Cryptography Meets Watermarking: Detecting Watermarks with Minimal or Zero Knowledge Disclosure

André Adelsbach
Universität des Saarlandes, FR 6.2
D-66123 Saarbrücken, Germany
adelsbach@cs.uni-sb.de

Stefan Katzenbeisser
Institute for Information Systems
Vienna University of Technology
skatzenbeisser@acm.org

Ahmad-Reza Sadeghi
Universität des Saarlandes, FR 6.2
D-66123 Saarbrücken, Germany
sadeghi@cs.uni-sb.de

ABSTRACT

Digital watermarking schemes embed additional information into digital data and are used in various applications, such as proof of ownership or fingerprinting. For such applications, the presence of watermarks must be provable to any possibly *dishonest* party. Standard watermark detection requires knowledge of sensitive information like the watermark or the embedding key. This is a major security risk, since this information is in most cases sufficient to remove the watermark.

Zero-knowledge watermark detection is a promising approach to overcome security issues residing around the process of watermark detection: cryptographic techniques are used to prove that a watermark is detectable in certain data, without jeopardizing the watermark. This paper gives an overview over such schemes and discusses their properties.

1 Introduction

Watermarking methods were invented in the 1990's in order to embed additional information (like an identity string of a copyright owner) in some multimedia object. Since then, watermarks have successfully been used as primitives in protocols for resolving disputes over copyright, in fingerprinting schemes or proofs of ownership. Until recently, all watermarking methods were *symmetric*, i.e., their watermark detection process required the watermark and the same key that was used in the embedding process. Once this information is disclosed to a party, this party can completely remove the watermark. This property is a strong limitation of the usability of symmetric watermarking, since most applications require at some point in time the detection of a watermark by a – in reality – not fully trustworthy party or device.

Two approaches were taken to tackle this problem. First, truly *asymmetric watermarking schemes* were proposed by different authors, which use different keys for watermark embedding and detection. However, most systems were broken; especially the presence of a watermark detector, i.e., knowledge of the public detection key, leads to an increased threat of oracle attacks.

On the other hand, the approach of *zero-knowledge watermark detection* applies cryptographic techniques to the detection of watermarks. These systems substitute the watermark detection process by an interactive cryptographic protocol, involving a *prover*, e.g., an alleged copyright owner, and a possibly dishonest verifier, e.g., a judge verifying the copyright claim.

The goal of zero-knowledge watermark detection is to prove the presence of a specific watermark in a digital object *without compromising the security of this watermark*. Therefore, these protocols should ideally fulfil the following requirements:

1. *Inputs conceal watermark and key:* The necessary inputs do not reveal any information about the watermark and the detection key.
2. *Protocol is zero-knowledge:* A run of the protocol does not disclose any information *in addition* to the inputs of the protocol.

These properties guarantee that a symmetric watermark stays as secure as if the protocol had not been executed at all. Thus, zero-knowledge watermark detection can improve the security of many applications, which rely on symmetric watermarking schemes, and can reduce the necessary trust in certain parties or devices. Weaker requirements may be sufficient for some applications, as long as the information leaked to the verifier (by the inputs or by the protocol-run) is insufficient to attack the underlying symmetric watermarking scheme.

After introducing necessary cryptographic primitives in Section 2, we provide three different constructions for zero-knowledge watermark detection in Sections 3 and 4. In Section 5 we compare the security offered by these constructions. Section 6 concludes and gives open problems and research directions.

2 Cryptographic Building Blocks

A commitment scheme $(com, open)$ for a message space M consists of a protocol com to commit to a value $m \in M$ and a protocol $open$ that opens a commitment. A commitment to a value m is denoted by $com(m, par_{com})$ where par_{com} stands for all public parameters needed to compute the commitment value. To open a commitment com the committer runs the protocol $open(com, par_{com}, sk_{com})$ where sk_{com} is the secret opening information of the committer. For brevity we omit par_{com} and sk_{com} in the notation of *com*() and *open*(). Furthermore, we use *com*() and *open*() on tuples over M, with the meaning of component-wise application of *com*() or *open*().

Commitment schemes fulfil a *hiding (secrecy)* and a *binding (committing)* property. The first one requires that a commitment $com(m)$ does not reveal any information about the committed message m. The second one requires that a dishonest committer cannot open a commitment to another message $m' \neq m$ than the one (m) to which he has committed before.

Additionally, we require the following *homomorphic property*: Let $com(m_1)$ and $com(m_2)$ be commitments to arbitrary messages m_1, $m_2 \in M$. Then the committer can open $com(m_1) * com(m_2)$ to $m_1 + m_2$ without revealing additional information about the contents of $com(m_1)$ and $com(m_2)$. The commitment scheme introduced in [8] fulfils all these properties.

Zero-knowledge proof systems (see [9]) allow a party, called prover $\mathcal{P}$, to convince another party, called verifier $\mathcal{V}$, of some fact without revealing any additional information by the proof. In [3] efficient and secure zero-knowledge proof systems for proving relations in modular arithmetic (addition, multiplication, exponentiation) between committed[1] numbers are proposed: Given commitments to the values $a, b, c, m \in M$ one can prove that $a+b \equiv c \bmod m$, $a*b \equiv c \bmod m$ or $a^b \equiv c \bmod m$. Furthermore, [2] described efficient zero-knowledge proof systems for proving that a committed number lies in an exact interval.

3 Solutions based on Interactive Proof Systems

Two protocols for proving watermark presence with minimum knowledge (that work with a potentially large class of watermarking schemes) were proposed in [6] and [7]. The first construction depends on a blinding process induced by a secret permutation of the watermarked object, whereas the other one draws its security from concealing a genuine watermark by some fake ones.

3.1 Secret Permutations

Assume we are given a watermarking system that allows a permuted watermark to be detected in an equally permuted object. Let τ be any permutation on n elements and G be a graph with n nodes. The public key of the content provider consists of G and $\tau(G)$, whereas τ is the private key and is therefore kept secret (as finding an isomorphism between two graphs is believed to be intractable, an attacker cannot infer the private key from the public key). The verification process consists of an interactive multi-round protocol between the prover and a verifier. In each round, the prover is able to cheat with a probability of 1/2. By performing several rounds, the verifier can gain any degree of certainty that a valid mark is actually present.

Let WM be a (secret) watermark and $\overline{O}$ be a watermarked object. Before the protocol starts, the content provider publishes $\tau(\overline{O})$ and $\tau(WM)$. A cryptographic protocol now proves that $\tau(\overline{O})$ is actually a permuted version of $\overline{O}$ and that the watermark is present. In each round, the content provider chooses two permutations σ_i and ρ_i with the property that $\sigma_i \circ \rho_i = \tau$.

He constructs an *ownership ticket*, containing commitments of both σ_i and ρ_i; furthermore, the ticket contains hashes of the permuted objects $\overline{O_i} = \rho_i(\overline{O})$ and graphs $G_i = \rho_i(G)$. The verifier first checks whether the scrambled watermark $\tau(WM)$ is indeed detectable in the scrambled object $\tau(\overline{O})$. He then proceeds by flipping a coin. Depending on the outcome of his coin flip, he asks the content provider to open either the commitment containing ρ_i or the commitment of σ_i.

If the commitment containing ρ_i is opened, the verifier is able to compute scrambled versions of the document $\overline{O}$ and graph G; he then hashes the scrambled object and checks whether the hash value agrees with the bits contained in the ownership ticket. If, however, the commitment containing σ_i is opened, the verifier applies the inverse permutation σ_i^{-1} to both the scrambled watermarked document $\tau(\overline{O})$ and graph $\tau(G)$; again he verifies whether the hash of the resulting objects equal the hash values in the ownership ticket.

3.2 Ambiguity Attacks

In an ambiguity attack, an attacker tries to guess a watermark WM and an alleged "original" object O such that WM is already contained in a given object $\overline{O}$ and $\overline{O}$ seems to be the watermarked version of O. By concealing a true watermark WM among a set of fake watermarks $WM_1, \ldots, WM_n$ constructed through ambiguity attacks, an attacker (equipped solely with a watermark detector) cannot decide which of the watermarks is not counterfeit. This is the basis of another minimum-knowledge watermark detection protocol.

A watermark will be called valid, if the prover knows its discrete logarithm (w.r.t a specific generator a) in a field $\mathbb{Z}_p$. However, as computing the discrete log seems to be intractable, he will be unable to provide the logarithm of watermarks produced through ambiguity attacks. In the watermark insertion process, the prover constructs a valid watermark, finds $n-1$ other marks through ambiguity attacks and arranges all marks in a random order:

- He constructs a valid watermark by choosing a random exponent e and computing $WM = a^e \pmod p$. Afterwards, he embeds WM in his object.
- Using ambiguity attacks, he determines $n-1$ counterfeit watermarks $WM_1, \ldots, WM_{n-1}$ and publishes all watermarks $WM, WM_1, \ldots, WM_{n-1}$ in a random order.

[1] Although not mentioned explicitly in [3], these protocols work also for the commitments from [8] (private communications with Jan Camenisch).

In the verification step, he has to prove that "most" of the watermarks WM, $WM_1, \dots, WM_{n-1}$ are still detectable in the watermarked object and that at least one of them is genuine (i.e. he knows its discrete logarithm), without revealing which one. As a potential attacker does not know the "genuine" mark, he potentially has to remove many watermarks until the verification process fails, which hopefully renders the object useless (on the average, $n/2$ watermarks must be removed). The verification process is again an interactive multi-round protocol:

- The verifier checks whether all watermarks $WM_1, \dots, WM_n$ are actually contained in the object in question (if one mark is not detectable any more, it is discarded from further computations).
- The prover constructs n blinding exponents $h_1, \dots, h_n$, computes $b_i = WM_i \cdot a^{h_i} \pmod{p}$ and publishes all blinded watermarks b_i in a permuted manner.
- The verifier flips a coin; if the result is heads, he challenges the prover to reveal the blinding exponents h_i to verify that the b_i are actually blinded versions of the WM_i.
- If the result is tails, the verifier asks the prover to reveal the discrete log of one of the values b_i. The prover can do this, as the discrete log of the blinded true watermark b_j is $e + h_i \pmod{p-1}$.

4 Zero-knowledge Watermark Detection

It is possible to construct watermark detection protocols which leak *no* information about security critical detection parameters at all. The protocols presented in [1] fulfil this strong security property: they hide all to-be-secret values in commitments from [8] and compute a commitment on the detection statistic of the underlying watermarking scheme, using the homomorphic-property and zero-knowledge proofs from [3] (see Section 2). Finally, using protocols from [2] the prover proves to the verifier in zero-knowledge, that the committed value of the detection statistic lies above the detection-threshold.[2] The idea underlying this approach is general and easily adaptable to any watermarking scheme that detects watermarks by computing a detection statistic, using operators $+, *, -$, and comparing it to a threshold.

We show the protocol for a well-known blind detection statistic proposed by Cox et al. [4]. For a protocol allowing *non-blind* zero-knowledge detection we refer to [1]. Blind detection of a watermark $WM = (wm_1, \dots, wm_k)$ in a digital image $\overline{O}$ works by computing the correlation value

$$corr = \frac{< DCT(\overline{O}, k), WM >}{\sqrt{< DCT(\overline{O}, k), DCT(\overline{O}, k) >}} \qquad (1)$$

between WM and the k largest DCT-coefficients $DCT(\overline{O}, k) = (DCT(\overline{O})_1, \dots, DCT(\overline{O})_k)$.[3] This value is a measure of confidence for the presence of WM in $\overline{O}$. The watermark is decided to be present in $\overline{O}$ iff $corr \geq \delta$ holds for a predefined *detection-threshold* δ.[4] For efficiency reasons the following equivalent[5] detection criterion is used:

$$C := \quad [(\underbrace{< DCT(\overline{O}, k), WM >}_{A})^2 - \underbrace{< DCT(\overline{O}, k), DCT(\overline{O}, k) > * \delta^2}_{B}] \overset{?}{\geq} 0$$

Zero-knowledge detection assumes that par_{com}, $\overline{O}$, $com(WM) = (com(wm_1), \dots, com(wm_k))$ and δ are common inputs to the prover $\mathcal{P}$ and verifier $\mathcal{V}$. The watermark is hidden in a commitment, to prevent removal by $\mathcal{V}$. Additionally, $\mathcal{P}$ has the secret opening information sk_{com} for $com(WM)$. The protocol allowing $\mathcal{P}$ to prove to $\mathcal{V}$ that the watermark, hidden in commitments $com(WM)$, is blindly detectable in $\overline{O}$ consists of the following steps:

1. $\mathcal{P}$ and $\mathcal{V}$ compute $DCT(\overline{O}, k)$.
2. $\mathcal{P}$ and $\mathcal{V}$ both locally compute part B from the equivalent detection criterion C. Now $\mathcal{P}$ generates a commitment $com(B)$ and sends it to $\mathcal{V}$. By opening $com(B)$ immediately to $\mathcal{V}$ the prover proves that this commitment contains the same (correct) value B which $\mathcal{V}$ computed himself.
3. Then, both $\mathcal{P}$ and $\mathcal{V}$ compute the commitment
$$com(A) := \prod_{i=1}^{k} com(wm_i)^{DCT(\overline{O})_i}$$
by taking advantage of the homomorphic property of the commitment scheme and $\mathcal{P}$ proves to $\mathcal{V}$ in zero-knowledge that $com(A)$ contains a value ≥ 0.
4. $\mathcal{P}$ computes the value A^2, sends a commitment $com(A^2)$ to $\mathcal{V}$ and proves to $\mathcal{V}$ in zero-knowledge (see [3]) that $com(A^2)$ contains the square of the value contained in $com(A)$.

[2] These protocols improve the results from [10], where the watermark coefficients are assumed to be RSA-encrypted and the correlation between the encrypted watermark and the stego-data is computed in a challenge-response manner. The protocol is not zero-knowledge since the verifier obtains a good estimation of the correlation value, thus enabling oracle attacks. Another drawback of the approach from [10] is that it is only applicable to *blind* watermark detection.

[3] Here, $< x, y >$ denotes the scalar product of two vectors x and y.

[4] In contrast to Cox et al., we assume that the watermark, DCT-coefficients and detection threshold are *integers* and not real numbers. Note that this is no real constraint, because we can scale the real values appropriately.

[5] Equivalency holds for $A \geq 0$, which is proven in step 3 of the detection protocol.

5. Now both $\mathcal{V}$ and $\mathcal{P}$ compute the commitment $com(C) := com(A^2)/com(B)$ on the value C.
6. Finally $\mathcal{P}$ proves to $\mathcal{V}$ in zero-knowledge, that the value contained in $com(C)$ is ≥ 0 using protocols from [2]. If $\mathcal{V}$ accepts this proof he can be sure that the watermark hidden in $com(WM)$ is contained in $\overline{O}$.

If any of the local tests or zero-knowledge proofs fails the verifier considers the watermark as being not detectable.

5 Comparison of the Protocols

Comparing the three approaches with regard to security and information leakage, we can give the following results:

Secrecy of inputs:

1. *Secret permutation:* A permuted version $\tau(WM)$ of the watermark WM is given to the verifier as a necessary input. This permutation does not hide the coefficients of WM perfectly, since it reveals characteristics like the minimum/maximum coefficient. However, this may be an advantage in some applications, since it allows certain tests by the verifier, e.g., whether it fulfils necessary statistical properties.
2. *Ambiguity attacks:* A valid watermark WM is hidden among a large number n (security parameter) of fake watermarks $WM_1, \ldots, WM_n$. Here, the secrecy of WM increases only linearly in the security parameter n, which makes a level of security comparable to cryptosystems impossible.
3. *Zero-knowledge watermark detection:* The verifier sees only commitments $com(wm_i)$ on the watermark-coefficients. The hiding property of the commitment-scheme guarantees that no information about wm_i is leaked.

Information disclosure by protocol-runs:

1. *Secret permutation:* The system is susceptible to an oracle attack, which shows that it does not satisfy a zero-knowledge property. By issuing different carefully modified test documents to the prover, τ can be recovered after several (independent) invocations of the protocol (see [7]).
2. *Ambiguity Attacks:* This scheme is susceptible to the following oracle attack, which requires on the average $n/2$ trials: In each step, the verifier removes a watermark WM_i until the prover is unable to present the discrete log of a mark. In this case the verifier knows that he has removed the genuine watermark.
3. *Zero-knowledge watermark detection*: The commitments that the verifier sees during a protocol-run reveal no information about their contents. Neither do the executed sub-protocols, since they are zero-knowledge proof protocols. Thus, due to the composition theorem for the sequential execution of zero-knowledge proofs (see [9]), the whole protocol is zero-knowledge.

6 Conclusion

We discussed the idea of zero-knowledge watermark detection, reviewed some constructions and compared the level of security they provide. The suggested constructions follow different approaches and are applicable to a large class of symmetric watermarking schemes. Many applications of symmetric watermarks can strongly benefit from these protocols. Examples are copyright dispute resolving where the trust necessary in the dispute resolver can be reduced. In proofs of ownership, zero-knowledge watermark detection can help improving the efficiency by allowing offline ownership proofs (see [1]).

Future research may investigate the benefit of zero-knowledge watermark detection in further applications in more detail. Another interesting question is whether there are asymmetric watermarking schemes which achieve a level of security comparable to zero-knowledge watermark detection.

References

[1] A. Adelsbach, A.-R. Sadeghi: Zero-Knowledge Watermark Detection and Proof of Ownership; Information Hiding: Fourth International Workshop, LNCS 2137, Springer, 2001, pp. 273–288.

[2] F. Boudot: Efficient Proofs that a Committed Number Lies in an Interval; Eurocrypt '00, LNCS 1807, Springer, 2000, pp. 431–444.

[3] J. Camenisch, M. Michels: Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes; Eurocrypt '99, LNCS 1592, Springer, 1999, pp. 107–122.

[4] I. Cox, J. Kilian, T. Leighton, T. Shamoon: A Secure, Robust Watermark for Multimedia; Information Hiding, LNCS 1174, Springer, 1996, pp. 185–206.

[5] I. Cox, J.-P. M. G. Linnartz: Some General Methods for Tampering with Watermarks, IEEE Journal on Selected Areas in Communications, Vol. 16, No. 4, May 1998, pp. 587–593.

[6] S. Craver: Zero Knowledge Watermark Detection; Information Hiding: Third International Workshop, LNCS 1768, Springer, 2000, pp. 101–116.

[7] S. Craver, S. Katzenbeisser: Security Analysis of Public-Key Watermarking Schemes; in Proc. SPIE vol. 4475, Mathematics of Data/Image Coding, Compression and Encryption IV, with Applications, 2001, pp. 172–182.

[8] E. Fujisaki, T. Okamoto: Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations; Crypto '97, LNCS 1294, Springer-Verlag, Berlin 1997, pp. 16-30

[9] O. Goldreich: Foundations of Cryptography: Basic Tools; Cambridge University Press, 2001

[10] K. Gopalakrishnan, N. Memon, P. Vora: Protocols for Watermark Verification; Multimedia and Security, Workshop at ACM Multimedia 1999, pp. 91–94.