A Cryptographic Method for Secure Watermark Detection

Michael Malkin[1] and Ton Kalker[2]

[1] Stanford University,
Stanford, CA, USA
mikeym@cs.stanford.edu
[2] Hewlett-Packard Laboratories
Palo Alto, CA, USA
ton.kalker@hp.com

Abstract. We present a semi-public key implementation of quantization index modulation (QIM) watermarking called Secure QIM (SQIM). Given a signal, a watermark detector can learn the presence of an SQIM watermark without learning anything anything else from the detection process. The watermark detector first transforms the signal with a secret transform, unknown to the detector, and then quantizes the transform coefficients with secret quantizers, also unknown to the detector. This is done with the use of homomorphic cryptosystems, where calculations are performed in an encrypted domain. A low-power, trusted, secure module is used at the end of the process and reveals only if the signal was watermarked or not. Even after repeated watermark detections, no more information is revealed than the watermarked status of the signals. The methods we present are for watermark systems with quantizers of stepsize 2.

1 Introduction

When watermarking occurs for the purposes of digital rights management (DRM), watermark embedding is performed in a trusted environment, while watermark detection is performed "in the wild". That is, the watermark detector is assumed to be a trusted party, but it is generally operating in a hostile environment where the end-user would like to circumvent the DRM. One way to keep the watermarking secret and functionality out of the hands of hostile parties is to embed it in a physically secure device, such as a smartcard, which can be operated in a black-box manner. However, such devices generally have very low computing capacity, and will be unable to perform watermark detection very quickly on their own. Our strategy, also proposed in other papers, is to have the watermark detector work in an encrypted domain and use a trusted secure device, called the *secure module*, to finish the detection process.

Secure QIM (SQIM) uses public and private keys much like public key cryptosystems such as RSA. The private key is used by the watermark embedder to generate watermarks while the public key is used by the watermark detector to

perform watermark detection in an encrypted domain. Finally, the secure module uses the private key to decrypt the results produced by the watermark detector. The secure module must be initialized by communication with the watermark embedder to receive the private key information. In a sense, this system is not truly asymmetric but rather semi-asymmetric, since the aid of a trusted third party (the secure module) is required.

Our first goal is to ensure that the act of detecting a watermark reveals as little information as possible to the watermark detector. Because the secure module is low-power and low-bandwidth, our second goal is to ensure that the watermark detector takes on as much of the computational burden as is possible, and transmits a few bits to the secure module as possible. Two cryptosystems are used to allow the watermark detector to perform the necessary calculations without learning any information about the watermarking secrets. These systems are *homomorphic*, meaning that an operation performed on ciphertexts corresponds to another operation performed on plaintexts. For example, in the Paillier cryptosystem (see Section 3.1), if $E(\cdot)$ is the encryption function, then $E(x)E(y) = E(x + y)$. The homomorphic properties of these cryptosystems are what make it possible for the watermark detector to run the algorithm without learning anything.

However, even though the watermark detector gains no extra knowledge through the detection process, knowledge of the presence or absence of watermarks is sufficient to mount *oracle attacks* (see Cox and Linnartz [5], Venturini [18], and Li and Chang [14], for example). The purpose of these attacks is to find the boundary separating watermarked signals from non-watermarked signals, and use this boundary to learn the watermarking secret. Such attacks are much more powerful than attacks on the cryptosystems presented in this paper, and are possible whenever a watermark detector can test signals for watermarks.

One defense against oracle attacks is to increase the time required for watermark detection, effectively limiting the speed of the "oracle" (See Venturini [18]). This would not be possible with a fully assymetric watermarking scheme, since the speed of such a watermark detector would be limited only by the speed of the machine that is running it. A trusted secure module could have a built-in delay, or a limit on the number of watermark detections per minute, and could thereby help to slow the rate of convergence of oracle attacks.

On the other hand, the use of a secure module introduces side channel attacks, for example timing attacks (see Kocher [10], and Brumley and Boneh [2]), and power attacks (see Kocher et al. [12]). In these attacks, the secure module is monitored externally to guess at the operations occurring internally. An implementation of SQIM would have to take side channel attacks into account, but a detailed discussion of these attacks is beyond the scope of this paper.

Quantization Index Modulation (QIM), developed by Chen and Wornell [4], embeds a watermark into an signal by manipulating the signal so that transform coefficients are quantized in a specific manner. A watermark detector transforms a signal and checks to see if the transform coefficients are appropriately quantized. There are two phases to securely detecting a QIM watermark. First

a *hidden transform* is performed on the signal, and second the transform coefficients are quantized via *hidden quantization*. After hidden quantization the secure module counts the number of watermarked transform coefficients and reveals whether a threshold of the coefficients were watermarked. The method presented in the paper only works for quantizers with stepsize 2.

Attempts have been made at completely asymmetric watermarking schemes (see Eggers et al. [6] and Hachez and Quisquater [7]), but these have generally not been completely successful. Another specific method of performing asymmetric watermarking involves multi-round zero knowledge proofs (see Adelsbach and Sadeghi [1], for example). Kalker [8] introduced the idea of using a secure module to enable semi-public key watermarking, using a variant of the Paillier cryptosystem to perform secure spread spectrum watermarking. In comparison with the spread spectrum scheme, a SQIM scheme must implement a nonlinear operation in an encrypted domain, namely quantization. For more discussion of secure watermarking and the SQIM system, see Malkin [11].

Section 2 outlines the QIM watermarking scheme. Section 3 reviews homomorphic cryptography and introduces the two cryptosystems used in this paper. Section 4 discusses how to perform a hidden transform, while Section 5 discusses how to perform hidden quantization. Section 6 presents the full Secure QIM system. Finally, in Section 7 we discuss the efficiency of SQIM and in Section 8 we prove that SQIM is zero knowledge and that it is secure.

2 QIM

We consider a simple variant of QIM with dithered scalar quantizers. Our purpose is not to improve the watermarking aspects of QIM, but to ensure that watermark detection is secure. Therefore, watermark embedding is not changed at all, and watermark detection is changed only in that all calculations are performed in a secure manner. We will discuss watermark detection first, and then explain how a watermark is embedded into a signal. We are only concerned with whether or not a signal was watermarked, so we do not use the watermark to embed data into a signal.

To detect a watermark in a signal, the signal is first transformed with a secret, random linear transform, for example a DCT or wavelet transform. For every transform coefficient t_i, there are two secret quantizers, Q_0^i and Q_1^i. If t_i is closer to a quantization point on Q_0^i, then it corresponds to 0, otherwise it corresponds to 1. In this way, each transform coefficient is assigned a value. If a threshold of transform coefficients have the correct, watermarked value, then the signal is considered to be watermarked. If not, the signal is considered not watermarked.

Embedding a watermark into a signal involves changing the signal so that the correct values are obtained after quantizing the transform coefficients. The most straightforward approach is to transform the signal, quantize the transform coefficients, and perform the inverse transform. Other embedding schemes, such as distortion-compensated QIM [4], may also be used.

3 Homomorphic cryptosystems

We use two cryptosystems, the Paillier cryptosystem and the Goldwasser-Micali cryptosystem, both of which are *probabilistic public-key* cryptosystems. They are public key in that a public key is used to encrypt plaintext, while a private key is needed to decrypt a ciphertext, and the two keys are computationally not easily derived from each other. They are probabilistic in the sense that the same plaintext is represented by a large number of ciphertexts. This is important when the range of possible plaintexts is small. For example, when encrypting 0 or 1, a non-probabilistic cryptosystem can produce only 2 possible ciphertexts, whereas a probabilistic cryptosystem can produce many different ciphertexts.

This last property is especially important in the current application. For example, if samples were in the range $[0, ..., 255]$, then there would be only 256 possible encryptions of the samples. This would make it much easier to break the system by looking at the transcripts of many watermark detections. Even relabelling the sample values would not solve the problem; there would be 256! possible relabellings, but statistical analysis could be used to easily find the correct one. With probabilistic cryptosystems, the values would be effectively *blinded*, so that this essentially brute-force searching attack would not be possible.

Both of these cryptosystems share another important property: they are homomorphic. This means that a mathematical operation performed on ciphertexts corresponds to a mathematical operation performed on plaintexts. For example, if $E(\cdot)$ corresponds to encryption in the Paillier cryptosystem, then we can write the homomorphism of the Paillier cryptosystem as

$$E(a_1)E(a_2) = E(a_1 + a_2).$$

Homomorphic cryptosystems enable the watermark detector to perform calculations without explicitly knowing what is being calculated or finding out the results of the calculation. For example, given $\alpha = E(a)$, but not knowing the value of a, we could compute the encryption of $7a + 3$ as

$$\alpha^7 E(3) = E(7a + 3).$$

Furthermore, with the right public values, it would be possible for us to compute, in the encrypted domain, any polynomial function of a given public input. For example, say $\alpha_1 = E(a_1)$, $\alpha_2 = E(a_2)$, and $\alpha_3 = E(a_3)$ were public, and we were asked to compute $a_1x^2 + a_2x + a_3$ in the encrypted domain. We could do this as

$$\alpha_1^{(x^2)}\alpha_2^x\alpha_3 = E(a_1x^2 + a_2x + a_3).$$

We would know the encryption of the polynomial, but have no knowledge of the actual value.

3.1 Paillier Cryptosystem

The Paillier cryptosystem is homomorphic, with multiplication of ciphertexts corresponding to the addition of the plaintexts. Furthermore, exponentiation of

a ciphertext corresponds to multiplication of the plaintext. We present a very brief summary of the system. See Paillier [13] for more details.

Let $N = pq$, where p and q are primes. Choose $g \in \mathbb{Z}^*_{N^2}$ such that the order of g is divisible by N. Any such g is of the form $g \equiv (1+N)^a b^N \bmod N^2$ for a pair (a, b), where $a \in \mathbb{Z}_N$ and $b \in Z^*_N$. Note that $(1+N)^a \equiv 1 + aN \bmod N^2$, so $g \equiv (1+aN)b^N \bmod N^2$. Let $\lambda = \text{lcm}(p-1, q-1)$. The public key is (g, N), the private key is λ. For message m and blinding factor $r \in \mathbb{Z}^*_N$, Paillier encryption is defined as

$$E_P(m, r; g, N) = g^m r^N \bmod N^2.$$

Note the equalities:

$$E_P(m_1, r_1; g, N) \cdot E_P(m_1, r_1; g, N) = E_P(m_1 + m_2, r_1 r_2; g, N)\,,$$

$$E_P(m, r; g, N)^k = E_P(mk, r^k; g, N).$$

In the Paillier cryptosystem, decryption is more complicated than encryption. First note that for any $x \in \mathbb{Z}^*_{N^2}$,

$$\begin{aligned} x^\lambda &\equiv 1 \,(\text{mod } N), \\ x^{N\lambda} &\equiv 1 \,(\text{mod } N^2). \end{aligned}$$

Given $c = E_P(m, r; g, N) = g^m r^N \bmod N^2$, we can see that

$$\begin{aligned} c^\lambda &\equiv g^{m\lambda} r^{N\lambda} \\ &\equiv (1+N)^{am\lambda} b^{\lambda N m} \\ &\equiv 1 + am\lambda N \ (\text{mod } N^2). \end{aligned}$$

Note also that $g^\lambda \equiv [(1+N)^a b^N]^\lambda \equiv 1 + a\lambda N \ (\text{mod } N^2)$. Therefore,

$$\frac{(c^\lambda \mod N^2) - 1}{N} = a\lambda m \quad \text{and} \quad \frac{(g^\lambda \mod N^2) - 1}{N} = a\lambda$$

To simplify, let $f_N(x) = \frac{(x \mod N^2) - 1}{N}$. Then we decrypt by computing

$$m = D_P(c; g, \lambda, N) = \frac{f_N(c^\lambda)}{f_N(g^\lambda)} \bmod N.$$

Optimizations are discussed by Catalano et al. [3], Damgård and Jurik [15], and Kalker [8].

3.2 Goldwasser-Micali Cryptosystem

The Goldwasser-Micali cryptosystem was developed in 1984 by Goldwasser and Micali [16]. It encrypts a single bit of information and is homomorphic in that multiplying ciphertexts corresponds to finding the XOR of the plaintexts. This cryptosystem is based on *quadratic residues*. A number is a quadratic residue modulo an odd prime p if it is the square of some number modulo p.

Definition 1 (Legendre symbol). *The Legendre symbol is defined as*

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x \equiv 0 \pmod{p} \\ 1 & \text{if } x \text{ is a quadratic residue modulo } p \\ -1 & \text{if } x \text{ is a quadratic non-residue modulo } p \end{cases}$$

By Euler's criterion[17], we compute $\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \pmod{p}$.

In the case of a composite modulus, the *Jacobi symbol* is used instead of the Legendre symbol.

Definition 2 (Jacobi symbol). *For $N = pq$, where p and q are odd primes, the Jacobi symbol is*

$$\left(\frac{x}{N}\right) = \begin{cases} 0 & \text{if } \gcd(x, N) > 1 \\ 1 & \text{if } \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) \\ -1 & \text{if } \left(\frac{x}{p}\right) = -\left(\frac{x}{q}\right) \end{cases}$$

Definition 3 (QR). *Let $QR(N)$ be the set of all quadratic residues modulo N.*

Lemma 1. *x is a quadratic residue modulo N iff $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$. If x is a quadratic residue modulo N^2 then it is a quadratic residue modulo N.*

Proof. If $x \in \mathrm{QR}(N)$ then $x = y^2 + kN = y^2 + kpq$ for some y, k, so $x \equiv y^2 \pmod{p}$ and $x \bmod p \in \mathrm{QR}(p)$. The same holds for q, so $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$. Given x such that $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$, we know that there exist a and b such that $a^2 \equiv x \pmod{p}$ and $b^2 \equiv x \pmod{q}$. By the Chinese Remainder Theorem [9], there exists a y such that $y \equiv a \pmod{p}$ and $y \equiv b \pmod{q}$. Since, $y^2 \equiv x \pmod{p}$ and $y^2 \equiv x \pmod{q}$, we know that $y^2 \equiv x \pmod{N}$, and therefore $x \in \mathrm{QR}(N)$. If $x \in \mathrm{QR}(N^2)$ then $x = y^2 + kN^2$ for some y, k, so $x \bmod N \in \mathrm{QR}(N)$. □

Definition 4 ($\tilde{\mathrm{QR}}$). *x is a* pseudosquare *modulo N if $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. Define $\tilde{QR}(N)$ to be the set of pseudosquares modulo N.*

It is easy to calculate Jacobi symbols, even if the factors of N are unknown (see Koblitz [9]). However, if the factorization of N is unknown, it is not always easy to determine quadratic residuosity. For any $x \in \mathrm{QR}(N) \cup \tilde{\mathrm{QR}}(N)$, $\left(\frac{x}{N}\right) = 1$, but determining if $x \in \mathrm{QR}(N)$ is a classical hard problem in cryptography and is assumed to be impossible without factoring N. If p and q are known, it is easy to determine if such an x is a quadratic residue by computing $\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \pmod{p}$ as above.

The Goldwasser-Micali Cryptosystem Let $N = pq$, where p and q are safe primes. Choose $g \in \tilde{\mathrm{QR}}(N)$. N and g are public while the factorization of N is private. Encryption takes as input a single bit b and a random blinding factor $r \in \mathbb{Z}_N^*$. The Goldwasser-Micali cryptosystem is defined as

$$E_{\mathrm{GM}}(b, r; g, N) = g^b r^2 \bmod N.$$

Decryption is defined as

$$D_{\mathrm{GM}}(x; p, q) = \begin{cases} 0 \text{ if } x \in \mathrm{QR}(N) \\ 1 \text{ if } x \in \tilde{\mathrm{QR}}(N) \end{cases}$$

If the factorization of N is known, decryption can easily be done by computing $\left(\frac{x}{p}\right) = x^{(p-1)/2} \bmod p$. Otherwise decryption is not possible, since it requires distinguishing members of $\mathrm{QR}(N)$ from members of $\tilde{\mathrm{QR}}(N)$ (see Section 3.2).

This system is homomorphic in that multiplying ciphertexts is equivalent to XORing plaintexts. Note the following equalities:

$$E_{\mathrm{GM}}(b_1, r_1; g, N) \cdot E_{\mathrm{GM}}(b_2, r_2; g, N) \equiv g^{b_1+b_2}(r_1 r_2)^2 \pmod N.$$

$$\equiv E_{\mathrm{GM}}(b_1 \oplus b_2, r_1 r_2; g, N) \pmod N.$$

The last equality holds because only the least bit of $b_1 + b_2$ matters, and $\oplus$ is equivalent to modulo 2 addition.

4 Phase I: Hidden Transform

The first phase of Secure QIM is a hidden linear transform. This means that the watermark detector takes the sample values from the signal and performs a transform on the sample values without learning the transform or the resulting transform coefficients.

Using the Paillier cryptosystem, we know how to perform addition and multiplication in the plaintext domain by performing the corresponding operations of multiplication and exponentiation in the ciphertext domain. Let a signal consist of m samples, $\boldsymbol{y} = (y_1, ..., y_m)^T$. The random transform takes $\boldsymbol{y}$ as input and produces n transform coefficients, $\boldsymbol{t} = (t_1, ..., t_n)^T$. Let the watermark embedder choose an orthogonal transform $\boldsymbol{S} = \{s_{ij}\}$, for $i = 1...n$ and $j = 1...m$, and let $\boldsymbol{s}_i$ be row i of the transform. Note that $\boldsymbol{t} = \boldsymbol{S}\boldsymbol{y}$ and $t_i = \boldsymbol{s}_i \cdot \boldsymbol{y}$.

The watermark detector is not allowed to know any of the values of $\boldsymbol{S}$, nor any of the values of $\boldsymbol{t}$. This is achieved by performing all the calculations in the Paillier encrypted domain. First, the watermark embedder chooses $N = pq$, where p and q are primes, and chooses a random $g \in \mathbb{Z}_{N^2}^*$ such that the order of g is divisible by N. Next, for $i \in [1, n], j \in [1, m]$, it generates random $\beta_{ij} \in \mathbb{Z}_N^*$. The public key consists of encryptions of the transform matrix $\boldsymbol{V} = \{v_{ij}\}$ where $v_{ij} = E_P(s_{ij}, \beta_{ij}; g, N)$.

The watermark detector wants to find $\boldsymbol{c} = (c_1, ..., c_m)$, the hidden transform coefficients. It does so by computing

$$c_i = \prod_j (v_{ij})^{y_j} \bmod N^2.$$

For later convenience in notation, define $w_i = \prod_{j=1}^{m} \beta_{ij}^{y_j}$. Then by the homomorphic properties of the Paillier cryptosystem, we have

$$c_i = \prod_j v_{ij}^{y_j} \bmod N^2 = \prod_j E_P(s_{ij}, \beta_{ij}; g, N)^{y_j} \bmod N^2$$

$$= \prod_j E_P(s_{ij} y_j, \beta_{ij}^{y_j}; g, N) \bmod N^2 = E_P(\sum_j s_{ij} y_j, \prod_j \beta_{ij}^{y_j}; g, N)$$

$$= E_P(\boldsymbol{s}_i \cdot \boldsymbol{y}, w_i; g, N) = E_P(t_i, w_i; g, N).$$

5 Phase II: Hidden Quantization

This section will present a simplified version of the hidden quantization scheme, uncoupled from the hidden transform, for a clearer presentation. The full version will be presented in Section 6.

The watermark embedder chooses $N = pq$, where p and q are safe primes, and $g \in \tilde{\mathrm{QR}}(N)$. It also chooses private quantization values $\boldsymbol{q} = (q_1, ..., q_n)$ where each $q_i \in \{0, 1\}$, and blinding values $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_n)$ where each $\gamma_i \in \mathbb{Z}_N^*$, and calculates $\boldsymbol{k} = (k_1, ..., k_n)$, $k_i = E_{\mathrm{GM}}(q_i, \gamma_i; g, N)$. It publishes g, N, and $\boldsymbol{k}$, and reveals the value of p to the secure module.

The watermark detector knows the public values g, N, and $\boldsymbol{k}$. Assume in this section that it has n unencrypted transform coefficients, $\boldsymbol{t} = (t_1, ..., t_n)$. If the signal is watermarked, these coefficients will each quantized so that $t_i \equiv q_i \,(\mathrm{mod} 2)$, but for any given coefficient, the watermark detector does not know the correct quantization value. The key point to notice is that if the signal is watermarked, $t_i \oplus q_i \equiv 0 \,(\mathrm{mod} 2)$.

First, the watermark detector chooses $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$, with $\alpha_i \in \mathbb{Z}_N^*$, and encrypts the transform coefficients,

$$c_i = E_{\mathrm{GM}}(t_i \bmod 2, \alpha_i; g, N).$$

Then it computes

$$\begin{aligned} f_i &= c_i k_i \\ &= E_{\mathrm{GM}}(t_i \bmod 2, \alpha_i; g, N) E_{\mathrm{GM}}(q_i, \gamma_i; g, N) \\ &= E((t_i \bmod 2) \oplus q_i, r_i; g, N). \end{aligned}$$

Note that f_i is a Goldwasser-Micali encryption of 0 if t_i is watermarked, 1 otherwise.

The secure module has a threshold function, $T(n)$. It is given as input $f_1, ..., f_n$, decrypts each f_i, sums the values, and announces that the data is watermarked if

$$\sum_{i=1}^{n} D_{\mathrm{GM}}(f_i; p, q) \leq T(n).$$

6 Secure QIM

This section presents the full system, in which the watermark detector performs a hidden transform on the input data and then quantizes the transform coefficients while still in the encrypted domain. It is a combination of the systems from Sections 4 and 5 with a careful choice of g and the blinding factors so that the ciphertexts of the hidden transform can be used for hidden quantization. In Sections 6.1, 6.2, and 6.3, we discuss the basic SQIM scheme, and in Section 6.4 we present modifications that prevent abuse by a malicious adversary.

6.1 Initialization

The watermark embedder chooses $N = pq$, where p and q are safe primes. Recall that for such N, $\lambda = \mathrm{LCM}(p-1, q-1)$. $g \in \mathbb{Z}^*_{N^2}$ is chosen so that $g \bmod N \in \tilde{\mathrm{QR}}(N)$ and the order of g, denoted $\mathrm{ORD}(g)$, is kN, where $k|\lambda$. All such g can be generated as follows. Choose $a \in Z^*_N$, so $\mathrm{GCD}(a, N) = 1$, and $b \in \tilde{\mathrm{QR}}(N)$. Let $g = (1+N)^a b^N \bmod N^2$. Because $(1+N)^a \equiv 1 + aN \bmod N$, we can write

$$g = (1 + aN)b^N \bmod N^2.$$

Claim. $g \bmod N \in \tilde{\mathrm{QR}}(N)$ and $\mathrm{ORD}(g) = kN$.

Proof. Notice that $g \equiv b^N \pmod N$. Since N is odd, $b^N \bmod N \in \tilde{\mathrm{QR}}(N)$, so $g \bmod N \in \tilde{\mathrm{QR}}(N)$. Because $\mathrm{ORD}(1+N) = N$ and $\mathrm{GCD}(a, N) = 1$, $\mathrm{ORD}(1+N)^a = N$. Let $k = \mathrm{ORD}(b^N)$. Since $b^N \in Z^*_N$, $k|\phi(N)$, and since N and $\phi(N)$ share no factors, $\mathrm{GCD}(k, N) = 1$. Therefore, $\mathrm{ORD}(g) = \mathrm{ORD}(1+N)\mathrm{ORD}(b^N) = kN$. □

6.2 Watermark Embedding

The watermark embedder chooses an orthogonal transform $\boldsymbol{S} = \{s_{ij}\}$, for $i = 1...n$ and $j = 1...m$. Let $\boldsymbol{s}_i$ be row i of the transform. The embedder also chooses $\boldsymbol{q} = (q_1, ..., q_n)$, with each $q_i \in \{0, 1\}$. It takes as input the signal $\boldsymbol{x} = (x_1, ..., x_m)$ and produces a watermarked signal $\boldsymbol{y} = (y_1, ..., y_n)$ such that for all i, $\boldsymbol{s}_i \cdot \boldsymbol{y} \equiv q_i \pmod 2$.

For $i \in [1, n], j \in [1, m]$, β_{ij} is chosen such that $\beta_{ij} \in \mathrm{QR}(N)$. For $i \in [1, n]$, γ_i is chosen so that $\gamma_i \in \mathrm{QR}(N)$. Let $\boldsymbol{V} = \{v_{ij}\}$ where $v_{ij} = E_P(s_{ij}, \beta_{ij}; g, N)$, and $\boldsymbol{k} = (k_1, ..., k_n)$ where $k_i = E_{\mathrm{GM}}(q_i, \gamma_i; g, N)$. The public watermarking key consists of N, $\boldsymbol{V}$, and $\boldsymbol{k}$.

6.3 Watermark Detection

First, the watermark detector finds the encrypted transform coefficients $\boldsymbol{c} = (c_1, ..., c_m)$. Recall that $t_i = \boldsymbol{s}_i \cdot \boldsymbol{y}$. Let $w_i = \prod_{j=1}^{m} \beta_{ij}^{y_j}$. Then

$$c_i = \prod_j v_{ij}^{y_j} \bmod N^2 = E_P(t_i, w_i; g, N) = g^{t_i} w_i^N \bmod N^2.$$

At this point we change from looking at ciphertexts modulo N^2 and begin looking at them modulo N. This is done so that the ciphertexts will be compatible with the Goldwasser-Micali cryptosystem.

We will now show that if $t_i \bmod 2 = 0$, then $c_i \bmod N \in \mathrm{QR}(N)$, otherwise $c_i \bmod N \in \tilde{\mathrm{QR}}(N)$. Since each $\beta_{ij} \in \mathrm{QR}(N)$, then $w_i \in \mathrm{QR}(N)$ and therefore $(w_i)^N \bmod N \in \mathrm{QR}(N)$. If $t_i \bmod 2 = 0$, then $g^{t_i} \bmod N \in \mathrm{QR}(N)$. Otherwise, since $g \bmod N \in \tilde{\mathrm{QR}}(N)$, $g^{t_i} \bmod N \in \tilde{\mathrm{QR}}(N)$. Therefore, if $t_i \bmod 2 = 0$ then $c_i \in \mathrm{QR}(N)$ otherwise, $c_i \in \tilde{\mathrm{QR}}(N)$. In both cases $\left(\frac{c_i}{N}\right) = 1$. Therefore,

$$c_i \bmod N = E_{\mathrm{GM}}(t_i \bmod 2, w_i^N g^{2\lfloor \frac{t_i}{2} \rfloor}; g, N).$$

Now we begin hidden quantization. Let $f_i = c_i k_i \bmod N$ and $z_i = w_i^N g^{2\lfloor \frac{t_i}{2} \rfloor}$.

$$f_i = c_i k_i \bmod N = E_{\mathrm{GM}}(t_i \bmod 2, z_i; g, N) E_{\mathrm{GM}}(q_i, \gamma_i; g, N) \bmod N$$

$$= E_{\mathrm{GM}}((t_i \bmod 2) \oplus q_i, \gamma_i z_i; g, N).$$

So, $D_{\mathrm{GM}}(f_i; p, q) = (t_i \bmod 2) \oplus q_i$, which is 0 if t_i was correctly quantized, 1 otherwise.

The secure module is given as input $f_1, ..., f_n$ and knows a threshold function $T(n)$. It decrypts each f_i, sums the values, and announces that the data is watermarked if

$$\sum_{i=1}^{n} D_{\mathrm{GM}}(f_i; p, q) \leq T(n).$$

6.4 Verification

It is possible for a malicious watermark detector to abuse the secure module. For example, if a watermark detector has a Goldwasser-Micali ciphertext $y = E_{\mathrm{GM}}(x, r; g, N)$ but does not know x, it can set the input to the secure module to be n copies of y. If the secure module says "watermarked", then the detector knows $x = 0$, otherwise it knows that $x = 1$. The watermark detector can trick the secure module into functioning as a Goldwasser-Micali decryption oracle.

To prevent this sort of abuse, we force the watermark detector to prove that each query to the secure module is valid in that is comes from an honest execution of the SQIM algorithm. If the a query is not proved to be valid, the secure module refuses to respond.

There are two steps to prove validity. First the detector proves *legitimacy*, the fact that the inputs to the secure module are the result of homomorphic

operations on ciphertexts from existing public watermarking keys. This prevents the detector from inventing new ciphertexts to use as input to the secure module, as in the example above. Next the detector proves *wholeness*, which prevents mixing and matching attacks. This prevents the watermark detector from mixing parts of multiple transform matrices, and forces it to calculate all transform coefficients with the same signal.

We now describe the modifications to the Secure QIM scheme that are necessary to prove legitimacy and wholeness. Section 8.1 describes how the modifications prove these properties.

Legitimacy Each public watermarking key must now include two encryptions of the transform matrix: $\boldsymbol{V} = \{v_{ij}\}$, $v_{ij} = E_P(s_{ij}, \beta_{ij}; g, N)$; and $\boldsymbol{W} = \{w_{ij}\}$, $w_{ij} = E_P(s_{ij}, \beta_{ij}; h, N)$; where h is formed as g is in Section 6.1, but $h \neq g$. The watermark detector uses $\boldsymbol{V}$ to compute $\boldsymbol{c} = \{c_i\}$ as in Section 6.3, and likewise uses $\boldsymbol{W}$ to compute $\boldsymbol{c'} = \{c'_i\}$. Note that the decryptions of c_i and c'_i are equal, but the ciphertexts that were used to generate c_i were encrypted with g, while those that were used to generate c'_i were encrypted with h.

Queries to the secure module now consist of n 4-tuples: (c_i, c'_i, k_i, f_i). The secure module checks that for all i,

$$D_P(c_i; g, \lambda, N) = D_P(c'_i; h, \lambda, N), \tag{1}$$

then checks that

$$f_i = c_i k_i \bmod N. \tag{2}$$

If either of these conditions do not hold for any i, then f_i is not legitimate and the secure module does not respond to the query.

Wholeness The wholeness constructions make use of a public hash function, $H(\cdot)$. Any hash function will do as long as it is collision-resistant. This means that it is difficult to find any pairs (x, y) where $x \neq y$ but $H(x) = H(y)$.

During initialization, the watermark embedder chooses a random $\boldsymbol{\theta} = \{\theta_1, ..., \theta_n\}$ where $\theta_i \in Z_N^*$. It includes in the watermarking public key $\sigma = E_P(H(\boldsymbol{\theta}), \rho_0; g, N)$ as well as $\boldsymbol{\Theta} = \{\Theta_1, ..., \Theta_n\}$, where $\Theta_i = E_P(\theta_i, \rho_i; g, N)$ and the ρ_i are random blinding factors. It also includes $\Delta = E_P(H(\boldsymbol{\theta}, \boldsymbol{k}), \tau; g, N)$, with blinding factor τ. Note that the hash in Δ includes $\boldsymbol{k}$, not $\boldsymbol{q}$.

In Section 6.2, the hidden transform was chosen as $\boldsymbol{S} = \{s_{ij}\}$, for $i = 1...n$ and $j = 1...m$. We now add an extra row, $\boldsymbol{s}_0$, to the top of the matrix $\boldsymbol{S}$. $\boldsymbol{s}_0$ is formed as a random linear combination of the all the other rows, $\boldsymbol{s}_i$:

$$\boldsymbol{s}_0 = \sum_{i=1}^{n} \theta_i \boldsymbol{s}_i.$$

Letting $k_0 = 0$ and $f_0 = 0$, we can say that the 4-tuple corresponding to $\boldsymbol{s}_0$ is (c_0, c'_0, k_0, f_0).

When the secure module receives a query, it is first given σ, $\boldsymbol{\Theta}$, and Δ. It decrypts σ and $\boldsymbol{\Theta}$ and checks if $D_P(\sigma; g, \lambda, N) = H(\theta_1, ..., \theta_n)$. If not, $\boldsymbol{\theta}$ is corrupt and the secure module refuses to respond. Next it checks if that $D_P(\Delta; g, \lambda, N) = H(\theta_1, ..., \theta_n, k_1, ..., k_n)$. If not, $\boldsymbol{k}$ is corrupt and the secure module refuses to respond. Finally, it checks if

$$D_P(c_0; g, \lambda, N) = \sum_{i=1}^{n} \theta_i D_P(c_i; g, \lambda, N). \tag{3}$$

If equation 3 does not hold, the transform was corrupt and the secure module does not respond.

7 Efficiency

In this section we will compare the efficiency of standard QIM with that of Secure QIM. Note that in both cases, the secure module starts off knowing the Paillier and Goldwasser-Micali private keys, but does not know any transform matrices or quantization values. In standard QIM, performed with a random transform on a secure module, the secure module is given a signal, an encrypted transform matrix, and encrypted quantization values, and performs QIM watermark detection by itself.

In our analysis we are concerned with the communication and computation required of the secure module. Let signals be of length m and let there be n transform coefficients. Samples of the signal are k-bit numbers and encryptions modulo N have $\ell = \log_2 N$ bits.

Communication In standard QIM, the secure module can receive all the information in an efficiently encrypted form. As such, we will calculate the number of bits that would be required unencrypted, and will assume that encryption adds negligible overhead. The secure module receives the signal (mk bits), the encrypted transform matrix (mnk bits), and the encrypted quantization values (n bits), for a total of $mnk + mk + n$ bits. The mnk term will dominate.

In SQIM, each number is encrypted individually modulo N or N^2. The secure module receives $(n + 1)$ 4-tuples of 6ℓ bits ($6(n + 1)\ell$ bits), σ (2ℓ bits), $\boldsymbol{\Theta}$ ($2n\ell$ bits), and Δ (2ℓ bits), for a total of $8n\ell + 10\ell$. The $8n\ell$ term dominates.

In comparison, standard QIM requires $\frac{m}{8}\frac{k}{\ell}$ times more bits than SQIM. Since m is the number of samples in the signal, this is a very large difference. For example, consider a signal of $m = 10^5$ samples, with $k = 24$, $n = 25$, and $\ell = 1024$. Then SQIM requires 26.3 kilobytes while standard QIM requires 7.4 megabytes. At a rate of 9600 bits per second, it would take 22.4 seconds to complete the SQIM data transfer, whereas it would take 1.8 hours for the standard QIM data transfer.

Computation The computational costs are harder to compute and more dependent on specific implementation. We estimate the cost of standard QIM as

the cost of performing the transform. Assuming straightforward matrix multiplication, this will have a running time of $O(nmk^2)$. We estimate the cost of SQIM based on the total number of decryptions. There are approximately $5n$ decryptions, which results in a running time of $O(n\ell^3)$. ℓ^3 very large, but ℓ is fixed based on security needs. k is generally in a small range, say 8 to 24 bits. So, the relative performance is highly dependent on the number of samples in the signal. With a small number of samples, standard QIM will be faster, while with a very large number of samples SQIM will be faster.

8 Security

Given the security of the Goldwasser-Micali and Paillier cryptosystems, our system is secure and zero knowledge. Length limitations prevent detailed proofs, so we offer proof sketches instead. More detailed proofs appear in Malkin [11].

8.1 Proof of Verification

The proof of zero knowledge rests upon the fact that invalid queries to the secure module will be rejected. Now we sketch proofs of this fact, taking our notation from Sections 6.2 and 6.3.

Legitimacy Recall that equation 1 requires that

$$D_P(c_i; g, \lambda, N) = D_P(c_i'; h, \lambda, N).$$

It is impossible for a watermark detector to produce c and c' such that this equation holds, other than by generating them homomorphically from hidden transform values.

Assume that algorithm $\mathcal{A}$ can produce c and c' that satisfy equation 1. By definition of Paillier decryption, that is equivalent to saying

$$c = g^m r_1^N \bmod N^2 \tag{4}$$

$$c' = h^m r_2^N \bmod N^2 \tag{5}$$

for some m and some blinding factors r_1 and r_2. Note however, that $\mathcal{A}$ doesn't know g or h. This means that equations 4 and 5 must simultaneously be true for all possible pairs (g, h) (perhaps with a different m per pair). This is not possible, so $\mathcal{A}$ can not exist. Note that this proof can also be extended to work when the algorithm $\mathcal{A}$ is probabilistic.

The only access to g and h that the detector has is through the encrypted values in the public watermarking key, so all c and c' are the result of homomorphic operations on such values.

Equation 2 ($f_i = c_i k_i \bmod N$) guarantees that the f_i were generated by performing hidden quantization on c_i with the supplied k_i. This means that hidden quantization was legitimately performed on a legitimate hidden transform.

Wholeness The watermark detector can not mix and match Θ_i from different public watermarking keys, because σ would not match $\boldsymbol{\Theta}$ and the collision resistance of H prevents the detector from constructing a match. Likewise, all the k_i must come from the same $\boldsymbol{k}$ or Δ won't match. Furthermore, $\boldsymbol{\Theta}$ and $\boldsymbol{k}$ must come from the same public watermarking key, or Δ won't match.

In any watermark detection, all the v_{ij} must come from the same $\boldsymbol{V}$, and that $\boldsymbol{V}$ is the one associated with $\boldsymbol{\Theta}$. If a watermark detector mixes and matches coefficients from different $\boldsymbol{V}$ it can not fulfill equation 3:

$$D_P(c_0; g, \lambda, N) = \sum_{i=1}^{n} \theta_i D_P(c_i; g, \lambda, N).$$

Encrypting both sides of equation 3, we see

$$c_0 = \prod_{i=1}^{n} {c_i}^{\theta_i} = \prod_{i=1}^{n} {\Theta_i}^{D_P(c_i; g, \lambda, N)}.$$

A mix-and-match attack would have to generate c_0, but neither of these equalities can be formed homomorphically. An algorithm $\mathcal{A}$ that can generate c_0 without knowing these decryptions can be used to compute the computational Diffie-Hellman problem: given g^a and g^b, compute g^{ab}. This is assumed to be impossible, so $\mathcal{A}$ can't exist.

We showed that for a given $\boldsymbol{\Theta}$ there is only one possible $\boldsymbol{k}$ that will be accepted, and also for a given $\boldsymbol{\Theta}$ there is only one possible $\boldsymbol{V}$ that will be accepted, and that no mixing and matching is possible

8.2 Proof of Zero Knowledge

By definition, SQIM is zero knowledge if the watermark detector can simulate what it sees during watermark detections. This is easy if the detector is honest.

The detector has a watermarking public key, is given (or chooses) a signal s, and receives a bit b from the secure module, so its view is $v = (s, b)$. s is chosen from a distribution D that includes whether or not s is watermarked. To simulate this view, first choose a transform matrix and quantization values, then choose a signal s' according to D. Since the transform matrix and quantization values are known, it is easy to see if s' is watermarked. Let $b' = 1$ if s' is watermarked, $b' = 0$ otherwise, and let the view of the simulation be $v' = (s', b')$. v and v' are identically distributed, so SQIM is honest-detector zero knowledge.

The verification system ensures that the view of a dishonest watermark detector is the same as that of an honest watermark detector because dishonest queries are ignored. Therefore, SQIM is zero knowledge from the perspective of the watermark detector.

Note that there are no zero knowledge proofs here. The secure module is trusted, so it does not have to prove that its calculations are correct.

8.3 Proof of Security

In both the Paillier and Goldwasser-Micali cryptosystems, any properly-formed N with large-enough factors is secure. The security of the Paillier cryptosystem is independent of the choice of g, and any g such that $g \bmod N \in \tilde{\mathrm{QR}}(N)$ is secure for the Goldwasser-Micali cryptosystem. Therefore, even though we choose g in a unique manner, it is still secure.

The security of Paillier encryption depends on the blinding factors, which we also choose in a unique manner. Since the blinding factors in standard Paillier encryption are chosen at random from Z_N^*, choosing them from a subset of non-negligible size does not introduce security problems; if it did, then choosing them at random from Z_N^* would have a non-negligible chance of encountering the same problems. A problem would exist if the subset were small enough to make a brute-force search possible, but $|\mathrm{QR}(N)| = \frac{1}{4}|\mathbb{Z}_N^*|$, so this is not a concern. Therefore, our choice of blinding factors is secure.

The only public watermarking values are either Paillier ciphertexts or Goldwasser-Micali ciphertexts which are acted upon homomorphically. Since N, g, and the blinding factors are chosen securely, these ciphertexts are as secure as the Paillier and Goldwasser-Micali cryptosystems, respectively.

The only area where the security of SQIM does not derive immediately from the Paillier and Goldwasser-Micali cryptosystems is when Paillier ciphertexts are taken modulo N and converted into Goldwasser-Micali ciphertexts. However, this is still provably secure. Consider these ciphertexts to belong to a new, hybrid cryptosystem. The essence of the proof is that any algorithm that has an advantage in decrypting these hybrid ciphertexts can be shown to have an advantage decrypting Goldwasser-Micali ciphertexts. Since we assume the security of the Goldwasser-Micali cryptosystem, no such algorithm can exist, and the hybrid cryptosystem is secure.

9 Conclusion

We have presented a Secure QIM system, one in which most of the work of watermark detection can be performed in the open without any information about the private key being leaked. A secure module, such as a smartcard, is used to perform the final portion of watermark detection, revealing only if the signal is watermarked and no other information about the watermark. Our hidden transform utilizes the Paillier cryptosystem, while our hidden quantization uses the Goldwasser-Micali cryptosystem, and a robust verification system is used to force all watermark detectors to honestly follow the SQIM algorithm.

The novelty of SQIM lies in the coupling of two cryptosystems to implement a semi-private key QIM, and in the verification system which prevents watermark detectors from cheating. Furthermore, our system is provably secure, assuming the security of the Paillier and Goldwasser-Micali cryptosystems.

References

1. A. Adelsbach and A . Sadeghi. Zero-knowledge watermark detection and proof of ownership. In *Information Hiding Workshop*, 2000.
2. D. Brumley and D. Boneh. Remote timing attacks are practical. In *12th USENIX Security Symposium*, 2003.
3. Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. Paillier's cryptosystem revisited. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 206–214, New York, NY, USA, 2001. ACM Press.
4. B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. on Information Theory*, 47(4):1423–1443, May 2001.
5. Ingemar J. Cox and John-Paul M. G. Linnartz. Public watermarks and resistance to tampering. In *International Conference on Image Processing (ICIP'97)*, pages 26–29, 1997.
6. J. Eggers, J. Su, and B. Girod. Asymmetric watermarking schemes, 2000.
7. Gael Hachez and Jean-Jaques Quisquater. Which directions for asymmetric watermarking? In *XI European Signal Processing Conference*, 2002.
8. Ton Kalker. Secure watermark detection. In *Allerton Conference*, 2005.
9. Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer Verlag, 1994.
10. P. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Crypto*, 1996.
11. M. Malkin. *Cryptographic Methods in Multimedia Identification and Authentication*. PhD thesis, Stanford University, http://theory.stanford.edu/∼mikeym/papers/malkin-thesis.pdf, 2006.
12. J. Jaffe P. Kocher and B. Jun. Differential power analysis: Leaking secrets. In *Crypto*. Springer Verlag, 1999.
13. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Eurocrypt '99*, volume 1592, pages 223–238. Springer-Verlag, 1999.
14. E.C. Chang Q. Li. Security of public watermarking schemes for binary sequences. In *Information Hiding Workshop*, 2002.
15. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Proc. of Public Key Cryptography*, 2001.
16. S.Micali s. Goldwasser. Probabilistic encryption. In *Journal of Computer and Systems Science*, volume 28, pages 270–299. 1984.
17. D. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
18. Ilaria Venturini. Counteracting oracle attacks. In *Proceedings of the 2004 Workshop on Multimedia and Security*, pages 187–192. ACM Press, 2004.