An Abstract Interpretation-Based Framework for Software Watermarking

Patrick Cousot
École normale supérieure
Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

Radhia Cousot
CNRS & École polytechnique
Radhia.Cousot@polytechnique.fr
www.stix.polytechnique.fr/~rcousot

POPL®'2004, Venice, Italy 14–16 Jan 2004

Requirements

Making the Software Watermarking Algorithms Public

- More confidence in public algorithms;

$\Rightarrow$ Make the embedding/extraction algorithms public;

$\Rightarrow$ By parameterizing with a secret:

Watermark embedding:

Program $\times$ Signature $\times$ Secret $\longrightarrow$ Watermarked program

Watermark extraction:

Watermarked program $\times$ Secret $\longrightarrow$ Signature

Both the signature and secret should be invisible in the watermarked program.

Stegomark Approach to Software Watermarking

Dynamic Software Watermarking

- Semantics-based approach
- Watermark embedding: the signature is hiden in the semantics of the stegomark
- Watermark extraction: execution of watermarked program with the secret input reveals the signature:

  Dynamic data structure watermarking: by building a data structure containing the signature

  Dynamic execution trace watermarking: by generating a succession of events (adresses/operations/...) encoding the signature

$\Rightarrow$ more robust (Collberg & Thomborson [POPL'97 & 98])

- Abs
- Wat sem prog
- Wat stat ceed

Formalization of Abstract Software Watermarking

1.a) Ingredients of a concrete semantics

- Programs: $P \in \mathbf{Program}$
- Concrete semantic domain: $\mathcal{D}$
- Concrete semantics of programs: $S \in \mathbf{Program} \mapsto \mathcal{D}$
- Observability abstraction: $\alpha_{\mathcal{O}}$

such that:

$\forall P \in \mathbf{Program}$, only $\alpha_{\mathcal{O}}(S[\![P]\!])$ is of interest

- Observability equivalence: $\equiv_{\mathcal{O}}$

$$P \equiv_{\mathcal{O}} P' \Leftrightarrow \alpha_{\mathcal{O}}(S[\![P]\!]) = \alpha_{\mathcal{O}}(S[\![P']\!])$$

Form

1.c) V

- Sign
- Sign
- Steg
- Steg

I[S

Form

2) Em

Formalization of Abstract Software Watermarking (Cont'd)

3) Extraction

Watermark extraction:

Watermarked program $\times$ Secret $\longrightarrow$ Signature

Signature extraction from P is by static analysis:

$E[\text{Secret}](S^\sharp[\text{Secret}][\![P]\!])$

Formalization of Abstract Software Watermarking (Cont'd)

5) Resistance to attacks

- Signature extraction without the secret is hard:
  - Computing $S^\sharp[?][\![I[?](P,Q)]\!]$ is hard
- Recovering the original program/stegomark elimination is hard:
  - Computing P from $I[\text{Secret}](P,Q)$ is hard (without Q)
- Ideally, stegomark obfuscation should be effectless:

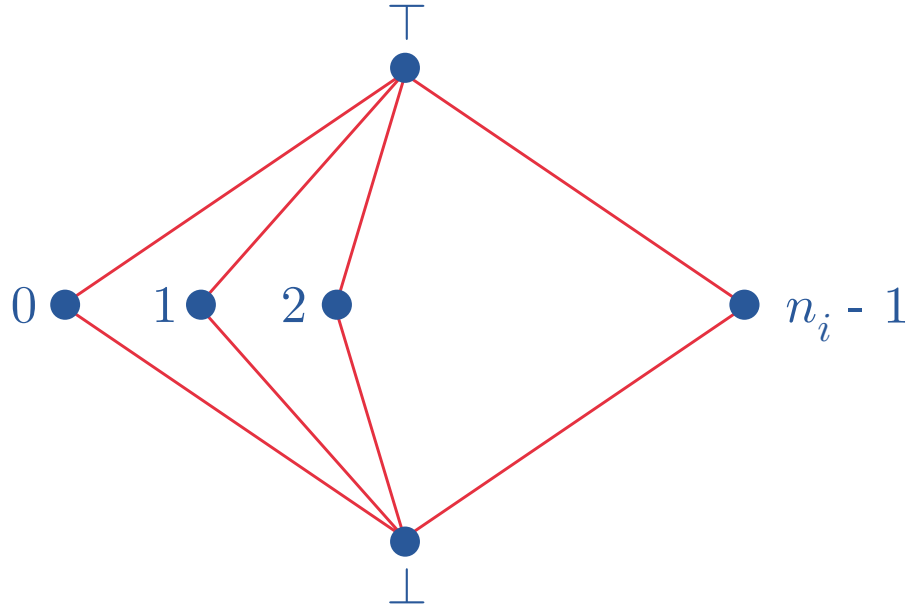
  If $\quad Q = M[\text{Secret}](A[\text{Secret}](\text{Signature}))$
  and $\quad P' \equiv_{\mathcal{O}} I[\text{Secret}](P,Q)$
  then $S^\sharp[\text{Secret}][\![P']\!] = S^\sharp[\text{Secret}][\![I[\text{Secret}](P,Q))]\!]$

- Pro
- Con
- Obs
  -
  -
  -

Static analysis

- ℓ times a variant of Kildall's constant propagation modulo the secret n_i:

$$\mathcal{D}^\sharp = \prod_{i=1}^{\ell} \mathcal{D}_i^\sharp \quad \text{where} \quad \mathcal{D}_i^\sharp =$$



- Extend pointwise/componentwise to environments, program points, etc.

ℓ steg

- W is prop
- W is

Obfuscating the stegomark for c_i

Obfuscation for 2^{nd} degree polynomials (computed by Horner method):

- $P(x) = (x - k_1)x + k_2$
  where $k_1 = (1 + c_i) + r_1.n_i)$
  $k_2 = (c_i + r_2.n_i)$
  r_1 and r_2 are random numbers;
- idem for $Q(x)$.

- Ass
- Can
  Finc
  – ex
  – an
  – ha
- Inde
  info
- So,
  the

Example of Watermarked Program

```
public class Fibonacci {
  public Fibonacci() {}
```

Attacks on erasing the stegomark for c_i

The stegomark contains:

- unusual large integer constants
- auxiliary variables with almost stochastic integer values in $\mathbb{Z}$

that might be recognized by monitoring the watermarked program execution to reveal the stegomark components for some c_i where $i \in [1, \ell]$

Cou

3) H
of c
-
-
-
inte
-
-

Sec

Obfus

Counter-attack on obfuscating the stegomark for c_i

1) obfuscate the watermarked program before distribution

2) refine the static analyzer

When

- The
- The
  und
  feas
- Not
  goo