These mappings are also invertible, and easy to compute using bitwise operations given the Hilbert curve's construction. In fact, the function $h_{n,1}$ is the n-bit Gray code [7].

2.3. SECURITY CONSIDERATIONS

Crucial to the security of the overall protocol is the number of interactive rounds performed, as Alice is able to cheat in each round with a probability of at most 1/2:

Information Leakage. During the protocol, no information about τ is leaked. If one cannot determine τ, then the value of ρ_i reveals

nothing about the value of σ_i, and vice versa, since S is a group. For every possible secret τ and every possible revealed permutation ρ_i, there exists a σ_i such that $\tau = \sigma_i \circ \rho_i$, and analogously for a revealed σ_i. More specifically, if τ is chosen uniformly, and in each round ρ_i is chosen uniformly (σ_i being determined by the values of ρ_i and τ), then σ_i is also uniform, and the probability that a given value of τ is being used remains unchanged when conditioned on the knowledge of ρ_i or σ_i. In other words, we satisfy the definition of perfect secrecy.

Forging a watermark. Suppose Bob wants Alice's mark to appear in a digital object $\overline{O}$ without knowledge of Alice's secret key τ. We will assume that he cannot change Alice's public key $\tau(G)$ and G (this is a reasonable assumption, e.g. if the public key is cryptographically signed by a certification authority). As Bob does not know the true value of τ, he can only guess permutations σ_i and ρ_i in each round that will—with high probability—satisfy $\sigma_i \circ \rho_i \neq \tau$. However, he can fool a verifier in each round with probability 1/2. Basically, he can choose between two strategies:

- He computes $O_i = \rho_i(\overline{O})$ and $G_i = \rho_i(G)$ and constructs the ownership tickets with hashes of these values and commitments of ρ_i and σ_i. In this approach, he is able to fool the verifier in case (ii). However, if the verifier asks him to open the first commitment, we will have $G_i' := \sigma_i^{-1}(\tau(G)) \neq G_i$; thus, the reconstructed object G_i' will differ from the true object G_i. The verifier will detect Bob's actions by comparing the hash values of G_i' and G_i. Similar results hold for the scrambled images $\overline{O}$.
- To fool the verifier in case (i), Bob takes a different approach: he simply adds Alice's scrambled watermark $\tau(W)$ to the scrambled watermarked object $\tau(\overline{O})$, yielding a faked scrambled object $\tau(\overline{O})'$; he pretends that $\tau(\overline{O})'$ is the "true" scrambling of $\overline{O}$ induced by τ. By using the permutation σ_i to construct the ownership ticket, the verifier can be fooled in case (i). However, the verifier will detect Bob's actions in case (ii), as $\tau(\overline{O})'$ is not the scrambled version of $\overline{O}$ and the permutation ρ_i was chosen arbitrarily.

Pretending the presence of a watermark. Suppose Bob wants his *own* watermark to appear in a digital object $\overline{O}$, although the object was not watermarked by him. Although he cannot change $\overline{O}$, he is free to add his own watermark W to $\overline{O}$ and scramble the object with his own secret key τ, yielding $\tau(\overline{O})$. Again, he pretends that $\tau(\overline{O})$ is the *true* scrambling of $\overline{O}$ induced by τ. For each ticket, he computes permutations so that $\sigma_i \circ \rho_i = \tau$. Again, he can fool a verifier in each

round with probability 1/2. As above, he may construct the ownership tickets by using either the permutation σ_i or ρ_i, thereby fooling a verifier in one of the two cases. However, the other case will reveal his actions, as $\tau(\overline{O})$ is *not* a scrambling of $\overline{O}$.

Inversion attacks. Another serious attack is inspecting $\tau(\overline{O})$, and constructing a $\tau(W)$ which is detectible in $\tau(\overline{O})$. The corresponding W is then shown to be a watermark present in $\overline{O}$ by using the probabilistic verification protocol.

Fortunately, our system can prevent these attacks by an appropriate design of W: we let a "legally" derived mark be defined as $W = \tau^{-1}(X)$, where X is the output of a pseudo-random generator seeded with a one-way hash. This proper generation of $X = \tau(W)$ can be verified without revealing W, and in fact this allows a more efficient representation of public keys.

3. MODIFIED DOCUMENTS

In case an attacker modifies $\overline{O}$, the above scheme is no longer applicable directly. As both Alice and Bob must work on the same watermarked object, Bob has to transmit the modified watermarked object to Alice. One might be tempted to construct a protocol in which Alice provides a scrambled version of the document to Bob and proceeds in proving the presence of the watermark in this document.

Unfortunately, the extended scheme becomes susceptible to a chosen plaintext attack. As Bob is free to provide Alice *any* modified document, he can gain some knowledge of the secret permutation τ by modifying the least significant bits of the watermarked document before asking Alice to prove the presence of the watermark. Suppose we have an image in which Bob manages to replace the least two significant bits. If the scheme involves a secret permutation on an array of 65536 elements, eight verification steps with different carefully chosen modified images are enough to break the scheme. Every pixel is assigned a unique number in the interval $[0, 65535]$; for the first modified image, the attacker embeds the least two significant bits of the pixel number into each pixel. In the second test image, he embeds bits 3 and 4 of the pixel number, etc. By comparing the original (modified) and the shuffled image, he can reconstruct the permutation.

One possible countermeasure is to wipe out the least significant bits before the watermark extraction process. Before computing the scrambled document $\tau(\overline{O})$, Alice replaces the (two) least significant bits of $\overline{O}$'s samples with the output of a random number generator (using a seed known to Bob). Alice and Bob then proceed in the usual manner. This

should not greatly hurt watermark detection, as most current watermarking systems are quite robust. However, it is not clear a priori how many bits of the samples must be replaced before the watermark detection process in order to prohibit the attack. Furthermore, this process surely decreases the overall robustness of the whole scheme.

A different approach would be to enforce watermark verification in the unmodified digital object *only* through a copyright protection protocol wrapped around the watermark verification process. Similar to the approach in [1], one might model ownership of digital works as an equivalence relation on the set of all possible digital objects. One then identifies a specific representative of each equivalence class. By assuming that if one digital object belongs to a specific owner, all other objects within the same equivalence class are courtesy of the same owner, it is sufficient to prove watermark presence in representatives of equivalence classes. Although an equivalence class should intuitively model "perceptual similarity", it is not clear how such an equivalence relation could be defined formally. Furthermore, it has the inherent danger that an attacker might be able to change a marked object imperceptibly, but succeeds in expelling the digital object outside of its equivalence class and thus rendering extraction impossible.

Another approach is to introduce a second permutation in the verification process in the following way: Bob sends the modified marked object $\overline{O}$ to Alice; Alice chooses another random permutation ξ, computes $\xi(\overline{O})$ and $\tau(\xi(\overline{O}))$. Both Alice and Bob proceed to prove the presence of the watermark in the scrambled object $\tau(\xi(\overline{O}))$, where $\xi(\overline{O})$ substitutes $\overline{O}$ in the presented protocol. Once Bob is convinced that a watermark is present in $\tau(\xi(\overline{O}))$ and Alice indeed knows the private key τ, Alice proves Bob that $\xi(\overline{O})$ is actually a legal scrambling of the object $\overline{O}$ without revealing ξ. This can e.g. be done by a similar technique that is used in interactive proofs for the graph isomorphism problem (see e.g. [8, p. 104]). Although this approach does not suffer from the problems mentioned previously, it considerably increases the complexity of the watermark verification protocol.

4. CONCLUSIONS

One of the most challenging problems in copyright protection includes the construction of public-key watermarking systems which use a private key to embed a watermark, but in which the watermark's presence can be checked by a public key. The key problem to be solved is that of providing enough information to show that a watermark is present in digital media without revealing enough information to erase the water-

mark. We have shown that it is—in theory—possible to construct such schemes out of traditional symmetric watermarking systems, if a probabilistic cryptographic protocol is used in the verification step.

However, the problem is far from solved. The protocol outlined in section 2.1 requires a large amount of data to be published in the verification process, although special coding of permutations might help in reducing the ticket size. Besides this issue, future research might concentrate on the construction of public key watermark protocols that do not use document scramblings as a basis for the construction in order to prevent chosen plaintext attacks.

Unfortunately, it seems that the "weak point" of todays watermarking systems is still the embedding method itself; most known watermarking systems do not merge the digital data and the watermark completely. Until this problem is solved, any watermarking protocol may be susceptible to attacks outside the scope of a protocol.

References

[1] A. Adelsbach, B. Pfitzmann, A. Sadeghi, "Proving Ownership of Digital Content", in *Proceedings of the Third International Workshop on Information Hiding*, Springer Lecture Notes in Computer Science, vol. 1768, 2000, pp. 117–133.

[2] S. Craver, B-L. Yeo, and M. M. Yeung, "Multilinearization Data Structure for Image Browsing," in *Proceedings of SPIE vol. 3656, Storage and Retrieval of Image and Video Databases,* February 1999, pp. 155–166.

[3] J. J. Eggers, J. K. Su, B. Girod, "Asymmetric Watermarking Schemes", in *Sicherheit in Mediendaten, GMD Jahrestagung, Proceedings*, Springer Verlag, 2000.

[4] F. Hartung, B. Girod, "Fast Public-Key Watermarking of Compressed Video", in *International Conference on Image Processing (ICIP'97)*, vol. I, 1997, pp. 528–531.

[5] F. Hartung, B. Girod, "Watermarking of Uncompressed and Compressed Video", in *Signal Processing*, vol. 66, no. 3, May 1998, pp. 283–301.

[6] S. Katzenbeisser, F. A. P. Petitcolas (eds.), *Information Hiding Techniques for Steganography and Digital Watermarking*, Boston, London: Artech House, 2000.

[7] H. Sagan, *Space-Filling Curves.* New York: Springer-Verlag, 1994.

[8] B. Schneier, *Applied Cryptography; Protocols, Algorithms and Source Code in C*, Wiley, 1996.