# Efficient Mesh Licensing

Stephen Toub
toub@fas.harvard.edu

Alexander Healy
ahealy@fas.harvard.edu

May 24, 2001

## Abstract

We present an efficient mesh watermarking scheme whereby a vendor can embed purchaser-specific information (i.e. a user license) into a mesh upon the sale of the mesh. This watermark is robust to translation, rotation, uniform scaling, cropping and random perturbation of vertices (up to a threshold). Furthermore, since this method imposes only minor changes to the geometry of the model, it can be used in conjunction with ownership watermarks such as the technique presented in [2].

## Introduction

The practice of *watermarking* geometric models involves a trade-off between several important properties: robustness (whether the model retains the watermark even after having been modified), information (how many bits can be encoded in the model), non-malleability (how secure the watermark is against forgery) and efficiency (how much preprocessing is required to embed the watermark and how much processing is required to recover it). In this work, we focus on the last two properties, non-malleability and efficiency, while still providing a modicum of robustness. We present an efficient mesh watermarking scheme whereby a vendor can embed purchaser-specific information (i.e. a user license) into a mesh upon the sale of the mesh. This watermark is robust to translation, rotation, uniform scaling, cropping and random perturbation of vertices (up to a threshold). The security of this watermark against forgery is grounded in the (conjectured) difficulty of computing discrete logarithms in a finite group, and, more generally, we illustrate a technique for representing a cryptographic/algebraic structure within the framework of polygonal meshes.

## Previous Work and Motivation

Steganography (information-hiding) is of great interest to many content providers including those who provide digital audio and digital images. The watermarking techniques from these areas have been extended and modified so that polygonal meshes can be watermarked. We focus on two noteworthy examples:

In [5], Wagner presents a robust watermarking scheme that uses the lengths of edges in the mesh to hide a message. The watermark is preserved even after affine-transformations and cropping. However, the process of embedding the watermark is quite costly and can necessitate solving very large linear systems. This is impractical if we want to embed a different watermark each time a mesh is sold (e.g. if the watermark is to include the name of the purchaser). Hence, we would like a watermarking scheme which is allows the purchaser-specific information to be embedded very efficiently.

In [2], the authors present a watermarking scheme in which the message is hidden in the geometry (as opposed to the connectivity) of the mesh. This scheme is robust to arbitrary affine transformations and re-meshing. In order to ensure that the watermark non-invertible (i.e. to prevent false ownership claims), the authors suggest using a cryptographic hash of the ownership information to seed a random number generator that produces the watermark that will be embedded. This works if a given owner wants to show that he owns the mesh; however, if we are embedding purchaser-specific information into the mesh, this can be problematic. Consider the following simple scenario:

Alice sells a mesh to Bob with a watermark that says “This is Alice’s mesh and Bob is licensed to use this copy.” Now Bob gives the mesh to Carol. Alice recognizes Carol's mesh, but she cannot determine that it was Bob that gave the mesh to Carol (illegally), unless she guesses that the watermark was "This is Alice's mesh and Bob is licensed to use this copy," and not "This is Alice's mesh and (anyone's name) is licensed to use this copy."

Hence, we would like a scheme in which we can guarantee that the watermark cannot be forged, but where we can recover the watermark text rather than just verifying it, i.e. where we do not have to know the watermarked information before it is recovered.

## Encoding Information

In this section we are concerned with the problem of encoding a string of $n$ bits into a mesh. In the next section we will address the issues of encrypting that string before it is embedded into the mesh. Let the original mesh be denoted by $\mathcal{O}$. From this mesh we will construct two meshes, $\mathcal{A}$ and $\mathcal{B}$, which have the same connectivity as $\mathcal{O}$, but where the vertices have been perturbed. In particular, for each vertex $v \in \mathcal{O}$, we compute the tangent plane at $v$, and we project the star of $v$ onto the tangent plane. Let $\epsilon$ denote the length of the shortest edge incident to $v$ in this projection. Next we randomly choose an angle $\theta \in [0, 2\pi)$, and a length $\ell \in (0, \frac{\epsilon}{\lambda}]$, where $\lambda$ is a constant parameter ($\lambda = 8$ works well in practice). In mesh $\mathcal{A}$, the vertex corresponding to $v$ is moved a distance $\ell$ in the direction $\theta$ on the tangent plane; in mesh $\mathcal{B}$, the vertex corresponding to $v$ is moved a distance $\ell$ in the opposite direction.

Recall that $n$ denotes the number of bits we wish to encode in the mesh. Now, we partition the vertices $V$ of $\mathcal{O}$ into $n$ sets, $S_i$, of size $\left\lfloor \frac{|V|}{n} \right\rfloor$ or $\left\lceil \frac{|V|}{n} \right\rceil$. To create a watermarked mesh, $\mathcal{W}$, with an $n$-bit text $T = b_1 b_2 \ldots b_n$ embedded into it, we do the following: For each $S_i$, we choose the positions of the vertices in $\mathcal{W}$ from $\mathcal{A}$ if $b_i = 0$ and from $\mathcal{B}$ if $b = 1$. We can now interpret the effect of such an encoding in a more conventional setting: While we cannot assume that the $S_i$'s will remain secret, we know that each vertex in $\mathcal{A}$ and $\mathcal{B}$ was produced by the exact same random process (recall that one vertex was perturbed by $\ell$ where as the other was perturbed by $-\ell$); therefore, to a third party who does not know the text $T$ which is encoded in the mesh, there is no way for them to determine whether a given vertex (and hence a set of vertices) came from $\mathcal{A}$ or $\mathcal{B}$. In this way, this encoding is information-theoretically secure, provided that $\mathcal{A}$ and $\mathcal{B}$ are kept secret. In particular, it is analogous to encoding $T$ as $T \oplus S$ for some randomly-generated secret key $S$.

To recover a message from such a watermarked mesh, $\mathcal{W}$, we need to realign the mesh and compare the locations of the vertices in each set $S_i$ of $\mathcal{W}$ to the locations of the vertices in $\mathcal{A}$ and $\mathcal{B}$. If a majority of the vertices in a given set $S_i$ are closer to the corresponding vertices in mesh $\mathcal{A}$, then bit $i$ is taken to be a 0. Otherwise, bit $i$ is taken to be a 1. The topic of realigning (or registering) the watermarked mesh is discussed in [2].

Finally, a note about efficiency: If we assume that the meshes $\mathcal{A}$ and $\mathcal{B}$ have been created as a preprocess, then the process of creating a mesh with an $n$-bit text $T$ encoded into it simply involves choosing the positions for the vertices from either $\mathcal{A}$ or $\mathcal{B}$. Hence, this encoding process is $O(|V|)$, where $V$ denotes the set of vertices of the mesh.

## Encryption

In the above scheme, if the encoded message is known and the $S_i$'s are known (this may be possible by comparing many encoded meshes) then such a watermark can be forged. In order to thwart such an attack, we will not encode the message $M$, but rather $E(M)$, where $E$ is a private-key encryption function.

Since the messages we encode into the mesh are simply elements of $\{0, 1\}^n$, it is natural to consider a message $M$ as an element of $\mathbb{F}_{2^n}$, realized as $\mathbb{F}_2[x]/f(x)$ where $f(x)$ is an irreducible polynomial of degree $n$ (since each polynomial in this quotient is easily represented a $n$-tuple of 0's and 1's, namely the coefficients of the polynomial). Furthermore, we choose $n$ so that $2^n - 1$ is prime, i.e. a Mersenne prime. This ensures that the multiplicative group $\mathbb{F}_{2^n}^{\times}$, which has order $2^n - 1$, is cyclic of prime order, and hence every message is a multiplicative generator, except for the messages $0 \ldots 00$ and $0 \ldots 01$, which we assume will never be used. Although Mersenne primes are relatively rare, there are several reasonable values of $n$ for this application, namely $n = 521, 607$ or $1279$ (see sequence $A000043$ in [4]).

Now, we can use the following private-key encryption function, with secret key $s \in \{1, 2, \ldots, 2^n - 1\}$. Given a message $M \in \mathbb{F}_{2^n}^{\times}$, the encryption, of M is defined by $E(M) = M^s$.

Such a private-key encryption function is desirable because its security can be shown to be equivalent to solving the Diffie-Hellman problem and, in this case, also the discrete logarithm problem.

**Problem 1 (Diffie-Hellman Problem).** *Given a cyclic group, $G$, a generator $g \in G$, $g^a$ and $g^b$ for some unknown integers $a$ and $b$, compute $g^{ab}$.*

**Problem 2 (Discrete Logarithm Problem).** *Given a cyclic group, $G$, a generator $g \in G$, and $g^a$, for some unknown integer $a$, compute $a$.*

It is conjectured that there is no efficient (i.e. polynomial-time) algorithm to solve these problems over various groups $G$, including $G = \mathbb{F}_{2^n}^{\times}$. Clearly, if one can solve the Discrete Logarithm Problem efficiently, then one can also solve the Diffie-Hellman problem efficiently, but the converse is not known to be true in general. Even so, in [1], Maurer shows that these two problems are computationally equivalent if $p-1$ or $p+1$ is sufficiently smooth for each prime factor $p$ of $|G|$. Since we have chosen $G$ to have prime order equal to $2^n - 1$, the only prime factor of $|G|$ is $2^n - 1$ and $(2^n - 1) + 1 = 2^n$ is 2-smooth; hence, the hypotheses of Maurer's result are met and the two problems are computationally equivalent.

This is of interest to us, since the following result relates the difficulty of forging an encryption $E(M)$ (given $M$, and without knowing the secret key $s$) to the difficulty of solving the Diffie-Hellman Problem.

**Proposition 1.** *Suppose an adversary has a polynomially-bounded number of pairs $(M_i, E(M_i))$, where the known plaintexts $M_i$, are randomly distributed over all possible messages in $\mathbb{F}_{2^n}$. Then, if the adversary can efficiently compute $E(M')$ for some known message $M'$, then she can efficiently solve the Diffie-Hellman Problem in $\mathbb{F}_{2^n}^{\times}$.*

*Proof.* First we show that only one pair $(M_0, E(M_0))$ is necessary. This is simply because given such a pair, we can construct random pairs $(M_i, E(M_i))$, by computing $(M_0^r, E(M_0)^r)$ for a random $r \in [1, 2, \ldots, 2^n - 1]$. Clearly, $E(M_0)^r = (M_0^s)^r = (M_0^r)^s = E(M_0^r)$, and since any message $M_0$ is a generator, $M_0^r$ is different for every $r$; hence a random choice of $r$ yields a random pair $(M, E(M))$ where $M = M_0^r$. Thus, if the adversary can efficiently compute the ciphertext $E(M')$, given a polynomial number of random pairs $(M_i, E(M_i))$, then she can can efficiently compute $E(M') = M'^s$ given a single pair $(M_0, E(M_0))$. However, $M' = M_0^t$ for some $t$, since $M_0$ is a generator. Therefore, the adversary is able to efficiently compute $M'^s = (M_0^t)^s = (M_0)^{st}$ given a generator $M_0$, $M_0^s = E(M_0)$ and $M_0^t = M'$. This is exactly the statement of the Diffie-Hellman Problem, and the result follows. □

**Corollary 1.** *Suppose an adversary has a polynomially-bounded number of pairs $(M_i, E(M_i))$, where the known plaintexts $M_i$, are randomly distributed over all possible messages in $\mathbb{F}_{2^n}$. Then, if the adversary can efficiently compute $E(M')$ for some known message $M'$, then she can efficiently solve the Discrete Logarithm Problem in $\mathbb{F}_{2^n}^{\times}$.*

*Proof.* By the previous proposition, we know that the adversary can efficiently solve the Diffie-Hellman Problem in $\mathbb{F}_{2^n}^{\times}$. By Maurer's result in [1], we know that the existence of an efficient algorithm to solve the Diffie-Hellman Problem in $\mathbb{F}_{2^n}^{\times}$ implies that there is an efficient algorithm for solving the Discrete Logarithm Problem in $\mathbb{F}_{2^n}^{\times}$. □

Unfortunately, this result depends on the known plaintexts, $M_i$, being random. It is possible, however, that an adversary would choose the $M_i$'s deterministically in order to improve his chance of forging messages. To thwart such an attack, we recommend padding the messages with random bits before encoding them, i.e. given $M = m_1 m_2 \ldots m_k$, with $m_i \in \{0, 1\}$ let $M' = m_1 m_2 \ldots m_k r_{k+1} r_{k+2} \ldots r_n$ where the $r_i$ are random bits, and compute $E(M')$. This will serve to randomize the plaintexts before they are encoded. The size of $k$, relative to $n$ will determine the security of this encoding (and conversely how much information is encoded).

## Application

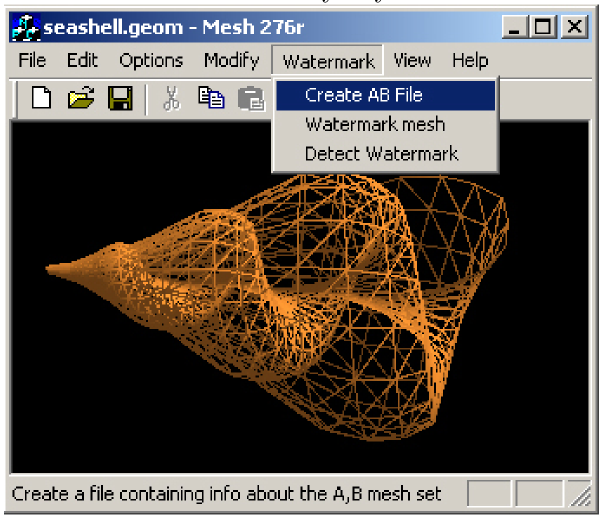
As we have noted before, this scheme only makes minor changes to the geometry of the model. For this reason, it could be used in conjunction with the watermarking scheme proposed in [2], as in the following example:

Meshmart licenses meshes that it owns to its customers. All of Meshmart's meshes are watermarked using Praun et al.'s scheme from [2]. Furthermore, each time a mesh is licensed to a customer, that customer's information is embedded using the scheme presented above. This way, Meshmart's meshes are always provided with both an ownership watermark and a license watermark, and so if Alice is found to have a mesh containing Meshmart's ownership watermark, but without a valid license watermark, she is culpable for mesh-fraud. Additionally, our method allows for the possibility that Alice's mesh will also retain its license watermark, revealing that Bob in fact purchased the mesh from Meshmart, and gave it (illegally) to Alice.

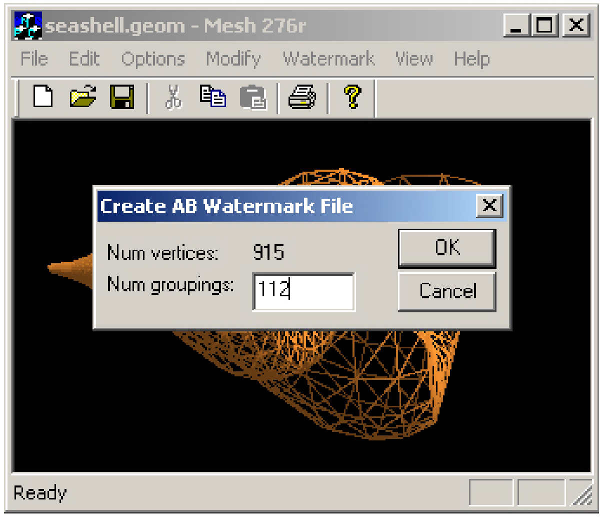
## Implementation

The accompanying implementation demonstrates the functionality of our proposed licensing scheme with triangular meshes. The application was written in C++, using OpenGL and our own mesh representation/manipulation library. Meshes are read and stored as .GEOM files. For simplicity, this application does not actually perform the encryption steps described above. Even so, exponentiation in $\mathbb{F}_{2^n}$ can be performed very quickly and so the difference in performance is negligible. The following screen-shots demonstrate the functionality of the system:

As a preprocess, a .MAB (Mesh A, B) file is created which stores the displacement information of the vertices and any key information:



Create a file containing info about the A,B mesh set



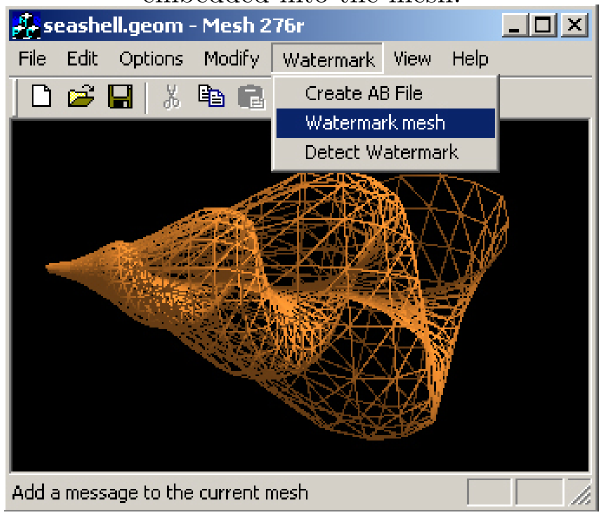
Create AB Watermark File

Num vertices: 915

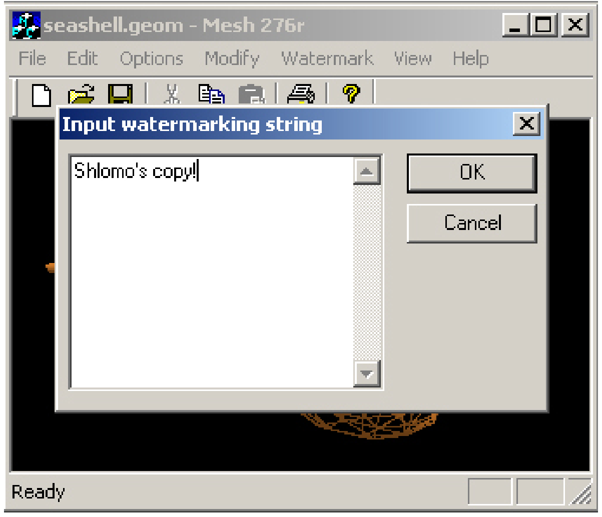
Num groupings: 112

Ready

Using this .MAB file, a license/watermark can be embedded into the mesh:



Add a message to the current mesh

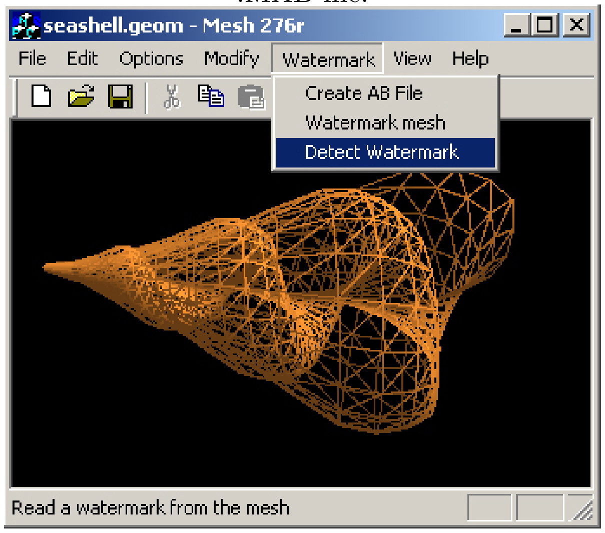


Input watermarking string

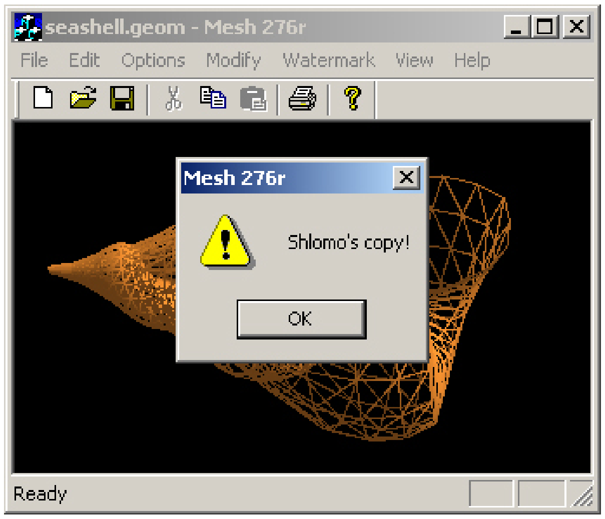
Shlomo's copy!

Ready

Then the watermark can be recovered, using the .MAB file:



Read a watermark from the mesh



Shlomo's copy!

Ready

## Conclusions and Future Work

We have presented an efficient watermarking scheme that incorporates algebraic cryptography in order to guarantee that the watermark cannot be forged. There are undoubtedly many cryptographic schemes that could be used in secure watermarking/licensing; however, the challenge lies in finding algebraic structures that are well-suited for embedding in meshes. For our purposes, $\mathbb{F}_{2^n}$ worked well, but there may be other encoding schemes which lend themselves to encoding elements of $\mathbb{Z}_N$ where $N$ is the product of two primes as in the RSA and Rabin cryptosystems, or other useful algebraic structures.

Also, as it is presented here, our watermarks are robust to translation, rotation, uniform scaling, cropping, and random perturbations of vertices. It would be of interest to extend this encoding technique so that the watermarks are also robust to arbitrary affine transformations, or even projective transformations. However, this would almost certainly come at the expense of performance.

We have focused on the issues of efficiency and non-malleability in polygonal mesh watermarking. We have exhibited an efficient licensing scheme that is secure against forgery and which allows the licensing information to be recovered, and not just verified. This is particularly important when embedding purchaser-specific information into meshes if we want to find out which party broke the conditions of the license. Furthermore, this scheme can be used in conjunction with the robust watermarking technique in [2] in order to generate meshes with robust ownership watermarks and efficient user-license watermarks.

## References

[1] Ueli M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In Yvo G. Desmedt, editor, *Proc. CRYPTO 95*, pages 271–281. Springer, 1994.

[2] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust mesh watermarking. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 49–56, Los Angeles, 1999. Addison Wesley Longman.

[3] Masaki Aono Ryutarou Ohbuchi, Hiroshi Masuda. Watermarking three-dimensional polygonal models. In *ACM Multimedia 97*, pages 261–272, 1997.

[4] N. J. A. Sloane. The online encyclopedia of integer sequences. In *http://www.research.att.com/ njas/sequences/*, 2001.

[5] M. Wagner. Robust watermarking of polygonal meshes. pages 201–208.