Algorithms to Watermark Software Through Register Allocation

William Zhu and Clark Thomborson*

Department of Computer Sciences,
University of Auckland, Auckland, New Zealand
fzhu009@ec.auckland.ac.nz, cthombor@cs.auckland.ac.nz

Abstract. Software security is a significant issue in the Internet age. In order to prevent software from piracy and unauthorized modification, many techniques have been developed. Software watermarking is such a technique that can be used to protect software by embedding some secret information into the software to identify its copyright owner. In this paper, we discuss algorithms of software watermarking through register allocation.

The QP Algorithm [1, 2] was proposed by Qu and Potkonjak to watermark a solution to a graph coloring(GC) problem to protect its intellectual property. In a recent paper by Myles and Collberg [3], the QP algorithm was corrected, and was, for the first time, implemented to watermark software through register allocation. It is called the QPS algorithm.

Our paper discusses some difficulties with the published descriptions of the QP and QPS algorithms, points out the problem in the extractability of the watermarks inserted by the QP algorithm through examples, proves the correctness of a clarified version of the QPS algorithm, and proposes an improvement for the QP algorithm. Finally, we give some potential topics for further research.

Keyword: Software Watermarking, Graph, Interference Graph, Graph Coloring.

1 Introduction

With the rapid development of the software industries, computer security [4, 5] and the protection of intellectual property of software from piracy becomes more and more important issues in computer business and academicia. Software watermarking is an approach to embed a message into software to claim the ownership of it [6, 7, 8, 9]. It is one of effective mechanisms to protect the intellectual property of the developers for a software.

Qu, Potkonjak, et al. developed some techniques to watermark the solutions to constraint problems such as the GC problem [1, 2, 10, 11, 12]. The GC problem is to color the vertices of a graph with the fewest number of colors such that

* Research supported in part by the New Economy Research Fund of New Zealand.

R. Safavi-Naini and M. Yung (Eds.): DRMTICS 2005, LNCS 3919, pp. 180–191, 2006.

no vertices connected by an edge receive the same color. In compiler, the GC problem is used to allocate the registers for variables of a program. Potkonjak and Qu proposed the QP algorithm [1, 2] in 1998, but they have not published any detailed implementation for QP algorithm; they even have not considered any attacks in their analysis, however resistance to attack is of vital importance in digital and software watermarking.

In 2004, Myles and Collberg [3], for the first time, implemented the QP algorithm to watermark software through register allocation, and conducted an excellent and thorough empirical evaluation of this algorithm. They tried various attacks on the QP algorithm to analyze its robustness. Furthermore, they propose the QPS algorithm to compensate for the flaws they discovered in the QP algorithm. However, for the reasons shown in this paper, there are still some confusing points in the QPS algorithm.

Le and Desmedt also pointed out some other flaws in the QP algorithm [13]. They claimed that watermarked solution resulted from the QP algorithm could be modified in such a way that any message could be verified, and thus the watermark inserted could not be used to show ownership of the solution.

This paper is organized as follows. Section 2 discusses some basic concepts in software watermarking systems. Section 3 introduces the QP algorithm and points out some flaws in the QP algorithm as a method of watermarking. Section 4 discusses the QPS software watermarking algorithm, a variant of the QP algorithm for software watermarking. It is pointed out that there are not clear points in the `if` statement in the QPS algorithm. A potential clarified version of the QPS algorithms is explored and we proved the soundness of this version of the QPS algorithm. Section 5 details our improved QP algorithm, which we call the QPI algorithm. Section 6 summarizes our conclusions about the QP algorithm, the QPS algorithm, and our QPI algorithm. Section 7 points out several topics for further research.

2 Software Watermarking Systems

A software watermarking system can be divided into two subsystems: embedding subsystem and extracting subsystem. Embedding subsystem tries to insert watermarks in programs, while extracting subsystem aims to take watermarks from watermarked programs. There are several software watermarking algorothms currently avaiable, among them is the graph-based algorithm, in which a watermark is encoded as a graph with some special properties. Venkatesan, Vazirani and Sinha [14] proposed the first graph-based Software watermarking algorithm called the VVS algorithm. It is a static software watermarking algorithm. Collberg and Thomborson [7] proposed the first dynamic graph algorithm, the CT algorithm which inserts a watermark encoded as a data structure graph and only running a watermarked program with a special input, called a key, does the watermark in the watermarked program appear.

A public cryptographic key could be used as a watermark value W [7, 8]. Only the owner of the public key should know the corresponding private key. If the key is sufficiently long, and if the watermark graph-extracting algorithm and

the decoding algorithm are sufficiently well-publicised, then an attacker would be unable to mount a convincing counterclaim of ownership. The attacker might produce a fraudulent decoder $d'()$ and/or a fraudulent graph-extractor $g'()$, such that $d'(g'(P)) = W'$, where $g'(P)$ is a graph found in a watermarked program P by extractor $g'()$ and W' is a public key whose corresponding private key is known to the attacker. However one or both of $d'()$ or $g'()$ would bear little resemblance to the well-publicised $d()$ and $g()$. Furthermore it would seem extremely difficult (and may someday be proved to be computationally infeasible) for the attacker to produce a watermark embedding process $e' : X \times W \to X$ such that their watermarking system will operate ideally, or near-ideally, over a wide range of programs X and watermarks W. In an ideal watermarking system, the extractor always finds an embedded watermark

$$\forall x \in X, \forall w \in W : d'(g'(e'(x, w)) = w$$

and it never finds a spurious watermark

$$\forall y \in X, \forall w \in W, \exists x \in X : (d'(g'(y) = w) \Longrightarrow (y = e'(x, w))$$

Note that it is trivial for an attacker to produce a spurious watermarking system (d', g', e') that will operate ideally over a very small range of programs and watermarks, for the extractor could do a table lookup on its inputs and then report a watermark that is arbitrarily chosen by the attacker.

3 The QP Watermarking Algorithm

Qu and Potkonjak proposed a watermarking algorithm for watermarking solutions to Graph Colouring (GC) problems [1, 2], which is called the QP algorithm in [3, 15]. It requires the vertices of the graph to be indexed, that is, each vertex must be labeled with a unique integer in the range 1 to $|V(G)|$. The QP algorithm relies heavily on the ordering of node indices. The followings are some concepts used in the QP algorithm.

Definition 1. *Cyclic mod n ordering [1, 2]: We use "$<_i$" to denote the cyclic mod n ordering relation for a fixed i, such that $i <_i (i+1) <_i \ldots <_i n <_i 1 <_i \ldots <_i i-1$. Where there is no confusion over the value of i, we omit the subscript in $<_i$.*

Definition 2. *Two nearest vertices that are not connected to a vertex v_i [1, 2]: For a vertex v_i of a graph G with $|V| = n$, we say $v_{i_1} \in V$ and $v_{i_2} \in V$ are the two nearest vertices that are not connected to a vertex v_i if $i <_i i_1 <_i i_2$; $(v_i, v_{i_1}) \notin E$; $(v_i, v_{i_2}) \notin E$; $\forall j : i <_i j <_i i_1$, $(v_i, v_j) \in E$; and $\forall j : i_1 <_i j <_i i_2$, $(v_i, v_j) \in E$.*

In this paper, if the above two vertices exist for a vertice v_i, we also say vertice v_i has two candidate vertices v_{i_1} and v_{i_2}.

The essence of the QP algorithm is to add an extra edge between every vertex v_i and one of its two candidate vertices. The choice between these two nearest

unconnected vertices is determined by the watermark bits to be embedded. It is important to notice that this concept is a dynamic one, since the two candidate vertices of v_i may change whenever an edge is added to the neighborhood of v_i.

After a watermark is inserted in a cover message using an embedding algorithm, an important question we may ask is if this watermark can be extracted by some algorithm.

3.1 The QP Embedding Algorithm

The original QP algorithm in Fig. 1 was proposed by Qu and Potkonjak [1, 2]. It inserts a watermark into a solution to a GC problem.

```
Input: an unwatermarked graph G and
       a message bits: W = w_1 w_2 ... w_m
Output: a watermarked graph G'.
Algorithm:
n = |V|
G' = G
for each i from 1 to n
     if v_i has two candidate vertices v_{i_1} and v_{i_2}
         if w_i = 0 connect v_i to v_{i_1} in G'(V, E')
         else connect v_i to v_{i_2} in G'(V, E')
return G'
```

Fig. 1. The original QP algorithm [1, 2]

We note a subtle problem in the algorithm of Fig. 1. We cannot expect to insert one bit for every vertex in an arbitrary graph G. In Fig. 2, we show an "obvious" adaptation of the QP embedding algorithm, to handle arbitrary G.

```
Input: an unwatermarked graph G(V, E) and
       a message bits: W = w_1 w_2 ... w_m
Output: a watermarked graph G'.
Algorithm:
n = |V|
G' = G
j = 0
for each i from 1 to n
     if v_i has two candidate vertices v_{i_1} and v_{i_2}
         j++
         if w_j = 0 connect v_i to v_{i_1} in G'(V, E')
         else connect v_i to v_{i_2} in G'(V, E')
return G'
```

Fig. 2. A clarified version of the QP algorithm [1, 2]

3.2 The QP Extraction Algorithm

The QP extraction algorithm in [2] is as follows. Given the graph G', for each vertex v_i we consider all vertices v_j such that v_i and v_j have different colors and $(v_i, v_j) \notin E(G)$. One bit of information can be decoded for each such pair of vertices, by counting the number $n(i, j)$ of nodes k with indices $i <_i k <_i j$ which are not connected to v_i. The value of the message bit is defined by the following case analysis on $n(i, j)$:

1. If $n(i,j) = 0$, the watermark bit is 0;
2. If $n(i,j) = 1$, the watermark bit is 1;
3. If $n(i,j) > 1$, then the watermark bit is 0 if $n(j,i)$ is 0; the watermark bit is 1 if $n(j,i)$ is 1; and the watermark bit is undefined otherwise.

The unwatermarked graph G plus its coloring is not enough to recognize the watermark embedded in the watermarked graph. Even the unwatermarked graph plus its coloring and plus the coloring of the watermarked graph is still not enough to recognize the watermark embedded in the watermarked graph. This can be seen from the following example.

Example 1. Let $G(V,E)$ have 3 vertices v_1, v_2, v_3 and no edges. We can color its all 3 vertices with color RED. After inserting a message $W = 0$, $G(V,E)$ becomes a new graph $G'_1(V,E'_1)$ with 3 vertices v_1, v_2, v_3, v_4 and 1 edge $\{v_1, v_2\}$. We can color it so that v_1 with color RED, v_2 and v_3 with color BLUE.

For the above graph $G(V,E)$, we can also color its all 3 vertices with color RED. After inserting a message $W = 1$, it becomes another new graph $G'_2(V,E'_2)$ with 3 vertices v_1, v_2, v_3 and 1 edge $\{v_1, v_3\}$. We can also color it so that v_1 with color RED, v_2 and v_3 with color BLUE.

The same original graph $G(V,E)$ has the same coloring for the watermarked graphs but different messages inserted.

Myles and Collberg has also pointed out that the above QP extraction algorithm is incorrect, but their example [3] for the extraction failure of the QP algorithm is itself not clear.

3.3 The QP Algorithm Is Not Extractable

The QP algorithm is not extractable, since, as shown in the following example, inserting two different messages into an original graph respectively, we get the same watermarked graph.

Example 2 (Extraction failure of the QP algorithm). Let $G(V,E)$ have 4 vertices v_1, v_2, v_3, v_4 and two edges $(v_1, v_3), (v_2, v_4)$.

The first message to embed is $W_1 = 010$.

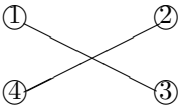
$E' = E$

For $i = 1$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 2$, $i_2 = 4$. For $w_j = 0$, we connect v_1 and v_2. Now $E' = E' \cup (v_1, v_2) =$ $\{(v_1, v_3), (v_2, v_4), (v_1, v_2)\}$.

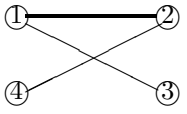
For $i = 2$, v_i has no the two nearest vertices that are not connected to v_i, so we cannot embed a bit for this vertex.

For $i = 3$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 4, i_2 = 2$. For $w_j = 1$, we connect v_3 and v_2. Now $E' = E' \cup (v_2, v_3) =$ $\{(v_1, v_3), (v_2, v_4), (v_1, v_2), (v_2, v_3)\}$.

For $i = 4$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 1, i_2 = 3$. For $w_j = 0$, we connect v_1 and v_4. Now $E' = E' \cup (v_1, v_4) =$ $\{(v_1, v_3), (v_2, v_4), (v_1, v_2), (v_2, v_3), (v_1, v_4)\}$. The following figure shows this embedding process.

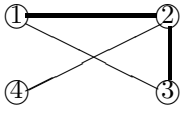


The original graph.

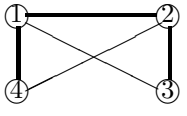


$i = 1$, $i_1 = 2$ and $i_2 = 4$, $w_j = 0$, so connect v_1 and v_2.

For $i = 2$, we cannot add any edge.



$i = 3$, $i_1 = 4$ and $i_2 = 2$, $w_j = 1$,,
so connect v_2 and v_3.



$i = 4$, $i_1 = 1$ and $i_2 = 3$, $w_j = 0$, so connect v_1 and v_4.

This is the watermarked graph.

The second message to embed is $W_2 = 111$.

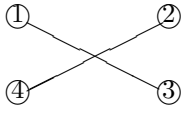
$E' = E$

For $i = 1$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 2$, $i_2 = 4$. For $w_j = 1$, we connect v_1 and v_4. Now $E' = E' \cup (v_1, v_4) = \{(v_1, v_3), (v_2, v_4), (v_1, v_4)\}$.

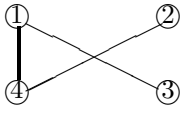
For $i = 2$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 3, i_2 = 1$. For $w_j = 1$, we connect v_1 and v_2. Now $E' = E' \cup (v_1, v_2) = \{(v_1, v_3), (v_2, v_4), (v_1, v_4), (v_1, v_2)\}$.

For $i = 3$, v_i has the two nearest vertices that are not connected to v_i with $i_1 = 4, i_2 = 2$. For $w_j = 1$, we connect v_3 and v_2. Now $E' = E' \cup (v_2, v_3) = \{(v_1, v_3), (v_2, v_4), (v_1, v_2), (v_3, v_2), (v_3, v_2)\}$.

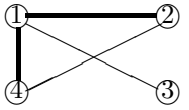
For $i = 4$, v_i has no the two nearest vertices that are not connected to v_i, so we cannot embed a bit for this vertex. Now we also have the same $E' = \{(v_1, v_3), (v_2, v_4), (v_1, v_2), (v_2, v_3), (v_1, v_4)\}$. The following figure shows this embedding process.



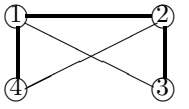
The original graph.



When $i = 1$, $i_1 = 2$ and $i_2 = 4$, $w_j = 1$,
so connect v_1 and v_4.



When $i = 2$, $i_1 = 3$ and $i_2 = 1$, $w_j = 1$,
so connect v_1 and v_2.



When $i = 3$, $i_1 = 4$ and $i_2 = 2$, $w_j = 1$, so connect v_2 and v_3.
For $i = 4$, we cannot add any edge.
This is the watermarked graph.

The problem is in that all bits of a message are embedded in an edge of E' which is not in E. For an edge $(v_k, v_l), k < l$, of E' but not in E, it may be connected in the following four possible cases:

1. when $i = k$ and $i_1 = l$
2. when $i = k$ and $i_2 = l$
3. when $i = l$ and $i_1 = k$
4. when $i = l$ and $i_2 = k$

In the first and third cases, the edge (v_k, v_l) means a bit 0 inserted, while in the second and fourth cases, it means a bit 1 inserted according to the QP embedding algorithm.

4 The QPS Software Watermarking Algorithm

After pointing out that a watermark inserted into a graph by the QP extraction algorithm cannot be extacted reliably, Myles and Collberg proposed the QPS software watermarking algorithm [3], a variant of the QP algorithm. In the QPS algorithm, two core concepts are used. They are "triple" and "colored triple" as follows.

Definition 3 (triple [3]). *For a graph $G = (V, E)$, if 3 vertices v, v', v'' of G satisfy the following two conditions:*

1. $v, v', v'' \in V$
2. $(v, v'),\ (v, v'),\ (v', v'') \notin E$

they are called a triple.

Definition 4 (colored triple [3]). *For a graph $G = (V, E)$, if a triple v, v', $v'' \in V$ are all colored the same color, then they are called a colored triple.*

Triples and colored triples change dynamically during the watermark embedding process, as did the cyclic mod-n ordering of our Definition 2. From the definition of GC, if three vertices are all colored the same, then condition 2 of Definition 3 is satisfied. If a triple is not a colored triple, then we call it a "multicolored triple".

4.1 The Original QPS Algorithm

Myles and Collberg applied the QP algorithm to software watermarking. The original QPS embedding algorithm [3] is in Fig. 3 and the QPS extraction algorithm [3] is in Fig. 4. In the QPS embedding algorithm, the input graph G of Fig. 3 would be the interference graph of a program P. The output graph G' would be the interference graph of a compiled program P', and the nodes of G and G' are the variables in P. Interference graph is a concept for register allocation in compilers [16]. If two variables interfere in P, then they cannot be assigned to the same register when P is compiled. This constraint on register allocation is modelled by introducing an edge between these two variables in P's interference graph G. A legal coloring of G is thus an acceptable register assignment for the compilation of P, if we consider each register to have a distinct color.

In the QPS embedding and extraction algorithm, the statement "v_{i_1} and v_{i_2} are not already in a triple G" is not clear.

4.2 A Clarified Version of the QPS Algorithm

From the example in [3], page 281, 4.2 Preliminary Example, a possible clarified version of the QPS embedding algorithm is given in Fig. 5. The corresponding extraction algorithm is as in Fig. 6.

This version of the QPS embedding algorithm works well in that a message embedded by itself can be recognized correctly by its corresponding extraction algorithm, however, the conditions in its "if" statement in it is so restricted that it can only embed much fewer bits of message into a graph than the QP algorithm can.

Proof of the correctness of the above QPS algorithm.

As said before, every bit of a message is embedded in an edge of E' which is not in E. For an edge (v_k, v_l), $k < l$, of E' but not in E, generally, there are two possible to connect it; when $i = k$ or when $i = l$. Now we prove that if we can get this edge when $i = k$, then we cannot get it when $i = l$ and vice verse. In fact, if

Input: an unwatermarked graph $G(V, E)$
a message $W = w_1 w_2 \ldots$ to be embedded into the $G(V, E)$
Output: a watermarked graph G' with message W embedded in it
Algorithm:
$n = |V|$
$G' = G$
j=0
for each i from 1 to n
if v_i is not in a triple G' AND possible find the nearest two vertices v_{i_1} and v_{i_2}
for v_i such that v_{i_1} and v_{i_2} are the same color as v_i in G'
AND v_{i_1} and v_{i_2} are not already in a triple in G'.
j++
if $w_j = 0$
connect v_i and v_{i_1} in G'
if $w_j = 1$
connect v_i and v_{i_2} in G'
return $G'(V, E')$ and the inserted message $W' = w_1 w_2 \ldots w_j$

Fig. 3. The QPS embedding elgorithm

```
Input: an unwatermarked graph G(V, E)
       a watermarked graph G'
Output: a message W embedded in G'
Algorithm:
n = |V|
j=0
for each i from 1 to n
     if v_i is not in a triple G' AND possible find the nearest two vertices v_{i_1} and v_{i_2}
          for v_i such that v_{i_1} and v_{i_2} are the same color as v_i in G
          AND v_{i_1} and v_{i_2} are not already in a triple in G.
          j++
          if v_i and v_{i_1} have the different colors in G'
              w_j = 0
              connect v_i and v_{i_1} in G
          else
              w_j = 1
              connect v_i and v_{i_2} in G
return W = w_1 w_2 ... w_j
```

Fig. 4. The QPS extraction algorithm

we can get this edge in the case of $i = k$, then $i_1 = l$ or $i_2 = l$. First we consider the case of $i_1 = l$. According to the clarified QPS embedding algorithm, there is a number h, such that $k < l < h$ and v_l, v_h are the two candidate vertices of v_k. Since we can connect the edge (v_k, v_l) only when $i = k$ or $i = l$, if we does not connect it when $i = k$, we would connect the edge (v_k, v_h) when $i = k$. When $i = l$, the vertices v_k and v_l are still not connected, so are the vertices v_l and v_h. Therefore, $i_1 \leq h$ or $i_2 \leq k$. If $i_2 < k$, it is impossible to connect the edge (v_k, v_l). The only possibility to connect the edge (v_k, v_l) is the two candidate vertices of v_l are v_h and v_k, i.e., $i_1 = h$ and $i_2 = k$. In this case, v_i, v_{i_1} and v_{i_2} are not a triple, for the edge (v_{i_1}, v_{i_2}) has been connected, so we cannot connect the edge (v_k, v_l).

In case of $i_2 = l$, in the same way, we can prove that when $i = l$, we cannot connect the edge (v_k, v_l).

We can also prove that if we can get this edge when $i = l$, then we cannot get it when $i = k$ in the same way as above. Therefore, if an edge of E' which is not in E can be used to embed one bit of message 0, it cannot be used to embed a bit 1.

```
Input: an original graph G(V, E)
       a message W = w_1 w_2 ... to be embedded into the G(V, E)
Output: a watermarked graph G' with message W embedded in it
Algorithm:
n = |V|
G' = G
WV = V
j=0
for each i from 1 to n
     if possible find the nearest two vertices v_{i_1} and v_{i_2} in G'
          such that v_i, v_{i_1}, v_{i_2} have the same color and are a triple in G' and v_{i_1}, v_{i_2} ∈ WV
          WV = WV − {v_{i_1}, v_{i_2}}
          j++
          if w_j = 0
              connect v_i and v_{i_1} in G'
          if w_j = 1
              connect v_i and v_{i_2} in G'
return G'(V, E') and the inserted message W' = w_1 w_2 ... w_j
```

Fig. 5. A clarified version of the QPS embedding algorithm

```
Input: an unwatermarked graph G(V, E)
       a watermarked graph G'(V, E')
Output: a message W embedded in G'(V, E')
Algorithm:
n = |V|
WV = V
j=0
for each i from 1 to n
    if possible find the nearest two vertices v_{i_1} and v_{i_2} from G
        for v_i such that v_i, v_{i_1}, v_{i_2} have the same color in G and are a triple in G'
        and v_{i_1}, v_{i_2} ∈ WV
        WV = WV − {v_{i_1}, v_{i_2}}
        j++
        if v_i and v_{i_1} have the different colors in G'
            w_j = 0
            connect v_i and v_{i_1} in G
        else
            w_j = 1
            connect v_i and v_{i_2} in G
return W = w_1 w_2 ... w_j
```

Fig. 6. A clarified version of the QPS extraction algorithm

5 The QPI Algorithm

We give an improved QP embedding algorithm, the QPI embedding algorithm, in Fig. 7. It is an informed software watermarking algorithm. We change the definition of the two candidate vertices $v_{i_1} \in V$ and $v_{i_2} \in V$ for a vertice $v_i \in V$. The original definition in [1, 2] used the cyclic mod n order for numbers $1, 2, \ldots, n$, while we use the order $1 < 2 < \ldots < n$ in our new definition.

```
Input: an original graph G(V, E)
       a message W = w_1 w_2 ... to be embedded into the G(V, E)
Output: a watermarked graph G' with message W embedded in it
Algorithm:
n = |V|
G' = G
j = 0
for each i from 1 to n
    if v_i has two candidate vertices v_{i_1} and v_{i_2}
        j++
        if w_j = 0
            connect v_i to v_{i_1} in G'
            change the color of v_{i_1} to different one from the current colors used in G'
        else
            connect v_i to v_{i_2} in G'
            change the color of v_{i_2} to different one from the current colors used in G'
return G'
```

Fig. 7. The QPI embedding algorithm

Definition 5. *Two candidate vertices: for a vertex v_i of a graph G with $|V| = n$ and a coloring of G, we say v_i has two candidate vertices $v_{i_1} \in V$ and $v_{i_2} \in V$ if $i < i_1 < i_2 \leq n$ and vertices v_i, v_{i_1}, and v_{i_2} have a same color and $(v_i, v_{i_2}) \notin E$; furthermore, $\forall j : i < j < i_1$ and $\forall j : i_1 < j < i_2 \leq n$, vertices v_i and v_j have different color.*

```
Input: an unwatermarked graph G(V, E) with n = |V|
       a watermarked graph G(V, E')
Output: the message W embedded in the watermarked graph G(V, E')
Algorithm:
j = 0
for each i from 1 to n
     if v_i has two candidate vertices v_{i_1} and v_{i_2}
         j++
         if v_i and v_{i_1} have different colors in G'
             w_j = 0
             connect v_i and v_{i_1} in G
             change the color of v_{i_1} to different one from the current colors used in G
         else
             w_j = 1
             connect v_i and v_{i_2} in G
             change the color of v_{i_2} to different one from the current colors used in G
return W = w_1 w_2 ... w_j
```

Fig. 8. The QPI extraction algorithm

The QPI embedding algorithm is an extractable algorithm. The proof of it is similar to that of QPS algorithm in Subsection 4.2 of this paper; we also give an extraction algorithm corresponding to the QPI embedding algorithm in Fig. 8.

For our QPI embedding algorithm, every edge (v_k, v_l), $k < l$ in G' while not in G is only connected when $i = k$, so there is only one possibility for an edge in G' while not in G to embed a bit of message. Furthermore, this improved QP embedding algorithm works for all cases.

6 Conclusions

Now we reach our following conclusions about the QP algorithm through the above discussions.

1. The message embedded into a graph by the QP embedding algorithm is not extractable in general.
2. The QP extraction algorithm is not correct. It tries to recognize a message just by the unwatermarked graph; it does not use the watermarked graph.
3. The QPS algorithm proposed by Myles and Collberg is the first one that implemented the QP algorithm for software watermarking, though it includs some not clear descriptions.
4. The QPS algorithm is the first one algorithm that watermark software through register allocation.
5. The QPI algorithm proposed by us can correctly realize Qu and Potkonjak's idea and can be used to software watermarking through register allocation.

7 Potential Research Directions

From the paper [1, 2], we think it is important to distinquish an extraction algorithm and a recognition algorithm in software watermarking. An extraction algorithm tries to extract all bits of the message inserted in a software, while a

recognition algorithm decides whether a watermark exists in a software. A good work to define these concepts is not as easy as it seems. We will explore this problem in our further works.

Another potential topic for future research is to design algorithms to embed a watermark into a graph such that it can still be recognized when the vertices of the graph have been reordered.

Acknowledgements. Thanks for Dr. F.-Y. Wang's stimulating suggestions on this paper. Thank Mr. Jun Ni, Mr. Han Zhang, and other colleages in the software security group at the University of Auckland for their helpful comments.

References

1. G. Qu, M. Potkonjak, Analysis of watermarking techniques for graph coloring problem, in: IEEE/ACM International Conference on Computer Aided Design, '98, 1998, pp. 190–193.
2. G. Qu, M. Potkonjak, Hiding signatures in graph coloring solutions, in: Information Hiding Workshop '99, 1999, pp. 348–367.
3. G. Myles, C. Collberg, Software watermarking through register allocation: Implementation, analysis, and attacks, in: LNCS 2971, 2004, pp. 274–293.
4. C. Collberg, G. Myles, A. Huntwork, Sandmark–a tool for software protection research, IEEE Security and Privacy 1 (4) (2003) 40–49.
5. W. Zhu, C. Thomborson, A provable scheme for homomorphic obfuscationin in software security, in: The IASTED International Conference on Communication, Network and Information Security, CNIS'05, Phoenix, USA, 2005, pp. 208–212.
6. C. Collberg, C. Thomborson, On the limits of software watermarking, in: Technical Report #164, Department of Computer Science, The University of Auckland, 1998.
7. C. Collberg, C. Thomborson, Software watermarking: Models and dynamic embeddings, in: Proceedings of Symposium on Principles of Programming Languages, POPL'99, 1999, pp. 311–324.
8. C. Collberg, C. Thomborson, Watermarking, tamper-proofing, and obfuscation - tools for software protection, IEEE Transactions on Software Engineering 28 (2002) 735–746.
9. W. Zhu, C. Thomborson, F.-Y. Wang, A survey of software watermarking, in: ISI 2005, Vol. 3495 of LNCS, 2005, pp. 454–458.
10. G. Qu, J. Wong, M. Potkonjak, Optimization-intensive watermarking techniques for decision problems, in: Design Automation Conference, '99, 1999, pp. 33–36.
11. G. Qu, M. Potkonjak, Fingerprinting intellectual property using constraint-addition, in: Design Automation Conference '00, 2000, pp. 587–592.
12. G. Qu, J. Wong, M. Potkonjak, Fair watermarking techniques, in: EEE/ACM Asia and South Pacific Design Automation Conference, '00, 2000, pp. 55–60.
13. T. Le, Y. Desmedt, Cryptanalysis of ucla watermarking schemes for intellectual property protections, in: LNCS 2578, Springer-Verlag, 2003, pp. 213–225.
14. R. Venkatesan, V. Vazirani, S. Sinha, A graph theoretic approach to software watermarking, in: 4th International Information Hiding Workshop, Pittsburgh, PA, 2001.
15. W. Zhu, C. Thomborson, On the QP algorithm in software watermarking, in: ISI 2005, Vol. 3495 of LNCS, 2005, pp. 646–647.
16. K. D. Cooper, T. J. Harvey, L. Torczon, How to build an interference graph, Software – Practice and Experience 28 (4) (1988) 425–444.