Public Watermark Detection Using Multiple Proxies and Secret Sharing

Qiming Li

Temasek Laboratories
National University of Singapore
tslliqm@nus.edu.sg

Ee-Chien Chang

Department of Computer Science
National University of Singapore
changec@comp.nus.edu.sg

Abstract. A central issue in public watermarking schemes is the design of a detector that will not reveal sufficient information that leads to the erasure of embedded watermark, even if an adversary knows the detection algorithm and the public detection key (if any). Insofar, there is no such detector in "stand-alone" setting that achieves satisfactory security requirements. Recently, [1] gives a zero-knowledge detector that achieves security by introducing a server. We propose an alternative setting of public watermarking that involves multiple servers. In this setting, the *owner* keeps a secret watermark W. A *verifier*, given an image J (or any digital media), wants to detect whether J is watermarked. The detection is to be carried out by a group of independent *proxies*. The owner does not trust the verifier nor any individual proxy, thus wants to keep W hidden from them. On the other hand, the verifier does not trust the owner and any individual proxy either, and wants to protect himself against cheating. The proxies, as a group, are tasked to maintain the secrecy of the watermark, and protect the interest of the verifier. We propose a scheme based on secret sharing schemes which support arithmetic operations. The security is maintained if not too many individuals (including the proxies, the owner and the verifier) collude. The proposed scheme is efficient in terms of computation cost, and the number of rounds and bandwidth required in the communications. The scheme is arguably easy to implement.

1 Introduction

A central issue in public watermarking schemes [4] is the design of a detector that will not reveal the secret watermarks embedded into the media, even if an adversary knows the detection algorithm and the public detection key (if any).

A possible approach uses asymmetric watermarking schemes [10, 8, 7], where the key to embed watermarks is different from the key required during detection. However, the detection keys of known methods do reveal some crucial information, which leads to a number of successful attacks (for e.g., [7] listed a few attacks on specific schemes).

A simple approach achieves secrecy by introducing a trusted third party P. In which case the owner of the watermark gives his watermark W to P, and

distributes images[1] with W embedded to the public. To check if an image is watermarked, a user sends it to P via the Internet, and the result is sent back from P. In this case both the secrecy of W and the interest of the users are protected assuming that P is honest.

It seems that the trusted third party is crucial to maintain security in the above simple approach. Recently, [1, 6] gave interesting methods to remove the assumption of a trusted third party. The methods employ a *prover* (the owner) who proves the existence of the watermark in given images to *verifiers* (the users). The prover is prevented from cheating by the means of commitment schemes, and the secrecy of the watermark is maintained through zero-knowledge interactive proofs. Although these schemes are cryptographically secure, a main drawback is the large number of rounds and bandwidth required in the communications, and they are not easy to implement in practice. Furthermore, if the owner wishes to designate another party to perform the checking and proving, he has to reveal the secret key to this trusted party.

In this paper, we propose an alternative setting that can be viewed as a modification of the above approach. In this setting, we remove the expensive zero-knowledge interactive proofs without assuming the existence of a trusted third party. Instead, we replace the trusted third party P with a group of *proxies*. Security is maintained if the majority of the proxies are honest.

The individuals in our setting are an owner, a few proxies, and a verifier. At the beginning, the owner generates a secret watermark W and performs a *registration* with the proxies. During registration, the owner distributes some information about the watermark W to the proxies, so that they can carry out watermark detections on their own. After that, the owner embeds the watermark W into his images, which are then released to the public. (The owner could also request the proxies to perform the embedding, without revealing the watermark. We will not describe this operation in this paper.) On the other hand, when a verifier wants to determine if a given image is watermarked by W, he requests the proxies to perform a *detection*. During detection, the verifier sends information about the image to the proxies, who then perform some computations and return the results back to the verifier. Based on these results, the verifier would be able to decide whether the image is watermarked. The registration and detection processes are illustrated in Fig. 1 below.

Since we do not assume the existence of a trusted third party, no individual can be trusted. Therefore, during registration, the owner does not trust any individual proxy, so he cannot simply send W to each of them. Instead, he needs to distribute the watermark in such a way that it is information-theoretically impossible to compute W even if some of the proxies collude. During detection, the verifier does not trust all proxies because some of them might give wrong results, either intentionally or accidentally. Therefore, there has to be some mechanisms to allow the verifier to detect errors, or even correct them. Furthermore, the verifier does not trust the owner either. A dishonest owner might distribute a watermark that correlates with many images (for instance, in the well-known

[1] The "images" here can in fact be any digital contents in any media.

spread spectrum method [5], a watermark with very high energy would likely give a high correlation value with a randomly chosen image). The dishonest owner might also collude with a few proxies to mislead the verifier.

Note that in this setting, the role of the proxies (as a group) is similar to that of a trusted third party, who will not leak any information of the secret W, and will not cheat the verifier. The main difference is that, in this setting, we only require the integrity of the proxies as a group, which is a much weaker requirement than having a trusted third party. It is also noted that our setting relieves the owner from performing the detections. This is a secondary advantage of using proxies for detection.

The proposed setting naturally suggests the use of secret sharing schemes [15] as a basic building block. A secret sharing scheme breaks a secret z into *shares* and distributes each to a server. No individual server will know the secret unless a number of dishonest servers collude. There are many secret sharing schemes, satisfying various useful properties. For example, Shamir's scheme is also a threshold scheme, and with further modifications it can be verifiable [3, 9] and proactive [11]. An important property required in our setting is that, both multiplications and additions can be supported on the shares. That is, if secrets z_1, z_2 and z_3 are integers and are shared among n severs, the shares of $z_1 z_2 + z_3$ can be generated without revealing the values of z_1, z_2, z_3, and $z_1 z_2 + z_3$.

We give a scheme based on secret sharing. This scheme achieves public watermarking as long as not too many individuals collude. This scheme is arguably easy to implement and is efficient in terms of computation and communication cost.

Outline. In Section 2.1, we describe the basic watermarking method (spread-spectrum method) used for discussion. Section 2.2 gives our proposed multiple proxies setting, and the security requirements. Section 3 gives a brief description of secret sharing schemes. Our scheme is described in Section 4, followed by the security analysis in Section 5. Some discussions on the error-correcting capability of the scheme will be given in Section 6.

2 Notations & Model

2.1 Watermarking Model

We employ a variant of the well-known spread spectrum method [5] to embed and detect watermarks. Other watermarking schemes can also be employed as long as the detection involves only multiplication and addition.

Our images and watermarks are "discretized". An image I is a vector $I = (x_1, x_2, \ldots, x_m)$ where each $x_i \in \{0, 1, 2, \ldots, d-1\}$ and d is some integer determined by the media/image representation. For example, d could be 256 if each x_i represents a pixel. The watermark W is also a vector $W = (w_1, w_2, \ldots, w_m)$ where each $w_i \in \mathbf{Z}$ is an integer. In addition, the energy of the watermark W is fixed, that is $W \cdot W = E$ where E is some predefined threshold, and $\cdot$ is the vector inner product. The constant E is made known to the public.

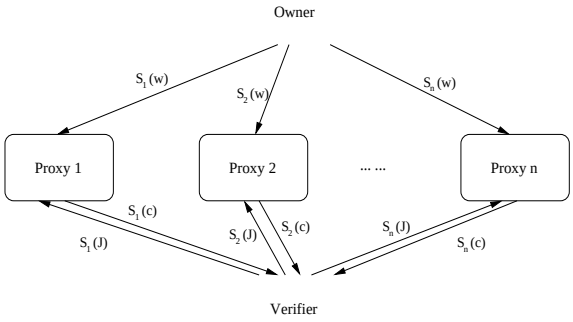

Fig. 1. The proposed setting.

During embedding, given an image I and the watermark W, the watermarked image $\widetilde{I}$ is

$$\widetilde{I} = \text{trunc}(I + W),$$

where the function trunc() truncates/rounds the coefficients where values are not in the range $\{0, 1, 2, \ldots, d-1\}$.

During detection, given an image J, the correlation value $(J \cdot W)$ is computed. If the correlation value exceeds certain threshold, then J is declared to be watermarked.

Note that we omit the normalization of images in the embedding and detection. Normalization is not required in our discussion, but still can be incorporated if required. Thus, it is omitted for simplicity.

2.2 Owner, Proxies & Verifier

The individuals in the proposed setting are an owner, n proxies $P_1, P_2, \ldots, P_n$, and verifiers. The number of verifiers is not important, so we assume that there is only one verifier.

During the *registration*, the owner generates a secret watermark W which satisfies $(W \cdot W) = E$, where E is a constant that every individual knows. The owner sends information of W to the n proxies. Let $S_i^{(0)}(W)$ be the data the owner sends to the proxy P_i. Let $S_i(W)$ be the data that P_i keeps and uses in subsequent computations. Note that it is not necessary that $S_i^{(0)}(W) = S_i(W)$. To guard against dishonest owner, the proxies might wish to transform the original $S_i^{(0)}(W)$'s distributed by the owner.

During the *detection*, a verifier wishes to know the correlation value $c = J \cdot W$ of a given image J with the secret watermark W. Firstly, the verifier splits J into n pieces, $S_i(J)$, $1 \leq i \leq n$, such that each of them contains partial information about J. Then it sends $S_i(J)$ to proxy P_i respectively. Next, each proxy P_i computes and sends the verifier partial information $S_i(c)$ of the correlation value

c. After receiving data from all proxies, the verifier reconstructs the correlation value $(J \cdot W)$ (Fig. 1).

In our proposed scheme, the partial information communicated through the network, namely $S_i^{(0)}(W)$, $S_i(W)$, $S_i(J)$ and $S_i(c)$, corresponds to the "shares" in secret sharing schemes (see Section 3). Therefore, we will refer to those pieces of information as shares in the rest of this paper, even if some of them may be fraud sent by dishonest individuals.

2.3 Security Requirements

A parameter for the security requirement is the *security threshold* t where $t \leq n$. Vaguely, the scheme should tolerate at most $(t-1)$ dishonest individuals (including the proxies, the verifier and the owner). The security requirements can be roughly classified into (a) maintaining the secrecy of W, and (b) protecting the interest of the the verifier. In the following, the first requirement $S1$ belongs to the first class and the remaining belong to the second class.

S1. Secrecy of W. The owner generates and keeps the watermark W. Recall that the energy $(W \cdot W)$ is a predefined constant E, which is known by everyone. For any $(t-1)$ proxies, even if they collude, they should not know W. Specifically, to any group of $(t-1)$ proxies, any vector W' satisfying $(W' \cdot W') = E$ is a possible candidate for the secret watermark.

On the other hand, after detection, from the n sets of data $S_i(c)$'s obtained, the verifier should not know W. Specifically, to the verifier, any vector W' satisfying $(W' \cdot W') = E$ and $(J \cdot W') = c$ is a possible candidate for the secret watermark.

In general, any $(t-1)$ individuals, including the proxies and verifier, should not know W. That is, by combining all the data held by them, any vector W' satisfying $(W' \cdot W') = E$ and $(J \cdot W') = c$ is a possible candidate for the secret watermark.

S2. Dishonest owner during registration. During registration, instead of honestly sending $S_i(W)$ to the proxies, an owner may send other values so as to mislead the proxies to give high correlation value during detection. Here is an example of dishonest owner: the dishonest owner chooses a watermark w and embeds it into images according to the scheme, but registers with the proxies another watermark Aw where A is a very large constant. Thus, in the subsequent detections, whatever correlation value determined will be based on Aw, instead of w. Owe to the large value of A, the probability that a randomly chosen image is wrongly declared as watermarked is higher. Therefore, we require that, after registration, with at most $(t-1)$ dishonest proxies, any malicious behaviour of the owner can be detected. Specifically, the proxies can check that, indeed, $(W \cdot W) = E$.

S3. Dishonest proxies during detection. During detection, some of the proxies may collude so as to mislead the verifier. Thus, we require that, if at most $(t-1)$ proxies are dishonest, the verifier can detect that.

S4. Collusion among proxies and the owner during detection. A more interesting case is collusion among the owner and proxies. The owner may collude with a few proxies and mislead the verifier. The main difference of this case from the case in previous paragraph is that: here, the owner can reveal the watermark to the proxies. With this extra information, it is easier for the proxies to influence the detection. Thus, we require that, if the owner colludes with at most $(t-2)$ proxies (so the total number of dishonest individuals is at most $t-1$), the verifier should be able to detect that.

S5. Collusion among the owner, proxies and a verifier. It is interesting that we should consider collusion among dishonest owner and verifier. The owner may collude with the verifier and a few proxies, so as to obtain the $S_i(W)$ held by an honest proxy P_i. After obtaining the information, they can use it to influence subsequent detections. Thus we require that, by combining information from a verifier, $(t-3)$ proxies, and the owner, no sufficient information on data held by honest proxies can be derived.

Remarks. Note that currently we do not consider verifier/proxies who keep the history of communications. For example, a verifier who probes the proxies by sending in a series of images. This type of attacks is generally known as sensitivity attacks. We will address this in Section 7.

2.4 Requirement on Error-Correcting

A secondary requirement is error-correcting capability. More specifically, even if some proxies are dishonest or failed, detection can still be carried out. Note the difference between security and error-correcting capability. A scheme that immediately shuts down when malicious activities are detected is considered to be secure, but it is not capable of correcting errors. We say that the *error-correcting threshold* of a scheme is R, if all detection operations can be carried out when there are at least R honest proxies.

3 Backgrounds on Secret Sharing Schemes

A (t, n) secret sharing scheme splits a secret (for example, a binary file) into n pieces, which are referred to as *shares*. Then the shares are distributed to n servers respectively. The knowledge of any $t-1$ shares will not reveal the secret and the secret is reconstructible by putting together any t shares. When $t < n$, it is also known as a t out of n *threshold* scheme. Shamir gave such a scheme in 1979 [15]. For a secret $z \in \mathbf{Z}_p$, where p is a large enough prime known to everyone, the share for the i-th server is $f(i) \pmod p$ where $f(x)$ is a random polynomial of degree $(t-1)$ whose free coefficient $f(0) = z$. No individual server knows the coefficients of $f(x)$, thus any $t-1$ servers can not derive z from their shares. However, if t servers put their shares together, they can solve for the coefficients of $f(x)$ and thus reconstruct the secret z.

Shamir's scheme can be modified to achieve useful properties. For example, the schemes can be verifiable [3, 9] and proactive [11]. It is also known that certain arithmetic operations of the secrets can be performed on their shares, such that the shares of the result can be obtained without revealing any of the secrets [2, 12]. In our scheme, we mainly make use of arithmetic operations on the shares. Proactive schemes can also be employed to enhance security.

3.1 Notations on Secret Sharing

A secret is an integer in $\mathbf{Z}_p$ where p is a prime. In this paper, all arithmetic operations (multiplications and additions) performed are followed by modulo p. Thus, for simplicity, we omit the notation (mod) when writing an arithmetic expression. For example, we simply write $z_1 + z_2 z_3 \pmod{p}$ as $z_1 + z_2 z_3$.

With respect to a secret sharing scheme, let $S_i(z)$ be the i-th share of z, the secret [2]. Let $S(V)$ be the i-th share of a vector $V = (v_1, v_2, \ldots, v_m)$. Note that the "secret" in the secret sharing scheme is an integer, whereas the watermark W and image I are vectors. To compute $S(V)$, we treat each v_i as an independent secret. That is, $S_i(V) = (S_i(v_1), S_i(v_2), \ldots, S_i(v_m))$. Since the shares are associated with the proxies, we also call $S_i(V)$ the share of V for the proxy P_i.

3.2 Arithmetic Operations on Shares

Consider two secrets α and β, which are encoded by $f(x)$ and $g(x)$ respectively as in Shamir's scheme, and the shares are distributed to $2t-1$ servers. Now, suppose the servers want to compute the shares of $\alpha + \beta$ without revealing the values of α, β or $\alpha + \beta$. This can be easily done by instructing each server to locally construct the new share by adding the two shares it holds.

The shares for $\alpha\beta$ can be computed similarly by computing $s_{i,1}s_{i,2}$, which is the share of $\alpha\beta$ encoded by $k(x) = f(x)g(x)$. However, the degree of $k(x)$ is raised to $2t-2$, thus $2t-1$ servers are required to reconstruct the secrets.

In our application, instead of general combinations of multiplications and additions, we require only inner products. That is, given the shares of $x_1, x_2, \ldots, x_m$, and $v_1, v_2, \ldots, v_m$, we want to compute the shares of the inner product $c = \sum_{i=1}^{m} x_i v_i$, without revealing the secrets $x_1, x_2, \ldots x_m$, $v_1, v_2, \ldots, v_m$, and the inner product c. For each server, its share of c can be easily computed locally by simply computing the summation of products on its shares of x_i, v_i's.

4 Public Watermark Detection using Secret Sharing

The multiple proxies setting naturally suggests the use of secret sharing as a basic construction block.

[2] Note that the shares are computed based on some randomly chosen numbers. Thus to be more precise, we should write $S_i(R, z)$ for the share where R is the chosen sequence of random numbers. For simplicity, we omit R in the notation.

Assume that there are n proxies. Suppose the security threshold we want to achieve is t, that is, if not more than $(t-1)$ individuals collude, the security is maintained. We also require that $(2t-1) \leq n$.

We choose a (t, n) secret sharing scheme where $(2t-1) \leq n$. Recall that n is the number of proxies, and $(t-1)$ is the number of dishonest individuals the system can tolerate.

The scheme consists of two parts: registration and detection.

REGISTRATION

§1. ***Distributing watermark.*** The owner, using the secret sharing scheme (with the notations defined in Section 2.2 and 3.1), computes $S_i^{(0)}(W)$, the share of W for each proxy P_i, $1 \leq i \leq n$. The owner then sends $S_i^{(0)}(W)$ to P_i secretly for all proxies.

§2. ***Refreshing the shares.*** After receiving the shares from the owner, the proxies refresh the shares of W using the mechanisms described in Section 3. At the end of this step, each proxy P_i has $S_i(W)$ as a new share of W, and old shares $S_i^{(0)}(W)$'s are discarded.

§3. ***Checking W is genuine.*** Each proxy P_i computes the value $(S_i(W) \cdot S_i(W))$, and broadcasts it to other proxies. Note that this value is also the share of the inner product $(W \cdot W)$ for each proxy. After receiving all data from other proxies, each proxy reconstructs $(W \cdot W)$ and confirms that indeed $(W \cdot W) = E$. If not, the registration fails.

The detection is initiated by a verifier. The verifier wants to know whether an image J is embedded with the watermark claimed by the owner.

DETECTION

§1. ***Distributing the image.*** The verifier computes the shares of J and sends the shares to the respective proxies.

§2. ***Computing the shares of the correlation value.*** Each proxy P_i computes the inner product $(S_i(J) \cdot S_i(W))$ and sends it back to the verifier.

§3. ***Reconstructing the correlation value.*** After receiving all the shares, the verifier reconstructs the correlation value $(J \cdot W)$. Recall that $(2t-1)$ shares are necessary and sufficient for reconstruction. Therefore, if $(2t-1) = n$, there is only one possible way to reconstruct such value. Otherwise, $(2t-1) < n$, and there are more than one group of $(2t-1)$ shares. For each group, the verifier reconstructs the value.

§4. ***Checking for corrupted data.*** Since the error-correcting threshold is $(2t-1)$, all proxies must be honest if $(2t-1) = n$. In this case, the only value the verifier reconstructed must be correct. Otherwise, the verifier checks whether there is any inconsistency among the values reconstructed from different groups of $(2t-1)$ shares. If so, it declares that some proxies are cheating.

To enhance security, the proxies can refresh their shares regularly. For example, after some number, say $m-1$ of detections have been carried out, the proxies can refresh the shares $S_i(W)$. This is to guard against sensitivity attacks. We will revisit these issues later.

5 Security Analysis

We want to show that the proposed scheme satisfies the requirements stated in Section 2.3. The requirements are generally in this form: if at most $(t-1)$ individuals collude, then either no extra information on W is revealed, or no sufficient information is revealed so that the colluders can manipulate the results.

In the following analysis, we treat each share as an equation, where the unknowns are the random numbers used to generate the share. To illustrate, consider a proxy P_i who is holding the share $S_i^{(0)}(W)$, and it wants to guess the watermark W. This share is generated by the owner using $(t-1)m$ random numbers (note that W is a vector of m coefficients). For example, let us consider only the first coefficient w_1, the proxy P_i can expresses what it has as the equation

$$S_i^{(0)}(w_1) = w_1 + r_1 i + r_2 i^2 + \ldots + r_{t-1} i^{t-1}$$

where $w_1, r_1, \ldots, r_{t-1}$ are the unknowns. If a proxy manages to gather t such equations or more, he would be able to solve for w_1. Otherwise, **any** value is a possible candidate for w_1.

Note that the security is achieved unconditionally. That is, even if the colluders have infinite computing power, they can not compute the secret. This is in contrast to schemes that are *computationally secure*, where the security is based on the assumption that certain problem is computationally difficult to solve. We will omit the details for security requirements ***S4*** and ***S5***.

S1. Secrecy of W. First, we investigate the case where all the $(t-1)$ colluders are proxies. Without loss of generality, let the colluders be the proxies $P_1, P_2, \ldots, P_{t-1}$. Note that each P_i has the shares $S_i(W)$ and $S_i^{(0)}(W)$, and all proxies know $S_j(E)$ for all j, and that $(W \cdot W) = E$. Now, we want to know whether combining the information from $(t-1)$ proxies will reveal additional information on W.

Let us consider only $S_i^{(0)}(W)$ first. For each coefficient w_i in vector W, $t-1$ random numbers are used to generate the shares. For w_i, there are t unkowns. On the other hand, the corresponding entry in each $S_i^{(0)}(W)$ proxy P_i possesses is equivalent to 1 equation. Therefore, for $t-1$ proxies, there are only $t-1$ equations, and any integer (in $\mathbf{Z}_d$) is a possible solution for w_i, as shown in [15]. This shows that these equations do not give the proxies any advantages in computing W. Furthermore, the new shares $S_i(W)$ are obtained after refreshing, and no information about W or $S_i^{(0)}(W)$ is exchanged among the proxies during this process. Therefore these values do not give any advantages in computing W either. Lastly, after computing $S_j(E) = S_i(W) \cdot S_i(W)$, information about the

elements of $S_i(W)$ is hidden in the inner product, given that m is sufficiently large (which is true for most practical applications).

Next, suppose $(t-1)$ colluders are the verifier and $(t-2)$ proxies $P_1, \ldots, P_{t-2}$. The verifier knows the shares $S_i(c)$ for all i, where $S_i(c) = S_i(W) \cdot S_i(J)$. Similar to the above argument, given sufficiently large m, the information contained in the inner product is useless in attempts to obtain W.

S2. Dishonest owner during registration. The owner could be dishonest and try to mislead the proxies so that they give false results in the detection. For instance, he could give a false watermark with high energy, so that the correlation value of the watermark and any randomly chosen image would be large with high probability. This is prevented in the registration because the proxies compute the energy of the watermark and compare it to a known constant E (Step §3 in registration). If the energy is not E, the watermark would be rejected by the proxies, and the registration would fail.

S3. Dishonest proxies during detection. The proxies could also send false results of their inner products $S_i(W) \cdot S_i(J)$ to mislead the verifier. However, since we have more than $2t-1$ proxies in the system, we can perform reconstruction of $J \cdot W$ multiple times from different set of shares (Step §4 in detection). It is unlikely that the results are consistent if some proxies cheat. In this case, we can employ the method mentioned in Section 6 to both detect and correct the error.

6 Analysis on Error-Correcting

Besides the security requirement that no more than $t-1$ proxies collude, we also require that the verifier can detect errors from proxies and correct them if there are at least $(2t-1)$ honest proxies [3].

Here are two methods of error correction. The first method is to let the verifier compute a new image $J' = kJ$, where k is some integer chosen by the verifier. If the proxies are honest, the resulting correlation value $c' = J' \cdot W = k(J \cdot W)$ would have the integer k as its factor. If some proxies are dishonest, it is highly unlikely that the reconstructed correlation will still have k as its factor.

The other method let the verifier repeat the detection using the same image J, but using different random numbers to generate the shares of J. Thus, a group of $2t-1$ proxies would be able to give consistent results only if all of them are honest. By repeating the detections, the correct results can be obtained with arbitarily high probability. It is noted that the above two methods can be used together.

7 Sensitivity Attacks

A dishonest verifier might probe the proxies for the watermark. By designing the probes carefully, it may be able to get a good approximation of, or erase, the

[3] If an accidental error happens, say, during network transmission from a proxy to the verifier, we consider it as a dishonest behaviour (of the proxy).

watermark, using small numbers of probes. This is generally known as sensitivity attacks. Some general attacks are given in [4, 13]. Practical attacks usually target at the image representation. For example, the well-know Stir-Mark provides a list of attacks [14].

We classify these attacks into two types. The first type is specific to our proposed scheme and not applicable to others schemes, for instance the zero-knowledge detector [1]. The second type of attacks are designed for general public watermarking schemes. For example, the attacks described in [4, 14, 13]. In this analysis, we focus on the first type. Further research is required to handle the second type of attacks.

Let us consider a dishonest verifier. The verifier may collude with $(t-2)$ proxies in attempt to get the secret watermark. Let $S = (s_1, s_2, \ldots, s_m)$ be the shares of W kept by an honest server. In each detection, the verifier knows the inner product of $S \cdot V$ where V is some vector chosen by the verifier. Although knowing $S \cdot V$ will not reveal any useful information of the watermark W, by sending in many different vectors, the verifier can determine S. If the verifier knows the inner product of $S \cdot V_i$ for $i = 1, \ldots, m$ and the V_i's are independent, then the verifier can solved for S. By knowing the shares in t proxies, the verifier can solve for W. Note that this attack is specific to our scheme. To prevent this, we can require the proxies to refresh their shares regularly, for example, after every $m-1$ detections.

8 Communication Cost

We measure the communication cost by the number of rounds of communication and the amount of data transmitted. The size of a coefficient is not more than $\lceil \log p \rceil$ bits, where p is the prime used in the secret sharing scheme. Note that we only require $p > n$ and p is larger than the range of the original image coefficient. Thus, it is not required to be very large. In contrast, for the zero-knowledge detector in [1], the size of one coefficient has to be large (for e.g., more than 200 bits), so that it is computationally infeasible to break the commitment scheme.

Let us assume that the size of each coefficient is 1 unit. Thus, the size of W and J is m. The size of each share is also m. During detection, the verifier sends the share $S_i(J)$ to each proxy P_i, and P_i returns the share $S_i(c)$ of the correlation value. Thus, only 1 round of communication is required. Since the size of each share $S_i(J)$ is m, and the size of each share $S_i(c)$ is 1, the total amount of data transmitted is $(mn + n)$. The zero-knowledge detector in [1] invokes an interactive proof protocol during detection. Due to the "probabilistic" nature of interactive proof, many rounds are required for high level of confidence.

Higher communication cost is required during registration. This is due to the communication required in refreshing. Fortunately, registration is only performed once for each watermark. During registration, refreshing without verification can be done in 1 round with mn^2 units of data. If verification is required, then the

communication cost depends on the commitment schemes and the interactive proof protocol employed.

9 Conclusion

We propose a setting of public watermark detection using multiple proxies. The owner registers its watermark with a group of proxies, whereas the verifier contacts the proxies to check whether an image is watermarked. We give such a scheme based on secret sharing. As long as not too many individuals collude, the secrecy of the watermark can be maintained, and the verifier can be protected from cheating. In other words, public watermark detection is achieved by the integrity of the community. The scheme is efficient in terms of communication cost and is arguably easy to implement.

References

[1] A. Adelsbach and A. Sadeghi. Zero-knowledge watermark detection and proof of ownership. *4th Int. Workshop on Info. Hiding*, LNCS 2137:273–288, 2000.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, 1988.

[3] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS*, pages 383–395, 1985.

[4] I.J. Cox and J-.P. Linmartz. Public watermarks and resistance to tampering. *IEEE Int. Conf. on Image Processing*, 3(0_3–0_6), 1997.

[5] I.J. Cox, M.L. Miller, and J.A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2002.

[6] S. Craver and S. Katzenbeisser. Copyright protection protocols based on asymmetric watermarking. In *CMS'01*, pages 159–170, 2001.

[7] J.J. Eggers, J.K Su, and B. Girod. Asymmetric watermarking schemes. *Sicherheit in Mediendaten*, September 2000.

[8] J.J. Eggers, J.K Su, and B. Girod. Public key watermarking by eigenvectors of linear transforms. *European Signal Processing Conference*, September 2000.

[9] P. Feldman. A practical scheme for non-interaive verifiable secret sharing. In *FOCS*, pages 427–437, 1987.

[10] T. Furon and P. Duhamel. An asymmetric public detection watermarking technique. *3rd Intl. Workshop on Information Hiding*, LNCS 1768:88–100, 2000.

[11] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *CRYPTO'95*, LNCS 963:339–352, 1995.

[12] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *ASIACRYPT'00*, volume LNCS 1976, pages 143–161, 2000.

[13] Qiming Li and Ee-Chien Chang. Security of public watermarking schemes for binary sequences. *5th Int. Workshop on Info. Hiding*, pages 119–128, 2002.

[14] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Attacks on copyright marking systems. *2nd Intl. Workshop on Information Hiding*, LNCS 1525:219–239, 1998.

[15] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.