An Improved Asymmetric Watermarking System Using Matrix Embedding

Scott Craver

Department of Electrical and Computer Engineering
Binghamton University

Abstract. In the asymmetric watermarking problem, we wish to embed a signal in a piece of multimedia and later prove that we have done so, but without revealing information that can be used by an adversary to remove the signal later.

There have been several published solutions to this problem, which suffer from the twin problems of size and time complexity. We provide a protocol similar to the one described in [5], but with substantial improvements in time and space. Our protocol is non-interactive, and each exchange of information between the prover and verifier involves a smaller payload of data. The algorithm uses a form of matrix embedding with pseudo-random Gaussian matrices. Aside from an improvement in efficiency, it possesses other practical advantages over previous protocols.

1 Introduction

Asymmetric watermarking refers to the problem of embedding a signal, and proving that it has been embedded, without actually revealing it or otherwise revealing information allowing an attacker to damage the hidden data. Several protocols for accomplishing this have been published, by placing detection within the framework of a zero-knowledge protocol [5,4,1] or by creating statistical artefacts which can be exhibited without revealing the signal [10].

Some of these systems suffer from inversion attack vulnerabilities: the attacker can simply pick a signal that correlates with the image, and prove asymmetrically that he knows it [6]. Because the prover is no longer allowed to reveal the watermark, verifying that it is a valid watermark becomes difficult. Those which are not vulnerable still have high space and time complexity, due to interactive protocols in which amounts of data comparable to the multimedia data itself are exchanged repeatedly.

In this paper we outline an asymmetric embedding method using pseudo-random Gaussian matrix embedding. A key K is used to generate a pseudo-random matrix $G[K]_{M\times N}$. Given a data vector $x_{N\times 1}$ extracted from the multimedia object, we tune in to one "channel" of the data by computing the product $y = G[K]x$. Embedding consists of determining the watermark signal w such that $\hat{y} = G[K](x + w)$ is the message we want to transmit. In practice we will embed multiple messages in multiple channels using multiple keys K_i by solving one system of linear equations.

J. Camenisch et al. (Eds.): IH 2006, LNCS 4437, pp. 15–25, 2007.

The asymmetric part uses the inversion attack principle described in [5] and [4]. There, the authors embed a few watermarks in an image and use inversion attacks to generate a large number of counterfeit ones. All of the watermarks are detectable in the test image, and a zero-knowledge proof is used to show that at least one watermark is real. An attacker who wants to render the real watermarks undetectable can attempt to damage a large number of the counterfeit marks; but since these counterfeit marks *are the image's content,* this amounts to damaging the image significantly.

In our algorithm, we embed a few watermarks using specific keys K_i, and supply a prover with a large number of extra random keys. Extracting a signal using most of these keys will result in channel noise, which serves the same role as the counterfeit watermarks of the previous protocol.

1.1 Why This New Method Is an Improvement

There are several factors that make this new system an improvement over the previous protocol. First, in this new protocol Alice only needs to provide Bob with a list of generating keys, not a list of watermarks themselves. In [5], the watermarks are as large as the watermarked data vector X, and hundreds must be provided as part of the proof. Owing to the construction of that secret watermark vector, Alice cannot provide the PRNG seed that generates the watermarks—doing so would reveal crucial information about how the watermarks are created, revealing which are real and which are fake. In our algorithm, Alice can provide Bob with a list of embedding keys which generate the watermarks. This results in a considerable size reduction in representing the watermark data. Instead of providing a bundle of watermarks M times the size of the "image," we can provide a bundle of keys M times some small size, such as 56 bits. The keys do not even need to be large enough to prevent brute force, because they are not kept secret. They are simply pseudo-random seeds.

Second, our verification protocol is non-interactive. This is not a trivial form of non-interactive protocol in which the same data payload is sent all at once; rather, we provide a single-step protocol whose payload is comparable to a single step of the interactive one. This provides a substantial reduction in protocol traffic.

Another advantage of our system is that Alice does not need to fix the number of counterfeit watermarks in advance. The protocol in [5] requires the image to be divided into shares in advance, that all counterfeit watermarks are constructed at the same time as the new watermarks are added. In our protocol, Alice need not perform any inversion attacks during embedding; she can invent as many false watermarks as she wants during verification.

A final advantage of our system is that the image data size is not strictly bound to the payload size. In other words, the cryptographic objects we embed do not have to be too large or too small owing to the size or nature of the data.

In [1], watermark vector elements are blinded by using them as exponents, for example $B_i = a^{w_i} \pmod{p}$. Cryptographic protocols have certain requirements for the bit-sizes of this data; a watermark vector element w_i needs to have