Software Obfuscation and Content Protection

Aggelos Kiayias

Obfuscation

- Confuse the meaning while retain the functionality?
- Is it possible at all?

Example

- Compare:
  - “Your location is the ENGR building; go to the ITE building and on the second floor enter room 201.”
  - “Your location is 41°48’35.03N’ 72°15’23.43W. Walk 20 steps SE, then turn NE, walk 15 steps, turn SE walk 100 steps ...”

Recreational

What do these two programs do?

```
#include "stdio.h"
#define e 3
#define g (e/e)
#define h ((g+e)/2)
#define f (e-g-h)
#define j (e*e-g)
#define k (j-h)
#define l(x) tab2[x]/h
#define m(n,a) ((n&(a))==(a))

long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
        case f:
            a=(b=(c=(d=g)<<g)<<g)<<g;
            return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
        case h:
            for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
        case g:
            if(n<h)return(g);
            if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
            else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
            if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
            return(o[b-g]%n+k-h);
        default:
            if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
            for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
        }
}
```

```
#include <stdio.h>
int main()
  {
  printf("Hello, World.");
  return 0;
  }
```

http://www.cise.ufl.edu/~manuel/obfuscate/obfuscate.html

Software Obfuscation

- Usefulness?
  - resistance to reverse engineering.
- Applications:
  - enforcing software licensing.
  - protecting software contents.
  - hiding keys into software.

What is an obfuscator?

- A compiler O: **L** => **L**
  - Suppose O(P) = P'
- P' has the same I/O as P.
- P' is not significantly slower or larger than P.
- P' is more obscure than P.

Lexical Obfuscation

```
private void CalcPayroll(SpecialList employeeGroup) {
   while (employeeGroup.HasMore()) {
        employee = employeeGroup.GetNext(true);
        employee.updateSalary();
        DistributeCheck(employee);
    }
}
```

```
private void a(a b) {
    while (b.a()) {
        a = b.a(true);
        a.a();
        a(a);
    }
}
```

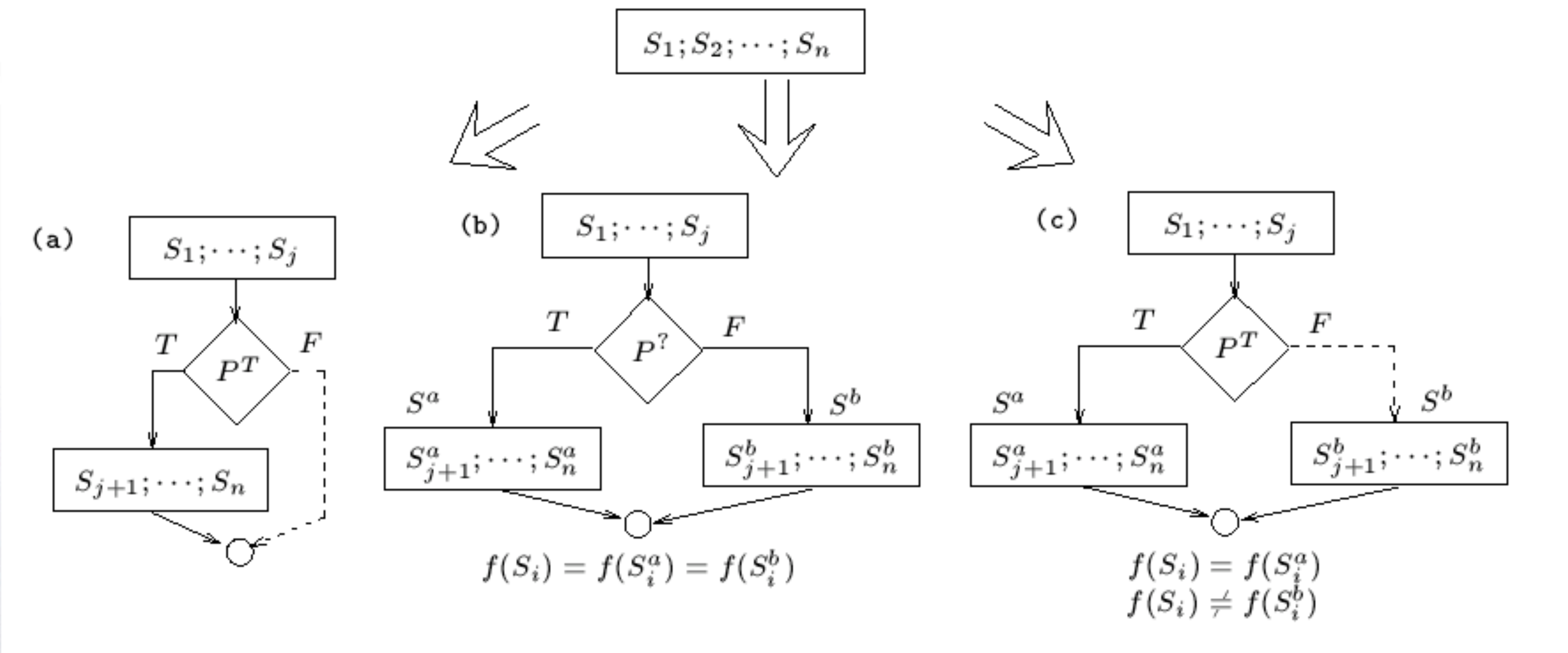
What is the advantage?

What is the disadvantage?

Example taken from http://www.preemptive.com/

Tampering with Control Flow

The branch insertion transformation



From: Christian Collberg, Clark Thomborson, Douglas Low, *A Taxonomy of Obfuscating Transformations*

Opaque Predicates

- Control flow obfuscation can be successful if we can devise predicates that:
  - have easily predictable values during obfuscation [based e.g., on local coins].
  - are hard to predict from the code only (outside of runtime).

Examples:

```
if (2>1) then ....
if (a==a) then ...
if Is_prime(23*2^496422-1) then ...
if (rand(10)>10) then ...
if (2 divides a^2 + a) then ...
...
a=5; b=7;
...
if (a<b) then ...
```

What constitutes a bad example? what would be better?

Static Analysis

- Given a program P analyze its properties without executing.
- Data flow analysis: reveal the interdependency of data in a sequence of statements.

Data Flow Example

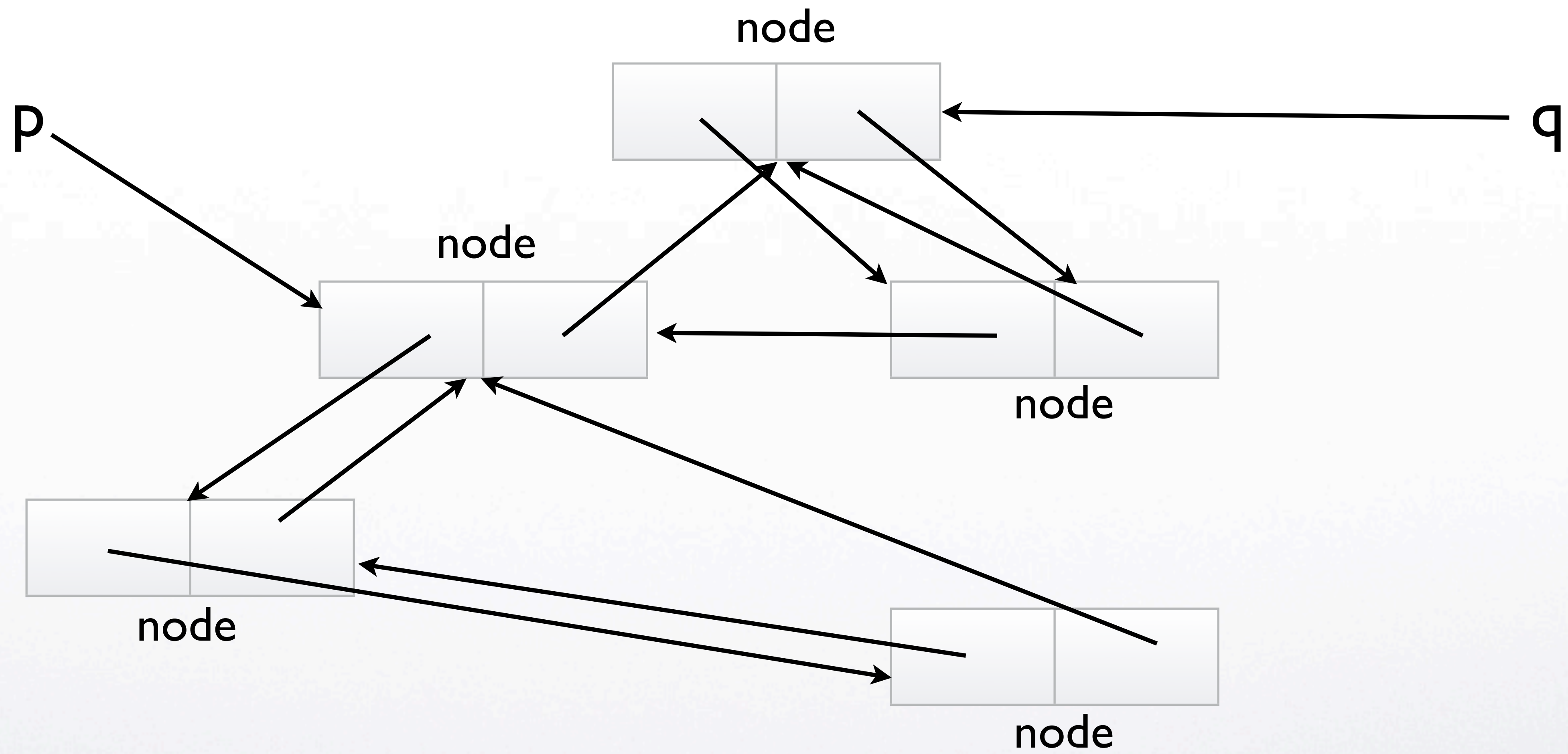
```
int function (int a, int b)
{
    int c=1,d=2;

    if (c == a+2)

        d = 10;

    if (d<=5)

        c = a+b;

    return d;
}
```

is used for code optimization.

Would reveal syntactic interdependencies and potentially used to compute opaque predicates.

Note : code optimization may undo some naive obfuscations.

Complex objects



Construct a complex object at random carrying two pointers p,q. Split the structure at some point into two disjoint components carrying p,q. Use p==q as a (false) opaque predicate.

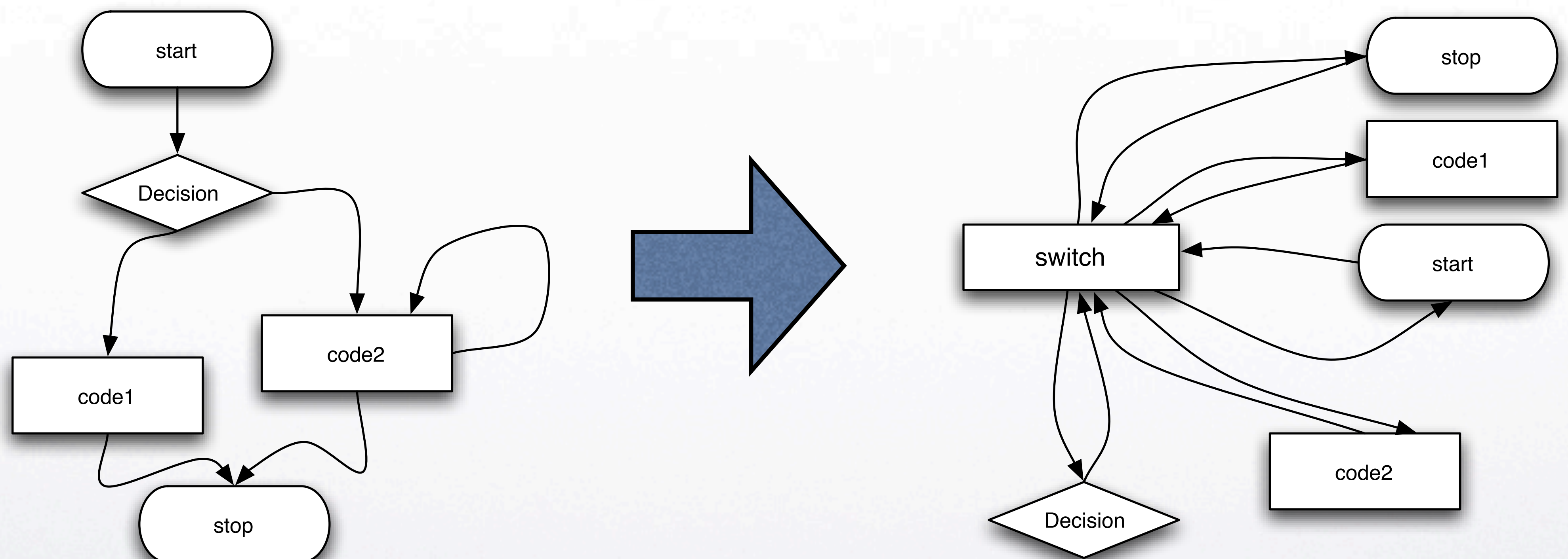
Shortcomings

- Knowledge of how obfuscator *works* can allow the reverse-engineer to defeat the obfuscation.
- Coping techniques: try to make opaque predicate declarations and operations very close to actual program declaration and operations.

Control Flow Analysis

- Static analysis of program control flow.
- Identify loops, conditionals and other important control structures.
- Flattening of control flow makes analysis more difficult.

Control Flow Flattening



In practice...

- Many tools exist especially for languages that are distributed in source code (JAVA, Perl, PHP, .NET) but also for general assembly.
- Obfuscators usually perform other primary functions such as reducing program size or speed/power optimizations.
- Examples: Sandmark, IBM JAX for Java, Diablo for machine code, many other packages.
- Obfuscation is *not* encryption

Hash-based Obfuscation

```
if (x == 17) then return 1 else return 0
```

obfuscator:

1. chooses `R`
2. computes `A = HASH(R,17)`

```
R = 15905123523
A = 68598209348
if HASH(R,x) = A then return 1 else 0
```

What does this technique remind you?

Technique hides 17 as long as "17" is not known :-)

entropy?

how can you break it?

Hash-based Obfuscation, II

```
if (x == 17) then return 410 else return 2104
```

obfuscator:

1. choose `R, R', R"`
2. computes `A = HASH(R,17)`
3. computes `B = HASH(R',17)`
4. computes `C = HASH(R",17)`
5. computes `D = B xor 410`
6. computes `E = C xor 2104`

```
R  = 15905123523
R' = 49320294012
R" = 66872035409
A = 68598209348; D = 90591213366; E = 48623049585;
if HASH(R,x) = A then return D xor HASH(R',x) else E xor HASH(R",x)
```

Applications

- Applies only to cases where the labels to be hidden have sufficient entropy.
- Suitable for access control mechanisms and certain database applications.

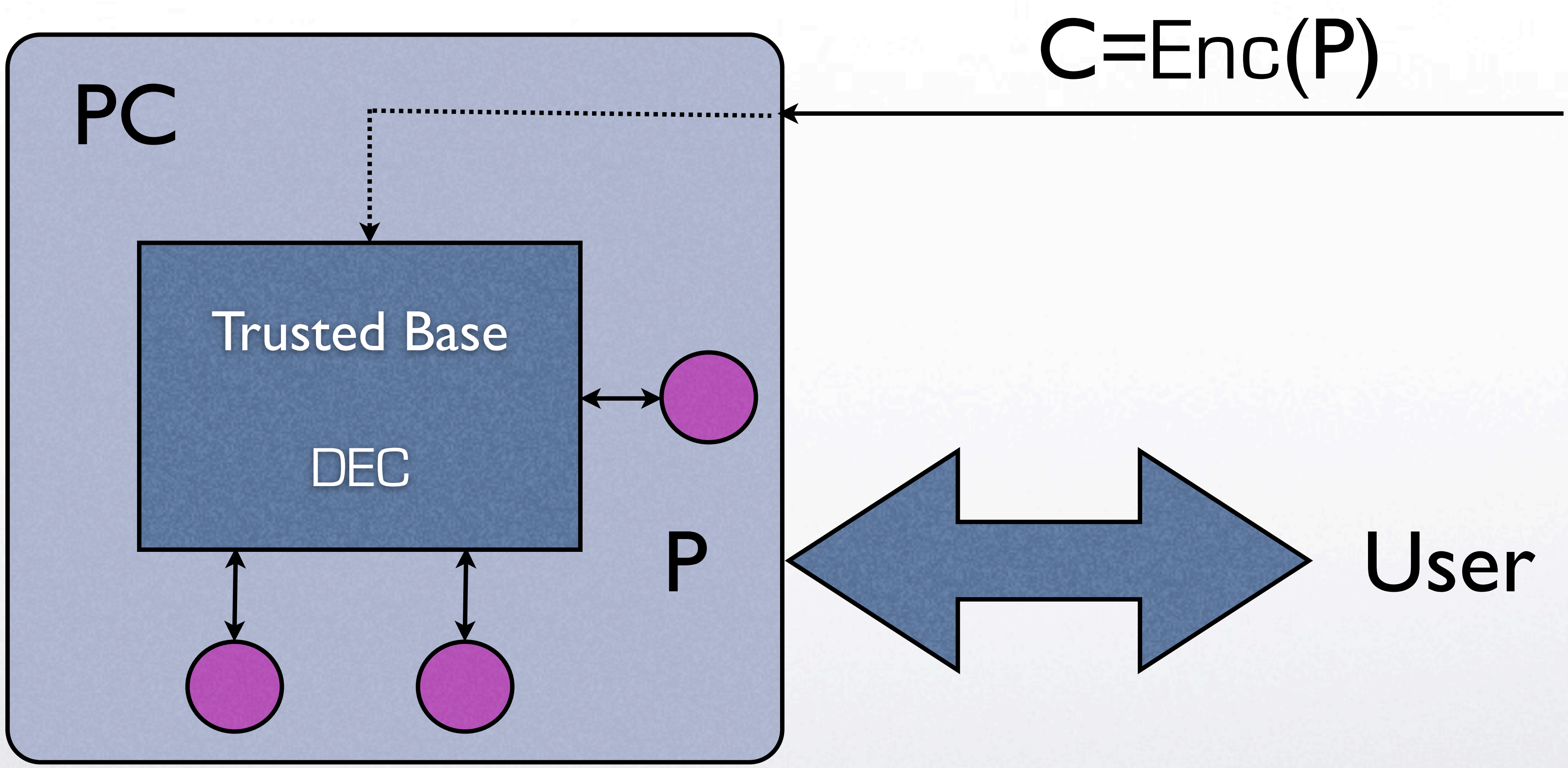
General Obfuscation

- Formally modeling the concealing property of an obfuscator (in line with modern cryptography definitions).
- An obfuscator O: **L** => **L** *satisfies the virtual black-box property against resource* G if: any G-resource-bounded adversary that is given the code of O(P) can be simulated using **only** I/O access to P.
- *Impossibility* result due to Barak et al. [Crypto '01] for polynomial-time resource bounds.

Impossibility Results

- Impossibility results based on explicit constructions of unobfuscatable function families.
- This leaves the possibility that some useful function families exist that can be obfuscated.

The Trusted Base approach



Code Watermarking

- Goal: embed a watermark into the code.
- Usefulness:
  - Ability to stamp code / *claim ownership*.
- Requirement:
  - Should be hard to remove the watermark.

Code Fingerprinting

- Goal: embed a fingerprint into the code.
- Usefulness:
  - Ability to personalize code / *track code redistribution*.
- Requirement:
  - Should be hard to remove the fingerprint also taking collusion attacks into account.

Embedding Techniques

- Identify redundancy in code.
- Take advantage of redundancy to embed information.

Simple example to embed *one* bit:

```
if A then B else C <=> if (not A) then C else B
```

Such watermarks can be destroyed.

Embedding Techniques 2

- Dynamic techniques may also rely on execution traces or run-time inspection of data structures.
- e.g., create linked list structure whose pointer links can be parsed to identify the embedding.
- Error-correction can be used to make embedding somewhat robust.

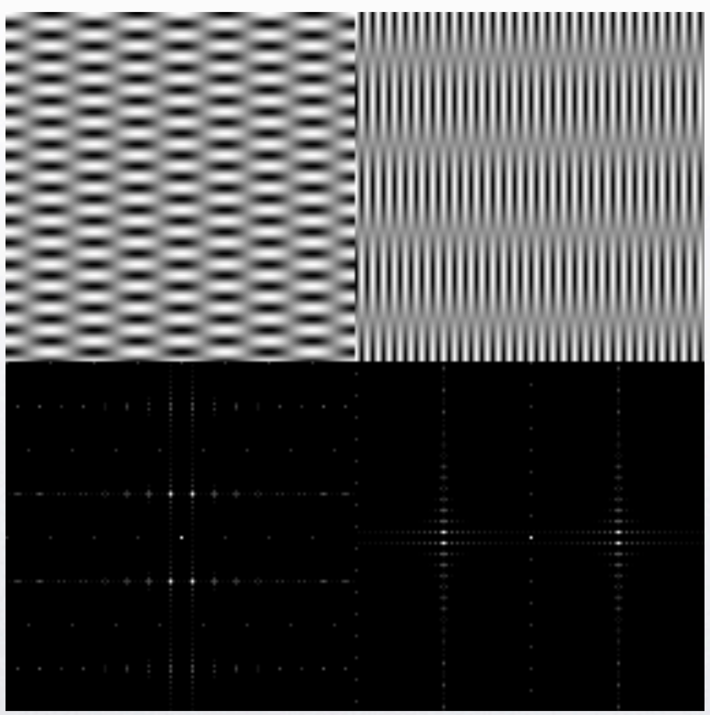
C. Collberg, S. Kobourov, E. Carter and C. Thomborson, Graph-Based Approaches to Software Watermarking, Graph-theoretic concepts in Computer Science 2003.

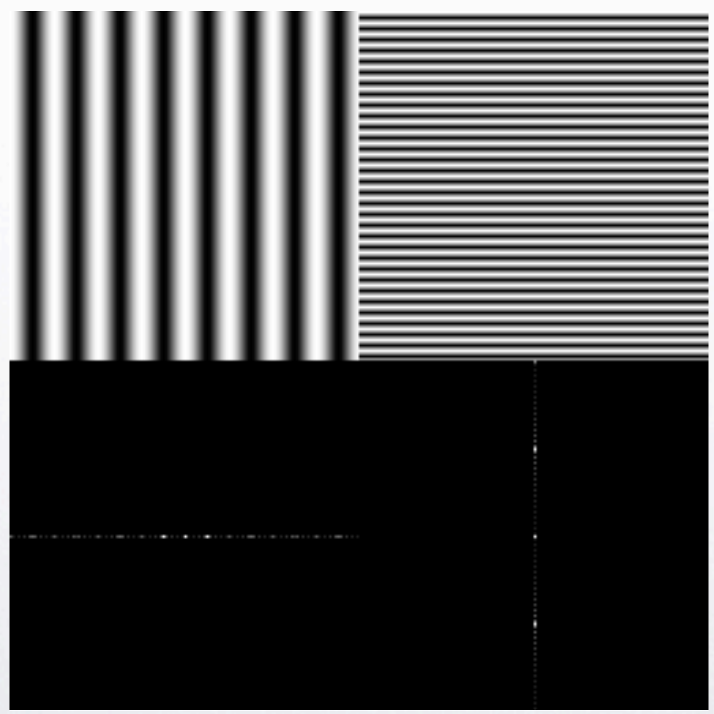
Image Embeddings

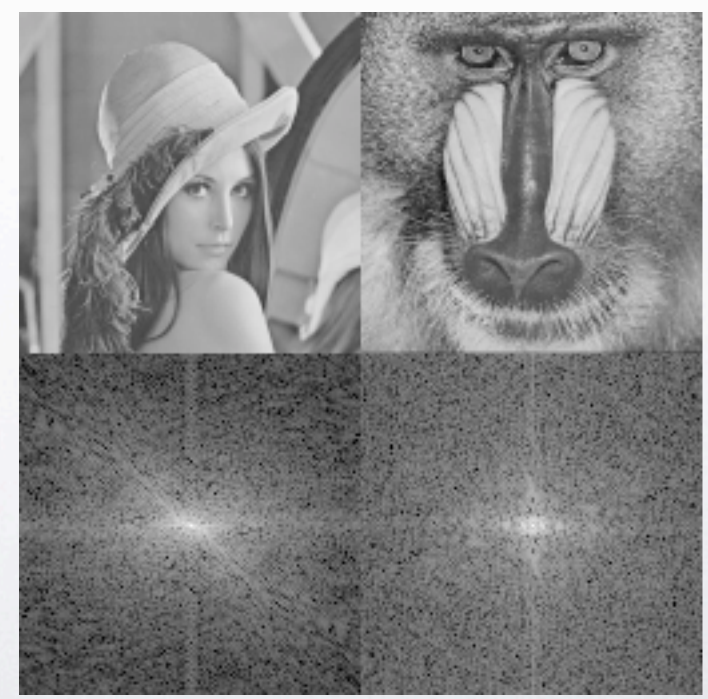
Use Discrete Fourier Transform

$$F(u,v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m,n] e^{-j2\pi(umx_o + vny_o)}$$

Analysis of Image into sinusoidal components



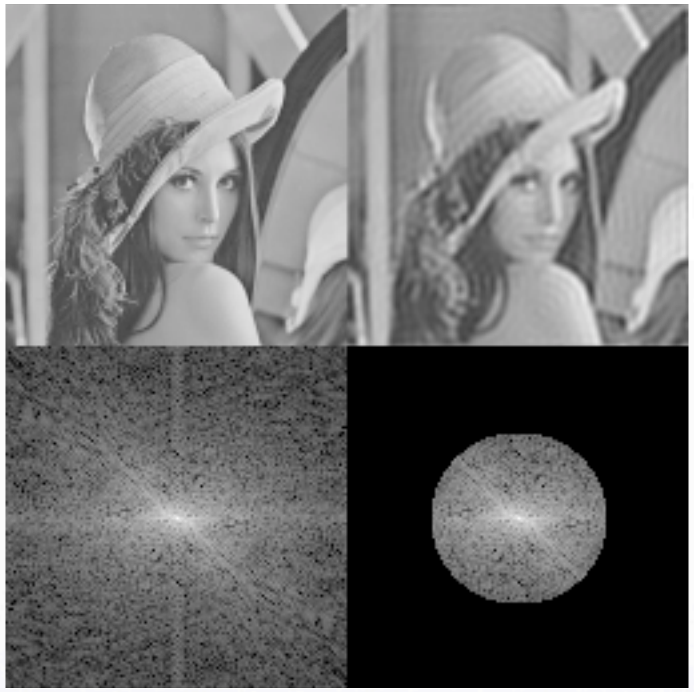




images from http://www.cs.unm.edu/~brayer/vision/fourier.html

Embedding with DFT's

effects of a low pass filter:



Given image compute DFT:
Then embed watermark information in the high amplitude image components "local peaks" (e.g., closer to center in pic above)

Fingerprinting via Embeddings

- If an embedding is robust we can read it even if some manipulation occurs.
- Robustness suggests minimal marking bandwidth.
- How to distinguish between marked objects ? What about collusions ?

Tool : Collusion Secure Codes

$$\langle n, v\rangle_2 - \text{collusion-secure-code } \mathcal{C}$$

Definition 26 *Given a set of codewords $C = \{\omega_{i_1}, \ldots, \omega_{i_t}\} \subseteq \mathcal{C}$ an undetectable position is a location $i \in \{1, \ldots, v\}$, such that $(\omega_{i_1})_i = \ldots = (\omega_{i_t})_i$. The set of undetectable positions is denoted by $U(C)$. The* feasible set *of C denoted by $F(C)$ is defined as:*

$$F(C) = \left\{\omega \in \{0, 1, ?\}^v \mid (\omega)_{U(C)} = (\omega_{i_1})_{U(C)}\right\}$$

Marking assumption : *A set of colluders can only compute a feasible codeword*

Traceability : *Given a feasible codeword recognize one colluder.*

Tool : Collusion Secure Codes

$\langle n, v \rangle_2$ − collusion-secure-code $\mathcal{C}$

Definition 26 *Given a set of codewords* $C = \{\omega_{i_1}, \ldots, \omega_{i_t}\} \subseteq \mathcal{C}$ *an undetectable position is a location* $i \in \{1, \ldots, v\}$, *such that* $(\omega_{i_1})_i = \ldots = (\omega_{i_t})_i$. *The set of undetectable positions is denoted by* $U(C)$. *The* feasible set *of* C *denoted by* $F(C)$ *is defined as:*

$$F(C) = \left\{\omega \in \{0, 1, ?\}^v \mid (\omega)_{U(C)} = (\omega_{i_1})_{U(C)}\right\}$$

Marking assumption : *A set of colluders can only compute a feasible codeword*

how to enforce?

Traceability : *Given a feasible codeword recognize one colluder.*

Example

Introduced [BS95]
[SW98,SW01]
Construction [Tar03]

$$v = \mathcal{O}(c^2 \log(n/\epsilon))$$

colluder 1	0011001100
	0000011111
colluder 2	0011000111
	----------
feasible set	001100?1??

code for two colluders:

C_1 : 1 0 0 0 1 1 0
C_2 : 1 1 1 0 0 0 0
C_3 : 0 0 1 1 1 0 0

1 1 0 0 1 0 0 ?

Fragile Embeddings

- Any perturbation of the program that destroys the watermark destroys the program's functionality.
- Existence of Obfuscation and existence of Fragile code watermarking is inconsistent.