Informed Recognition in Software Watermarking

William Zhu*

Department of Computer Science,
University of Auckland, Auckland, New Zealand
fzhu009@ec.auckland.ac.nz

Abstract. Software watermarking is a technique to protect programs from piracy through embedding secret information into the programs. As software unauthorized use and modification are ubiquitous in the world, progresses in software watermarking will certainly benefit software research and industry. In this paper, we study one of core concepts in this area – informed recognition. To recognize a watermark in a software is to judge the existence of such a watermark in the corresponding software code.

1 Introduction

Since the unauthorized use and modification of software are pervasive around the world, software security becomes an important issue [5,6,7,19,20]. Software watermarking is a method to protect copyright of programs by inserting secret messages into the programs. With the rapid development of intelligence and security informatics [8,15], we find a new potential application area of software watermarking. Combined with other techniques, software watermarking can also be used in database protection [2] and information security problems [1].

The basic definitions of software watermarking concepts appeared in the early papers by Collberg et al. [5,6]. They also defined the extraction and recognition of software watermarks, but these definitions are not very formal and detailed. Nagra et al. [12,13] and Thomborson et al. [14] classified software watermarks from a functional view. Concepts and techniques of software watermarking also abound in [6,9,11,16].

Zhu and Thomborson formally defined embedding, extraction, and recognition in papers [19] and [20]. This paper follows the above two papers to define concepts such as positive-partial informed recognitions, negative-partial informed recognitions, and informed recognitions corresponding to embedding algorithms.

This paper is organized as follows. Section 2 gives the concepts of embedding, extraction and recognition. Section 3 is the focus of this paper. We define the concepts of informed recognition such as positive-partial informed recognition, negative-partial informed recognition, and informed recognition. Section 4 concludes our paper.

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

C.C. Yang et al. (Eds.): PAISI 2007, LNCS 4430, pp. 257–261, 2007.

2 Embedding, Extraction and Recognition

A software watermarking system must do two basic things: embed a watermark into a software object, then extract all bits of the watermark inserted by itself, or recognize whether or not there exists a watermark embedded by itself. In this section, we introduce some concepts of embedding a watermark into, extracting a watermark from, and recognizing a watermark from a software program. These issues were already addressed in our previous paper [19,20]. We need these concepts to define the concepts of informed recognition.

Informally speaking, to embed a software watermark into a program is to insert a secret message into this code. We formally define this concept as follows.

Definition 1. *(Watermark) A watermark is a message of bits of 0 and 1 with a finite length* ≥ 0*. We denote the set of all watermarks as* $\mathbf{W}$*.*

Definition 2. *(Embedding) Let* $\mathbf{P}$ *denote the set of programs and* $\mathbf{W}$ *the set of watermarks. We call a function* $A : \mathbf{P} \times \mathbf{W} \rightarrow \mathbf{P}$ *an embedder.*

If $P' = A(P, W)$ *for some* $P \in \mathbf{P}$ *and some* $W \in \mathbf{W}$*, then the* P' *is called the watermarked program and* $P \in \mathbf{P}$ *the original program.*

After a watermark is inserted in a cover message using an embedder, an important consideration is the potential for an algorithm to extract this watermark. The following definition specifies all potential watermarks an embedder can insert into a program. This set excludes messages which do not change a cover program.

Definition 3. *(Set of candidate watermarks) A* $W \in \mathbf{W}$ *is called a candidate watermark with respect to a program* P *and an embedder* A *if* $A(P, W) \neq P$*. All candidate watermarks constitute the set of candidate watermarks of the program* P *and the embedder* A*. This set is denoted as candidate*(P, A)*.*

As for the detailed concepts of extraction and recognition in software watermarking, please refer to papers [19,20].

3 Informed Recognition

For positive-partial recognizers, negative-partial recognizers, and recognizers defined in last section, we provide the original program and the suspected watermarked program as their inputs to judge whether the suspected watermarked program has a watermark. In this section, we define a new type of recognition, informed recognition, in which a recognizer is given an original program, a suspected watermarked program, and a suspect watermark as its inputs.

We divide informed recognition into three classes: positive-partial informed recognition, the negative-partial informed recognition, and informed recognition. For a positive-partial informed recognition, if a program really has a specific watermark, the recognition will detect it. But, such an informed recognition might say a program has a watermark while this program actually has no such a watermark.

Definition 4. *(Positive-partial informed recognition) Let* $A : \mathbf{P} \times \mathbf{W} \to \mathbf{P}$ *be an embedder,* $PIR : \mathbf{P} \times \mathbf{P} \times \mathbf{W} \to \{\text{TRUE}, \text{FALSE}\}$ *a function. If* PIR *satisfies that* $\forall P, P' \in \mathbf{P}$*, if there is a* $W \in candidate(A, P)$ *such that* $P' = A(P, W)$*, then* $PIR(P', P, W) = \text{TRUE}$*, we call* PIR *a* positive-partial *informed recognition function corresponding to the embedder* A*, or simply a* positive-partial *informed recognizer.*

The partial recognition concepts are very flexible. The following is an example of positive-partial informed recognition.

Example 1. (Trivial positive-partial informed recognizers) For an embedder $A : \mathbf{P} \times \mathbf{W} \to \mathbf{P}$, define a function $S : \mathbf{P} \times \mathbf{P} \times \mathbf{W} \to \{\text{TRUE}, \text{FALSE}\}$, as follows:

$$\forall P', P \in \mathbf{P}, W \in \mathbf{W},\ S(P', P) = \text{TRUE}.$$

This is a positive-partial informed recognition corresponding to A. We call such a function a trivial positive-partial informed recognizer corresponding to A and denote it as $TPIR(A)$.

For a negative-partial informed recognizer, if it says a program has a watermark, this program really has a watermark. But, such a recognizer might say a program do not have a specific watermark while this program actually has such a watermark.

Definition 5. *(Negative-partial informed recognition) Let* $A : \mathbf{P} \times \mathbf{W} \to \mathbf{P}$ *be an embedder,* $NIR : \mathbf{P} \times \mathbf{P} \times \mathbf{W} \to \{\text{TRUE}, \text{FALSE}\}$ *a function. If* NIR *satisfies that* $\forall P, P' \in \mathbf{P}$*,* $NIR(P', P) = \text{TRUE} \implies P' = A(P, W)$ *for some* $W \in \mathbf{W}$*, we call* NIR *a* negative-partial *informed recognition function corresponding to the embedder* A*, or simply a* negative-partial *informed recognizer.*

We present an example of negative-partial informed recognition.

Example 2. (Trivial negative-partial informed recognizers) For an embedder $A : \mathbf{P} \times \mathbf{W} \to \mathbf{P}$, define a function $S : \mathbf{P} \times \mathbf{P} \times \mathbf{W} \to \{\text{TRUE}, \text{FALSE}\}$ as follows:

$$\forall P', P \in \mathbf{P}, W \in \mathbf{W},\ S(P', P, W) = \text{FALSE}.$$

This is a negative-partial informed recognizer corresponding to A. We call such a function a trivial negative-partial informed recognizer and denote it as $TNIR(A)$.

For a complete informed recognizer, if a program has a watermark, the recognizer will say that this program has a watermark; if a program has no watermarks, the recognizer will say that this program has no watermarks.

Definition 6. *(Informed recognizer) For an embedder* $A : \mathbf{P} \times \mathbf{W} \to \mathbf{P}$*, if a function* $R : \mathbf{P} \times \mathbf{P} \to \{\text{TRUE}, \text{FALSE}\}$ *satisfies* $\forall P, P' \in \mathbf{P}, W \in \mathbf{P}$*,* $R(P', P, W) = \text{TRUE} \iff P' = A(P, W)$ *for some* $W \in candidate(A, P)$*, we call* R *a* complete informed recognition function *for the embedder* A*, or simply an informed recognizer. We say that* A *is informed recognizable if there exists a recognizer for* A*.*

Theorem 1. *For every embedder A, there exists one and only one informed recognizer corresponding to A. We denote the unique recognizer corresponding to A as $IR(A)$.*

Proof. $\forall P, P' \in \mathbf{P}, W \in \mathbf{P}$, define $IR(P', P, W)$ as follows:

$IR(P', P, W)$ = TRUE, if there is some $W \in candidate(A, P)$ such that $P' = A(P, W)$.

$IR(P', P, W)$ = FALSE, otherwise.

It is easy to see IR is a recognizer corresponding to A. □

From Theorem 1 and Example 8 in [19], not all embedders are extractable, but every embedder is informed recognizable.

Theorem 1 shows there is one and only abstract informed recognizer, but there might be several concrete recognition algorithms to realize such an informed recognizer.

Property 1. For every embedder A, $IR(A)$ is both the positive-partial and the negative-partial informed recognizers corresponding to A.

An extreme positive partial informed recognizer will always say a program has a watermark while an extreme negative partial informed recognizer will always say a program has no watermarks. These two informed recognizers are not useful in practice. Now we consider the relative strength of two informed recognizers.

Definition 7. *(Strength of partial informed recognizers) Let $PIR1$ and $PIR2$ be two positive-partial informed recognizers corresponding to an embedder A. If $\forall P, P' \in \mathbf{P}, W \in \mathbf{W}$, $PIR_2(P', P, W)$ = TRUE $\Longrightarrow$ $PIR_1(P', P, W)$ = TRUE, we say PIR_2 is at least as strong as PIR_1.*

Let NIR_1 and NIR_2 be two negative-partial informed recognizers corresponding to an embedder A. If $\forall P, P' \in \mathbf{P}, W \in \mathbf{W}$, $NIR_1(P', P, W)$= TRUE $\Longrightarrow$ $NIR_2(P', P, W)$ = TRUE, we say NIR_2 is at least as strong as NIR_1.

Property 2. For an embedder A, $TPIR(A)$ is the weakest positive-partial informed recognizer and $IR(A)$) is the strongest positive-partial informed recognizer corresponding to A.

$TNIR(A)$ is the weakest negative-partial informed recognizer and $IR(A)$ is the strongest negative-partial informed recognizer corresponding to A.

4 Conclusions

Recognition is a very complicated concept in software watermarking. In this paper we define the concepts involved in informed recognition. We have not considered the attack issue in this paper. How to recognize watermarks from attacked programs is challenging research topic in software watermarking. We will also study how to combine software obfuscation [3,4,17,18] and software watermarking to develop more secure software watermarks.

References

1. Aleman-Meza, B., Burns, P., Eavenson, M., Palaniswami, D., Sheth, A.: On ontological approach to the document access problem of insider threat. In: ISI 2005. Volume 3495 of LNCS. (2005) 486–491
2. Chen, Y., Chu, W.W.: Databases security protection via inference detection. In: IEEE ISI 2006. Volume 3975 of LNCS. (2006) 452–458
3. Collberg, C., Thomborson, C., Low, D.: A taxonomy of obfuscating transformations. In: Tech. Report, No.148, Dept. of Computer Sciences, Univ. of Auckland. (1997)
4. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: POPL'98. (1998) 184–196
5. Collberg, C., Thomborson, C., Low, D.: On the limits of software watermarking. In: Technical Report #164, Department of Computer Science, The University of Auckland. (1998)
6. Collberg, C., Thomborson, C.: Software watermarking: Models and dynamic embeddings. In: Proceedings of Symposium on Principles of Programming Languages, POPL'99. (1999) 311–324
7. Collberg, C., Thomborson, C.: Watermarking, tamper-proofing, and obfuscation - tools for software protection. IEEE Transactions on Software Engineering **28** (2002) 735–746
8. Mehrotra, S., Zeng, D.D., Chen, H., Thuraisingham, B.M., Wang, F.Y.: Intelligence and security informatics. In: ISI 2006. Volume 3975 of LNCS. (2006)
9. Monden, A., Iida, H., ichi Matsumoto, K., Inoue, K., Torii, K.: Watermarking java programs. In: International Symposium on Future Software Technology '99. (1999) 119–124
10. Moulin, P., O'Sullivan, J.: Information–theoretic analysis of information hiding. IEEE Transactions on Information Theory **49** (2003) 563–593
11. Myles, G., Collberg, C.: Software watermarking through register allocation: Implementation, analysis, and attacks. In: LNCS 2971. (2004) 274–293
12. Nagra, J., Thomborson, C., Collberg, C.: A functional taxonomy for software watermarking. In Oudshoorn, M.J., ed.: Twenty-Fifth Australasian Computer Science Conference (ACSC2002), Melbourne, Australia, ACS (2002)
13. Nagra, J., Thomborson, C.: Threading software watermarks. In: IH'04. (2004)
14. Thomborson, C., Nagra, J., Somaraju, He, Y.: Tamper-proofing software watermarks. In: Proc. Second Australasian Information Security Workshop(AISW2004). (2004) 27–36
15. Xia, Z., Jiang, Y., Zhong, Y., , Zhang, S.: A novel policy and information flow security model for active network. In: ISI 2004, LNCS. Volume 3073. (2004) 42–55
16. Zhu, W., Thomborson, C., Wang, F.Y.: A survey of software watermarking. In: IEEE ISI 2005. Volume 3495 of LNCS. (2005) 454–458
17. Zhu, W., Thomborson, C.: 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) 208–212
18. Zhu, W., Thomborson, C., Wang, F.Y.: Application of homomorphic function to software obfuscation. In: WISI 2006. Volume 3917 of LNCS. (2006) 152–153
19. Zhu, W., Thomborson, C.: Extraction in software watermarking. In: ACM MM&Sec'06, Geneva, Switzerland. (2006) 175–181
20. Zhu, W., Thomborson, C.: Recognition in software watermarking. In: 1st ACM Workshop on Content Protection and Security, in conjuction with ACM Multimedia 2006, October 27th, 2006, Santa Barbara, CA, USA. (2006) 29–36