



Software Watermarking

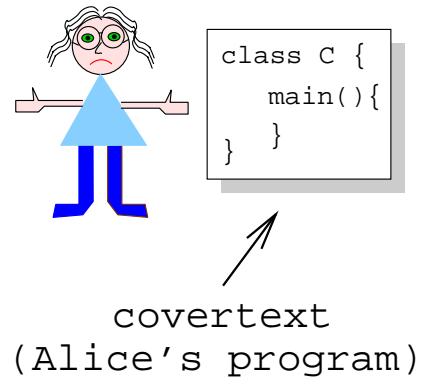
Christian Collberg
University of Arizona

September 17, 2006



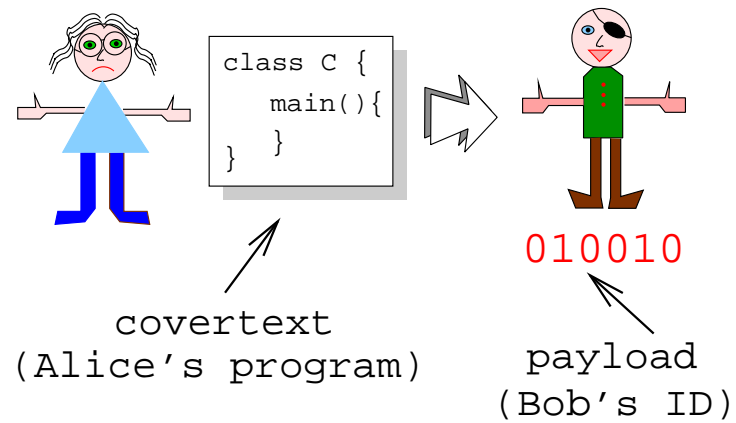
Hiding Secrets in Software

Introduction



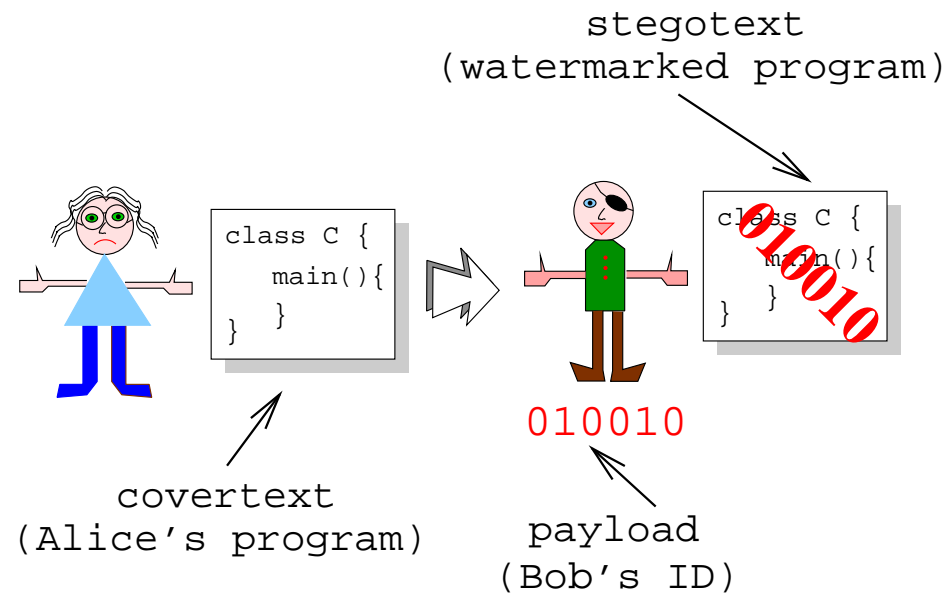
Hiding Secrets in Software

Introduction



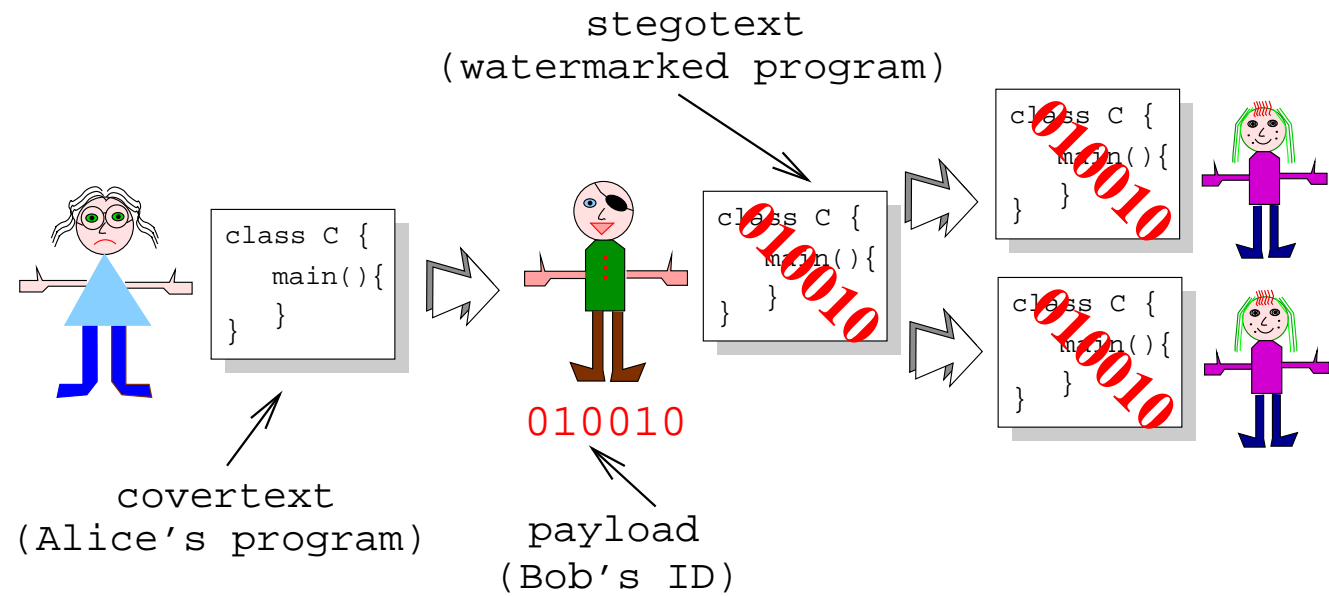
Hiding Secrets in Software

Introduction



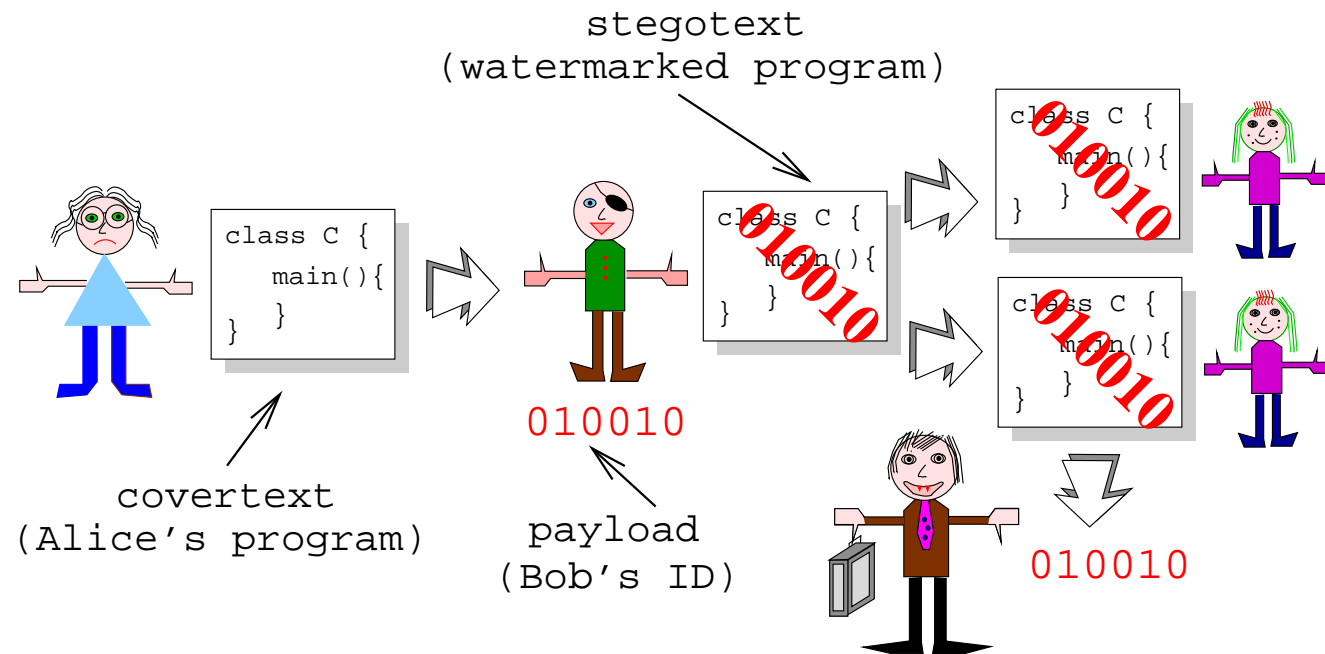
Hiding Secrets in Software

Introduction



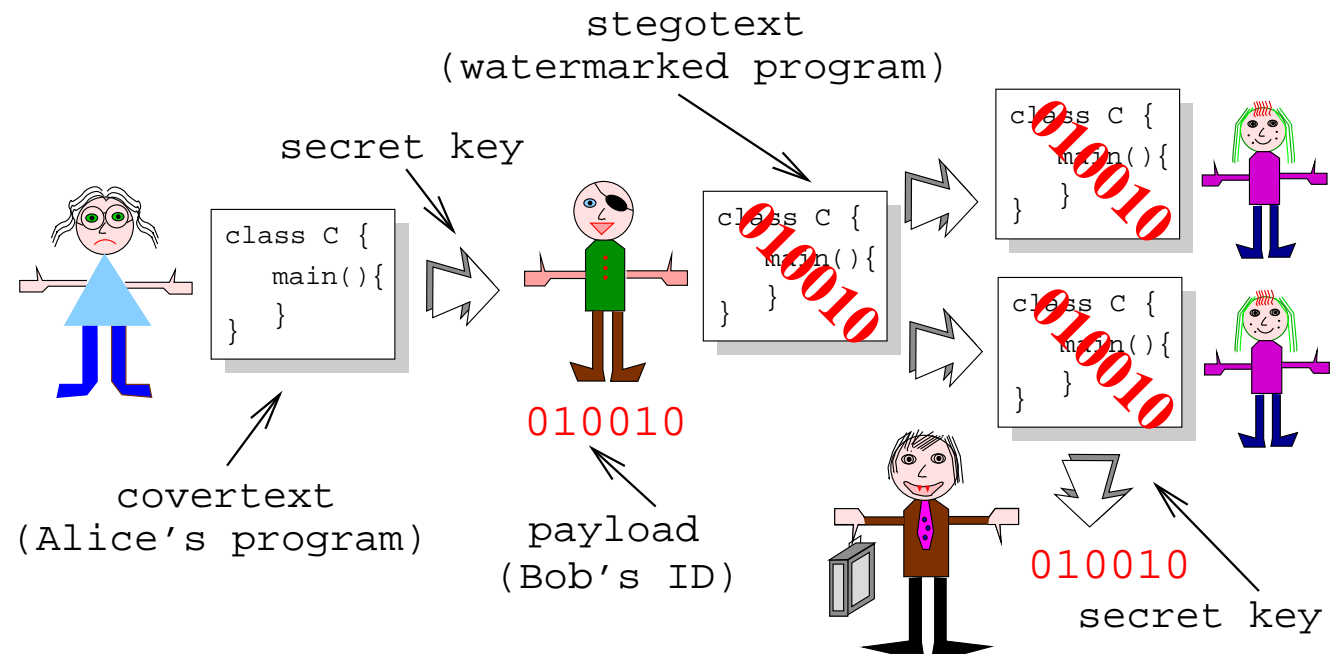
Hiding Secrets in Software

Introduction



Hiding Secrets in Software

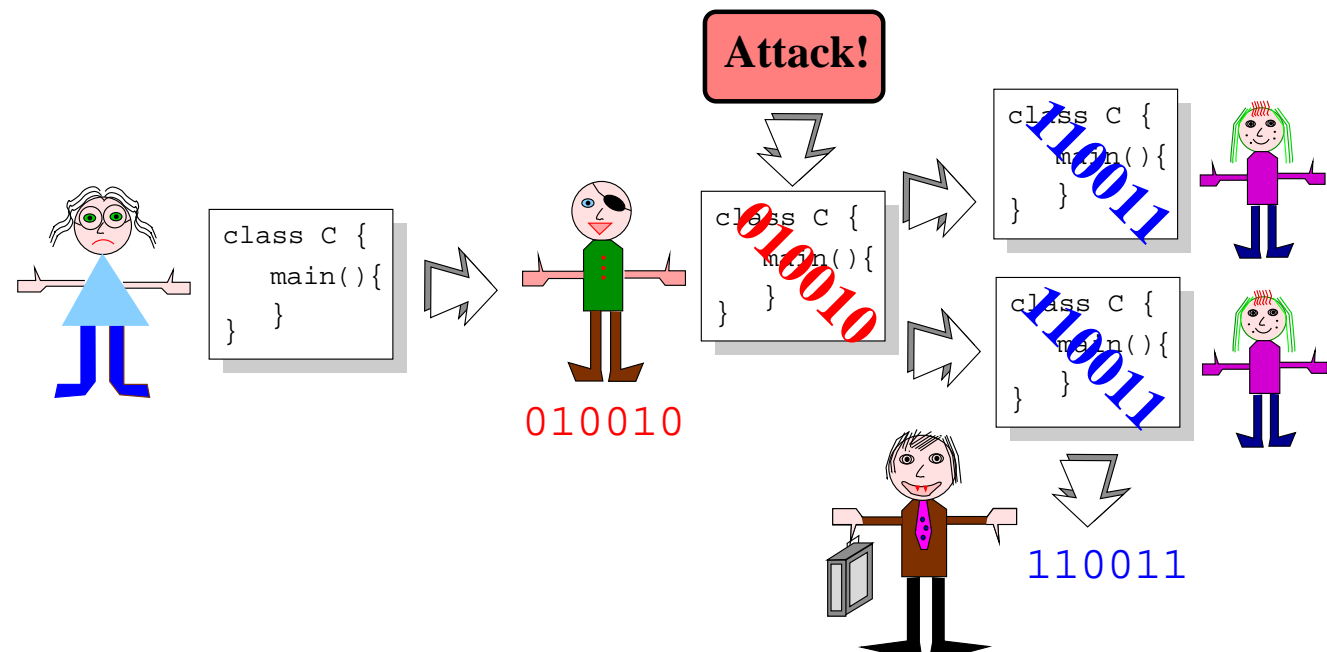
Introduction





Hiding Secrets in Software

Introduction



- Applications: Software fingerprinting, steganography,...



Prisoners' Problem

Alice, Bob, and
Wendy

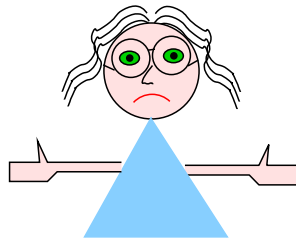
Prisoners' Problem

Cryptography

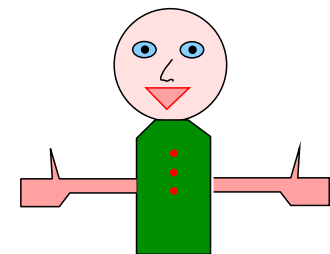
Steganography

Coverttexts

Alice



Bob





Prisoners' Problem

Alice, Bob, and
Wendy

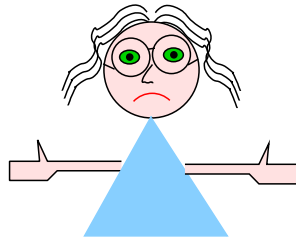
Prisoners' Problem

Cryptography

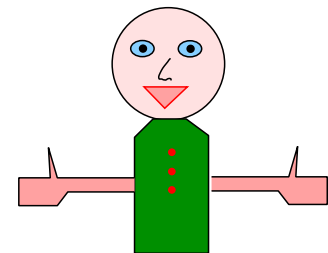
Steganography

Covertypes

Alice

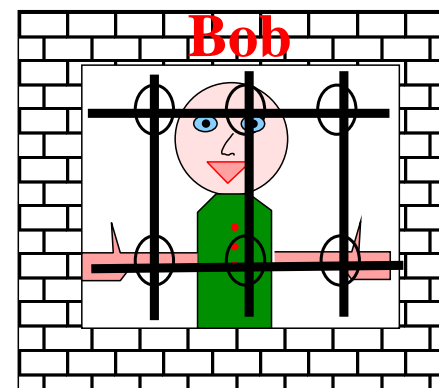
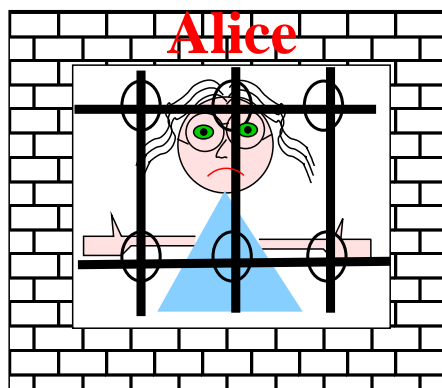
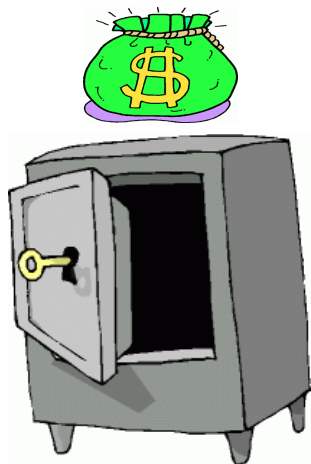


Bob





Prisoners' Problem



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexs



Prisoners' Problem

Alice, Bob, and
Wendy

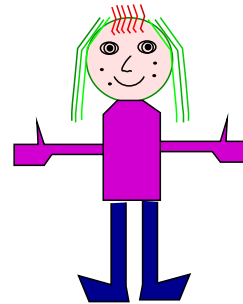
Prisoners' Problem

Cryptography

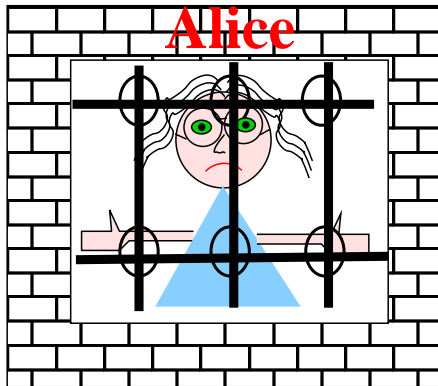
Steganography

Covertex

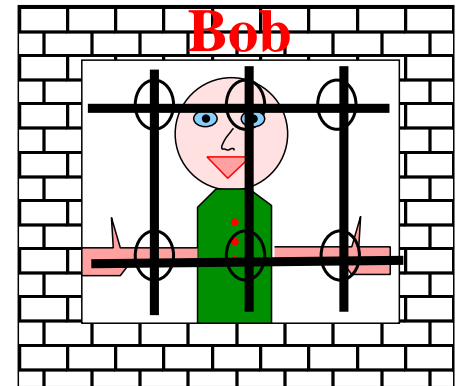
Wendy



Alice



Bob





Prisoners' Problem

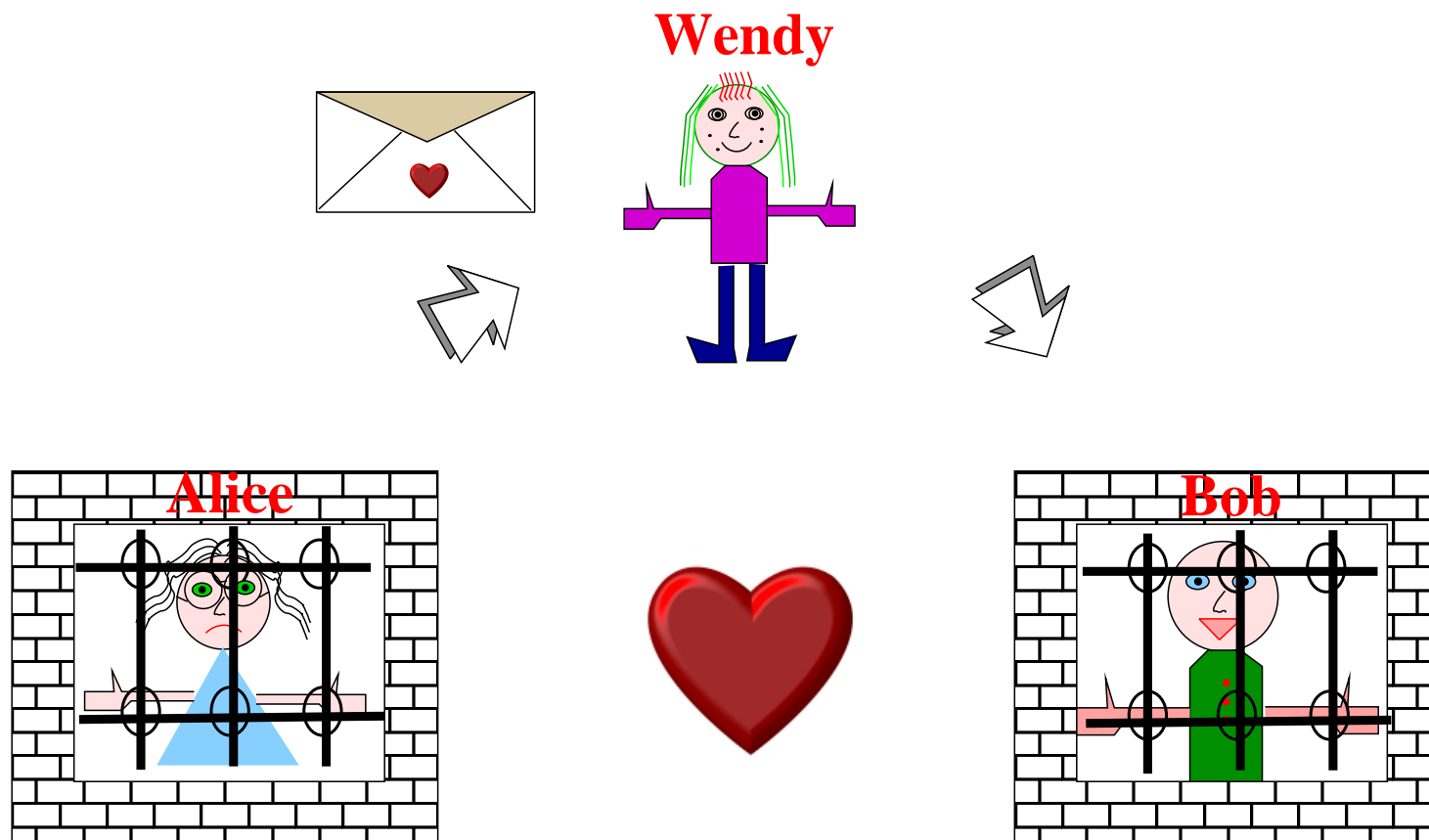
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





Prisoners' Problem

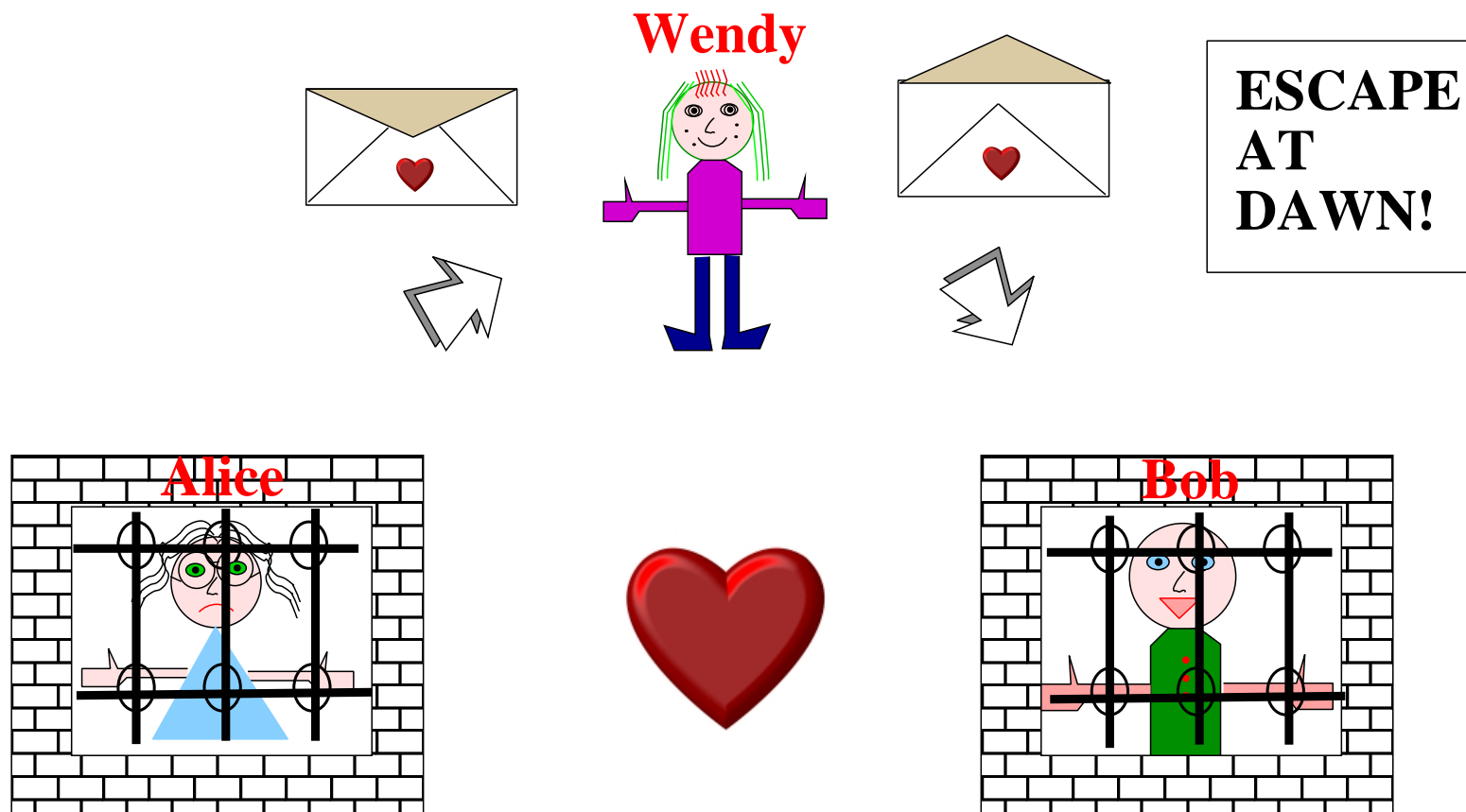
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertex





Prisoners' Problem

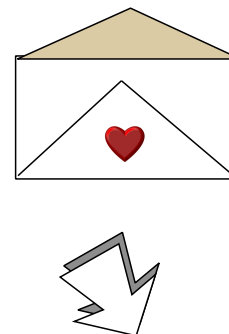
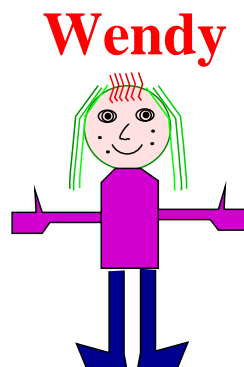
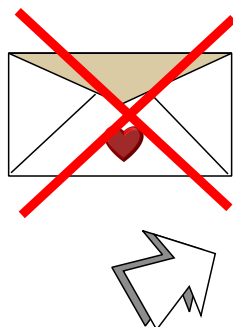
Alice, Bob, and Wendy

Prisoners' Problem

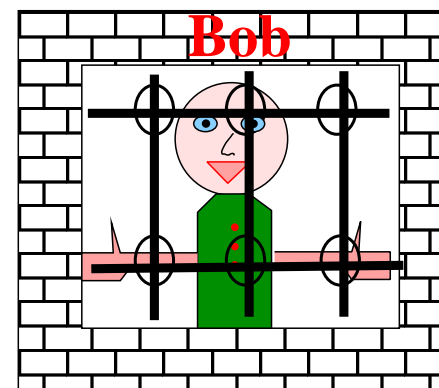
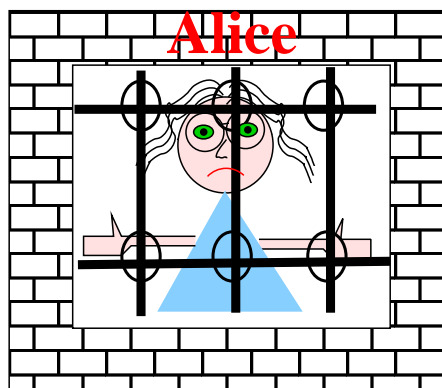
Cryptography

Steganography

Covertypes



**ESCAPE
AT
DAWN!**





First Try — Cryptography

ESCAPE AT DAWN

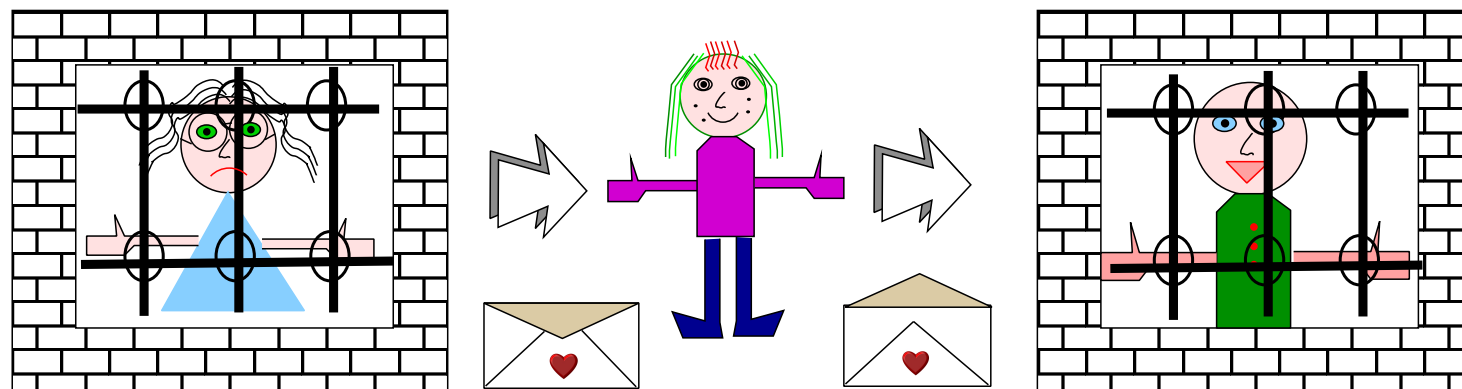
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





First Try — Cryptography

ESCAPE AT DAWN



Encrypt

HJ*^(KDLF*^(DJHGA

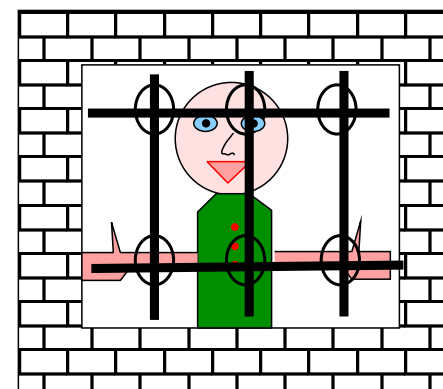
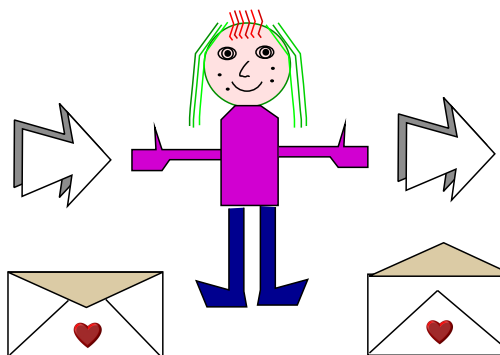
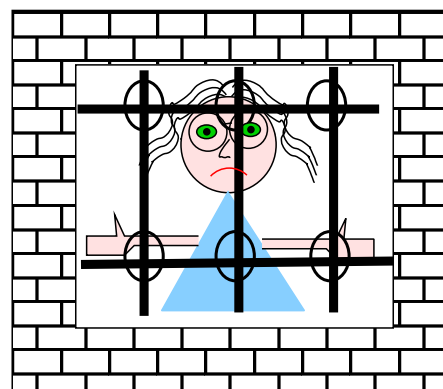
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

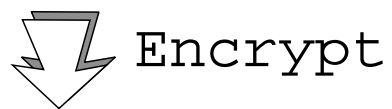
Covertexs





First Try — Cryptography

ESCAPE AT DAWN



$HJ^*(KDLF^*(DJHGA$

$HJ^*(KDLF^*(DJHGA$

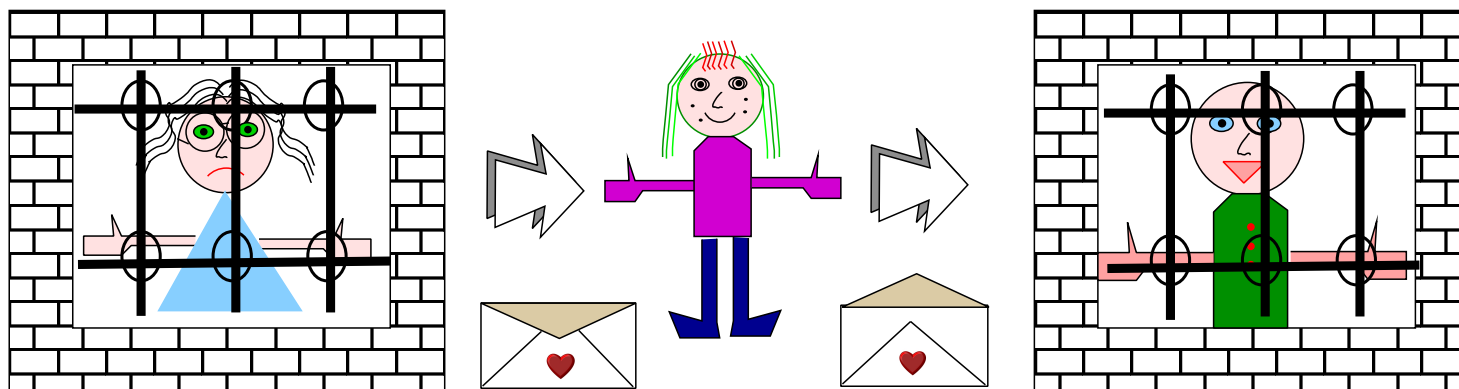
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

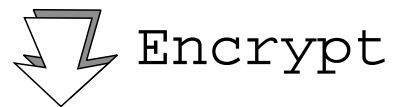
Covertexs





First Try — Cryptography

ESCAPE AT DAWN

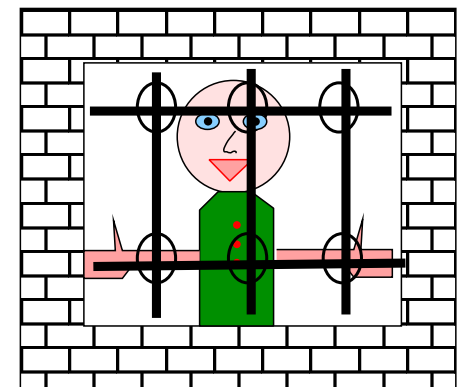
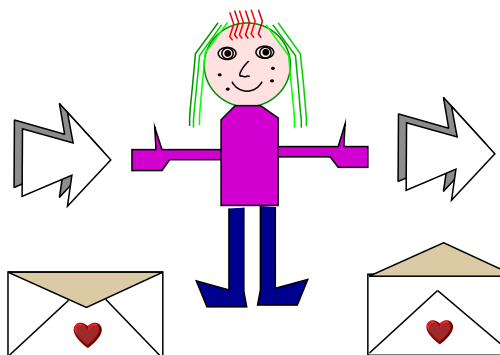
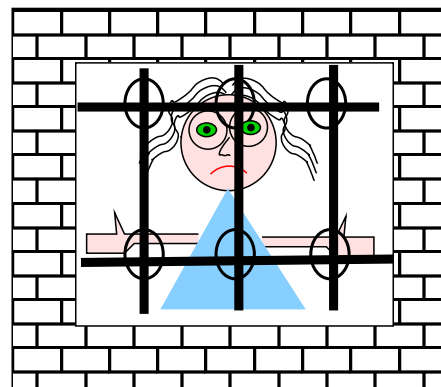


HJ*^(KDLF*^(DJHGA

HJ*^(KDLF*^(DJHGA



ESCAPE AT DAWN



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

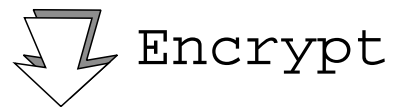
Steganography

Covertexs



First Try — Cryptography

ESCAPE AT DAWN

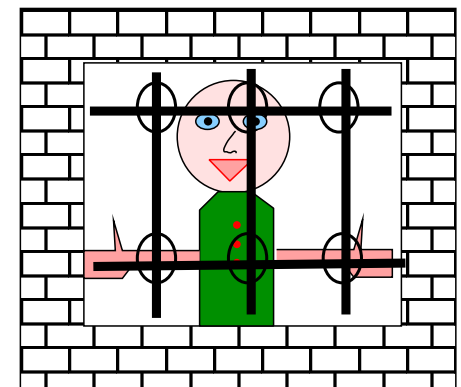
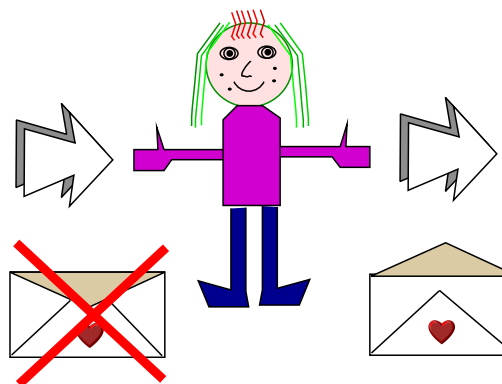
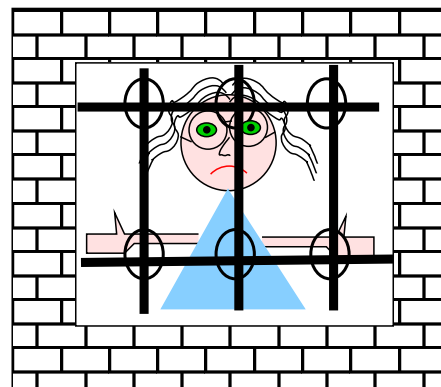


HJ*^(KDLF*^(DJHGA

HJ*^(KDLF*^(DJHGA



ESCAPE AT DAWN



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexs



Second Try — Steganography (A Null Cipher)

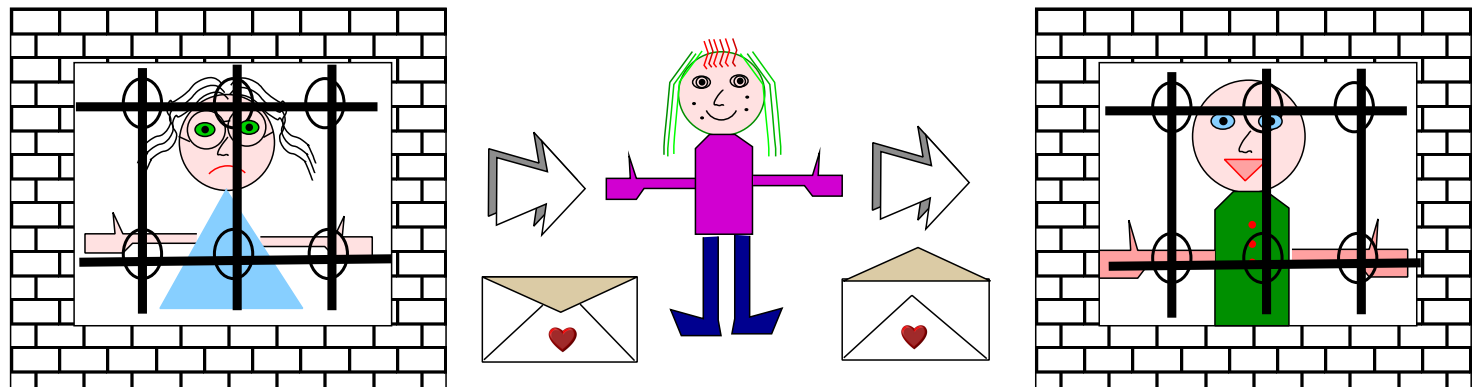
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Coverttexts

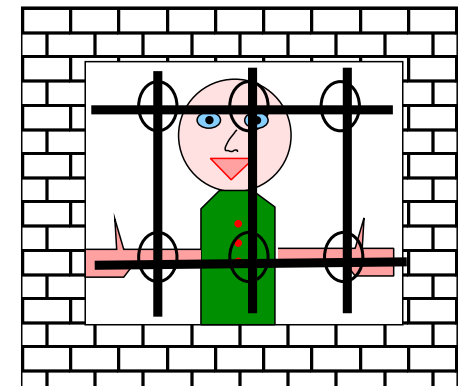
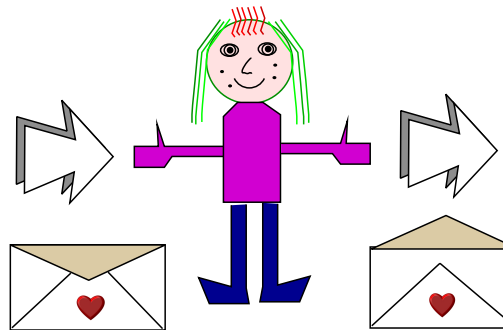
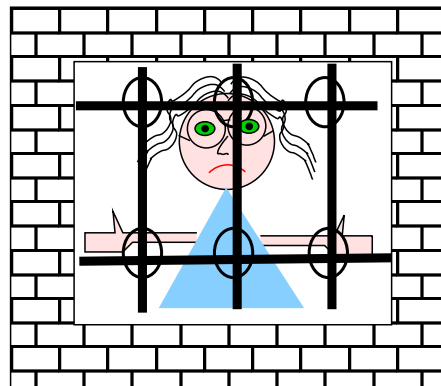




Second Try — Steganography (A Null Cipher)

Alice, Bob, and
Wendy
Prisoners' Problem
Cryptography
Steganography
Coverttexts

Easter is soon, dear!
So many flowers!
Can you smell them?
Are you cold at night?
Prison food stinks!
Eat well, still!
Are you lonely?
Take care!
Don't worry!
All is well!
Wendy is nice!
Need you!):

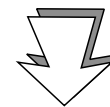




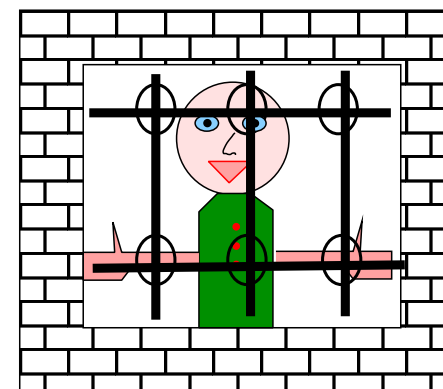
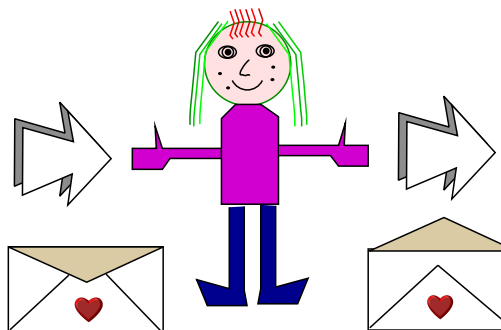
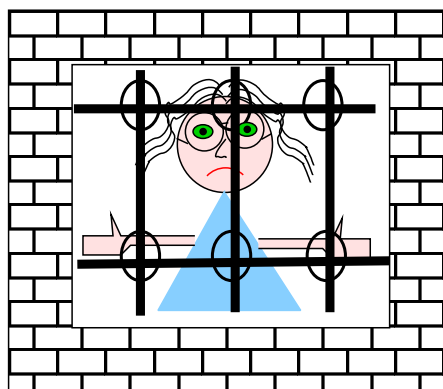
Second Try — Steganography (A Null Cipher)

Alice, Bob, and
Wendy
Prisoners' Problem
Cryptography
Steganography
Coverttexts

Easter is soon, dear!
So many flowers!
Can you smell them?
Are you cold at night?
Prison food stinks!
Eat well, still!
Are you lonely?
Take care!
Don't worry!
All is well!
Wendy is nice!
Need you!):



ESCAPE AT DAWN





Second Try — Steganography (A Null Cipher)

Alice, Bob, and
Wendy
Prisoners' Problem
Cryptography
Steganography
Coverttexts

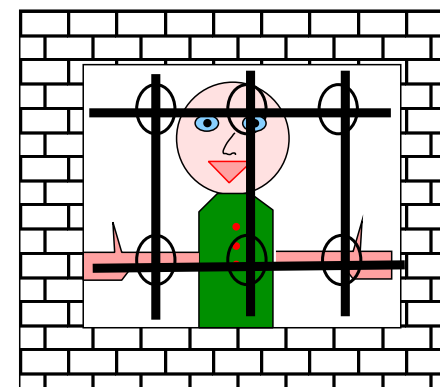
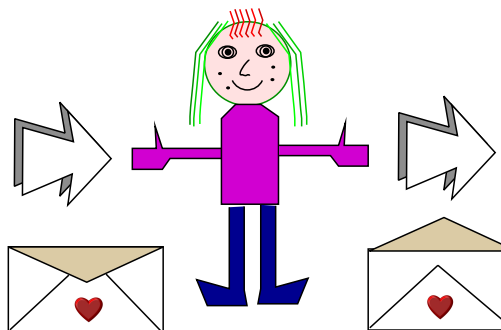
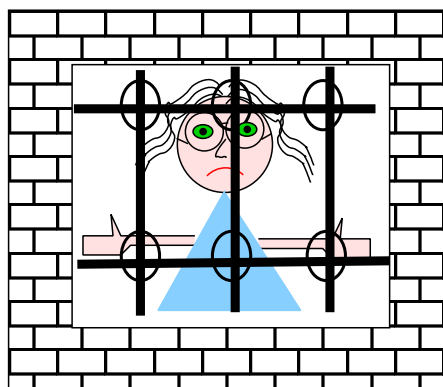
Easter is soon, dear!
So many flowers!
Can you smell them?
Are you cold at night?
Prison food stinks!
Eat well, still!
Are you lonely?
Take care!
Don't worry!
All is well!
Wendy is nice!
Need you!):

Coverttext

**Hidden text
(payload)**



ESCAPE AT DAWN





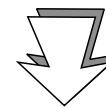
Second Try — Steganography (A Null Cipher)

Alice, Bob, and
Wendy
Prisoners' Problem
Cryptography
Steganography
Coverttexts

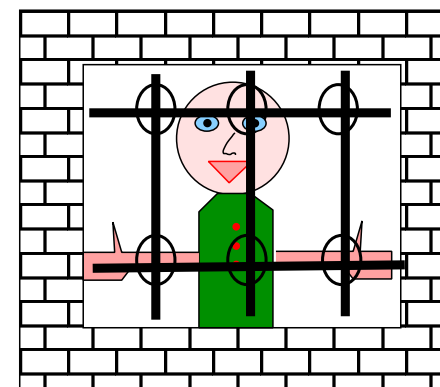
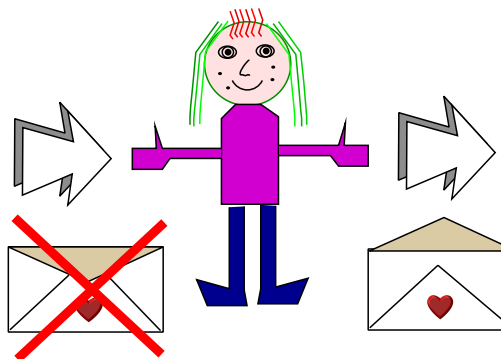
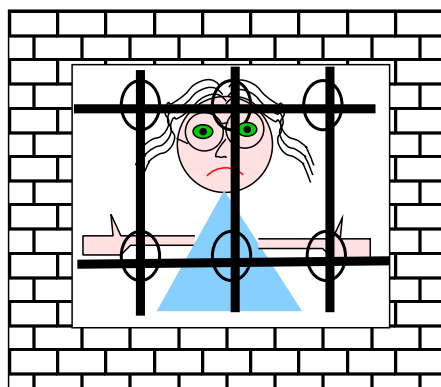
Easter is soon, dear!
So many flowers!
Can you smell them?
Are you cold at night?
Prison food stinks!
Eat well, still!
Are you lonely?
Take care!
Don't worry!
All is well!
Wendy is nice!
Need you!):

Coverttext

**Hidden text
(payload)**



ESCAPE AT DAWN



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Coverttexts



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covert texts



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Coverttexts

Simulation!



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covert texts



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Coverttexts



Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Coverttexts

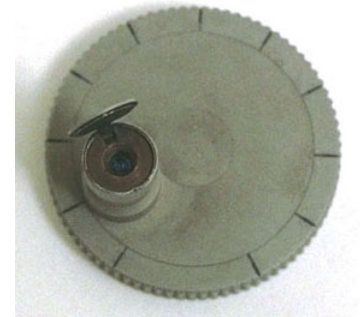


Classic Steganography



Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Coverttexts





Choice of Covertext

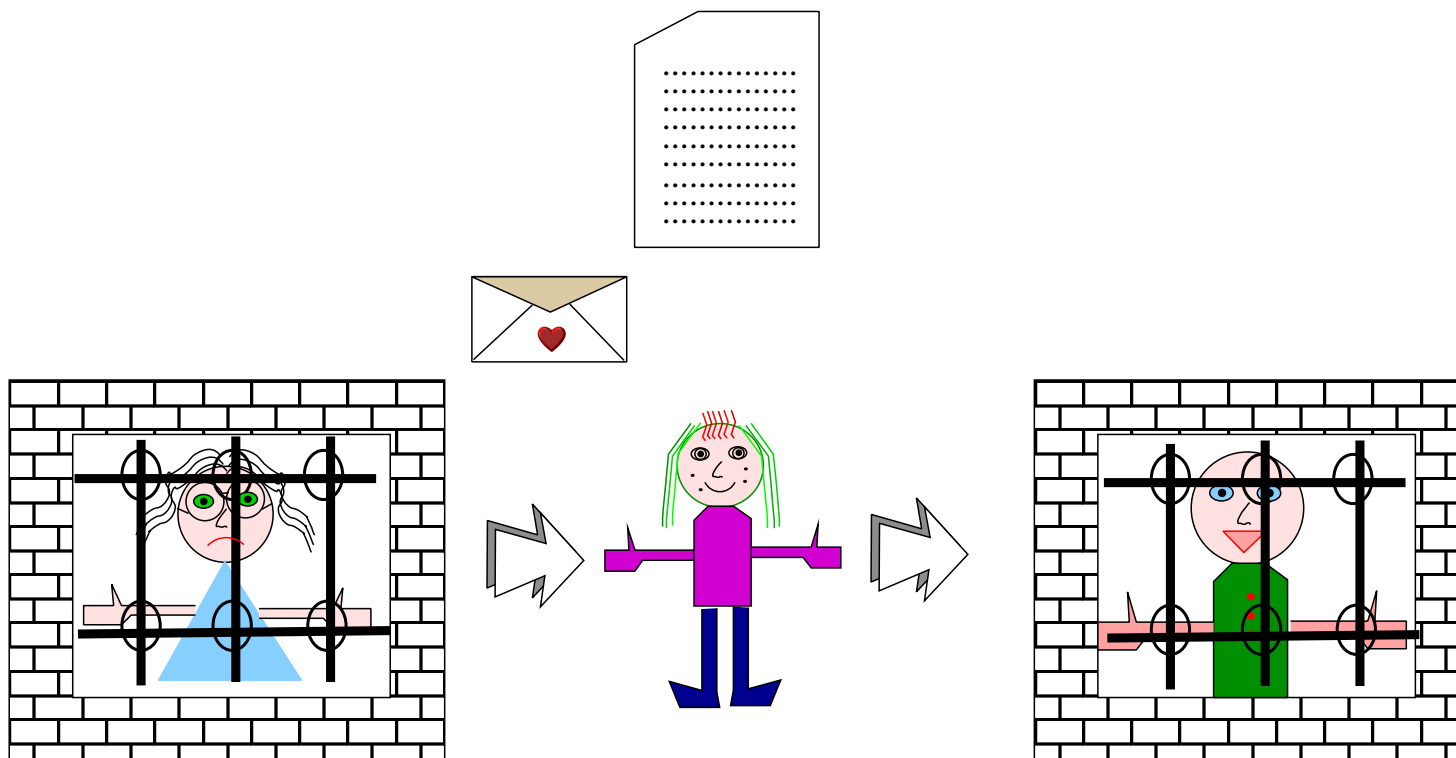
Alice, Bob, and Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





Choice of Covertext

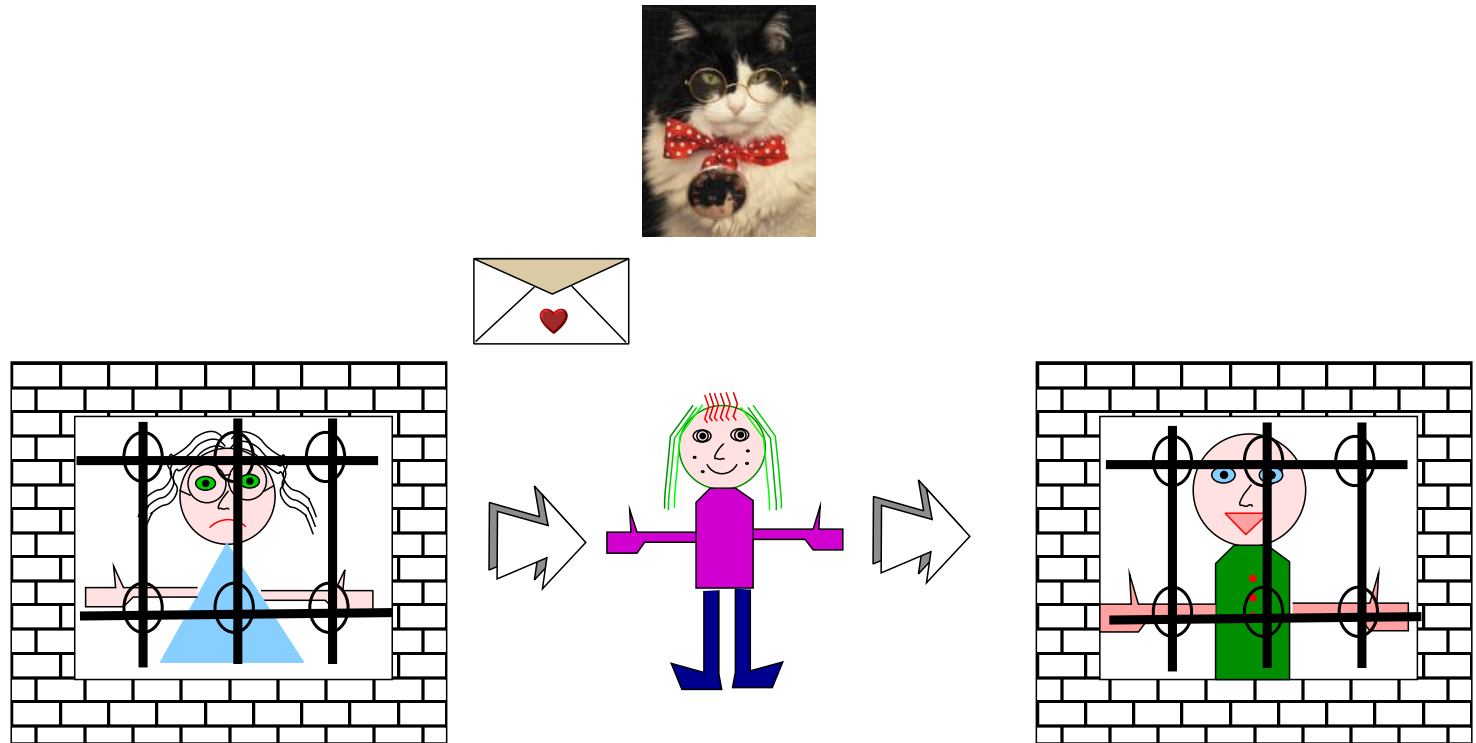
Alice, Bob, and Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





Choice of Covertext

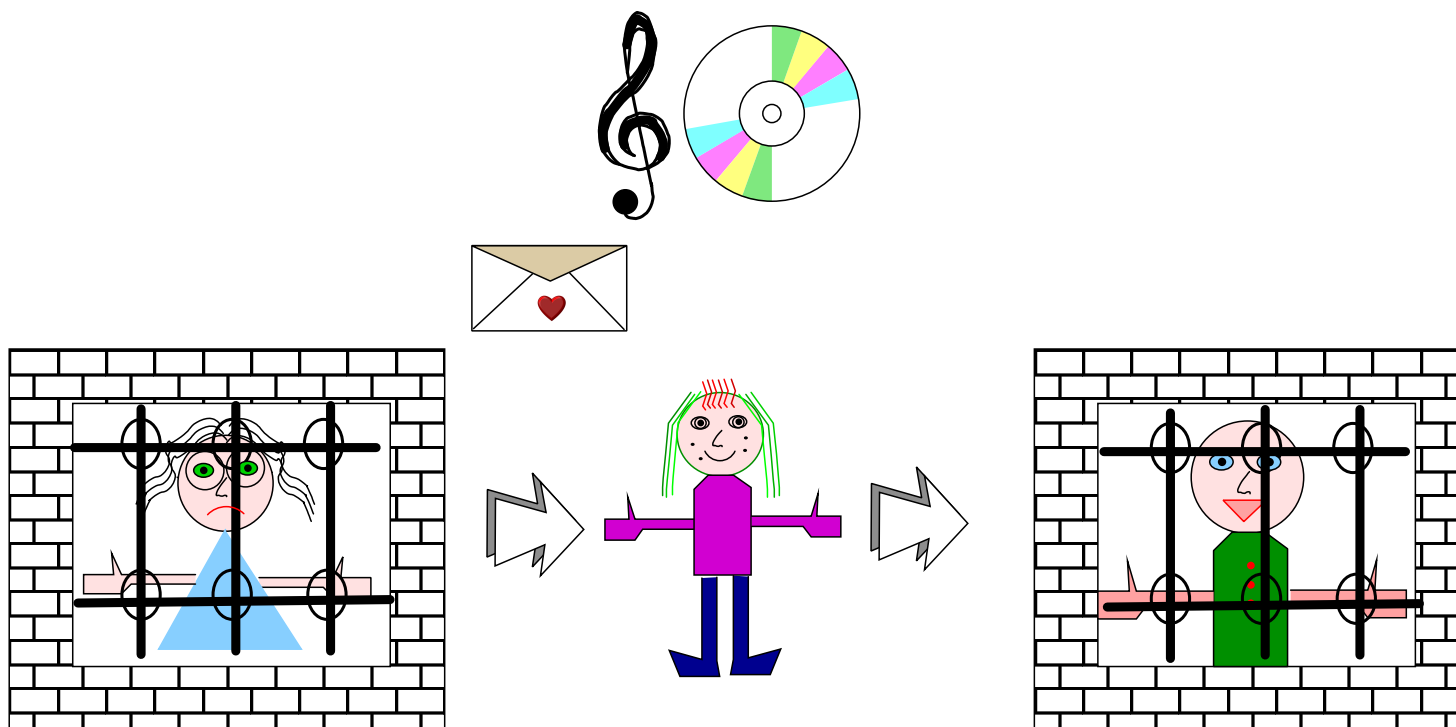
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





Choice of Covertext

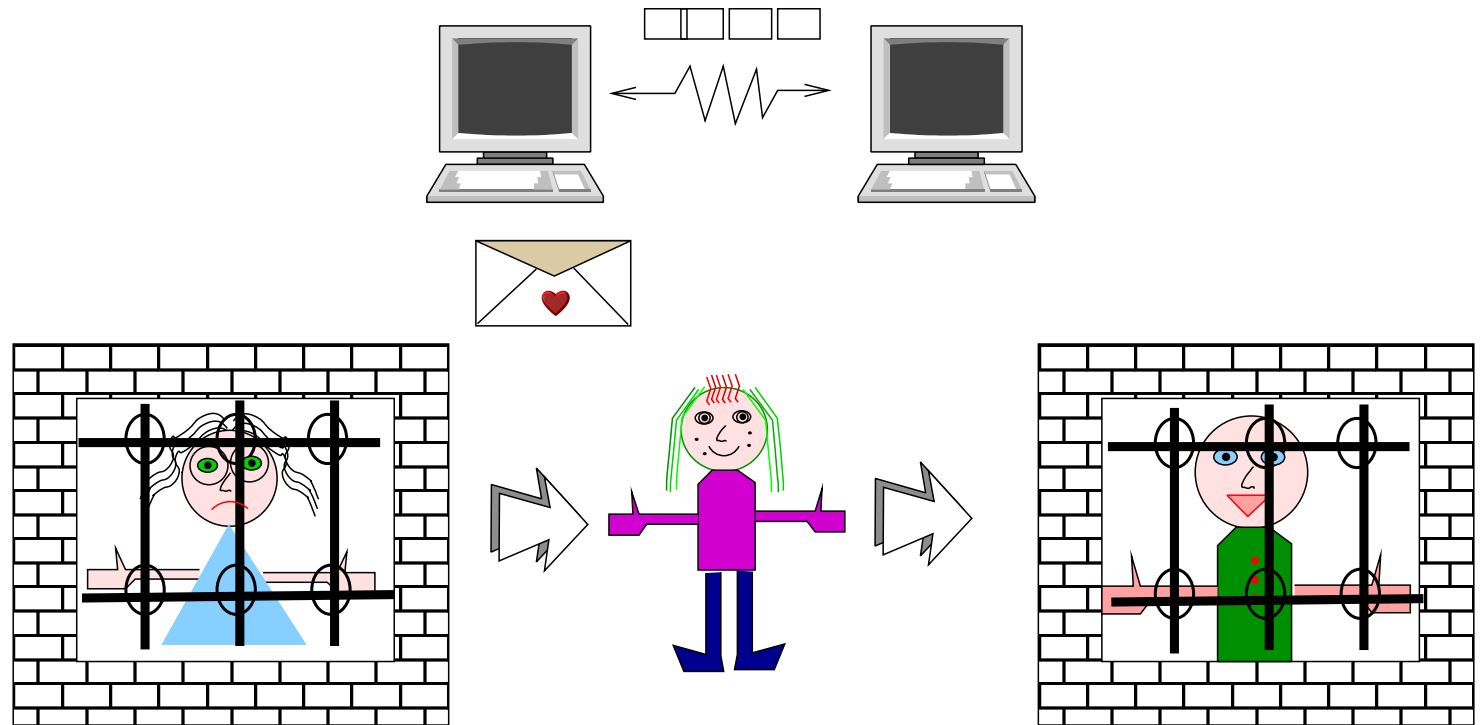
Alice, Bob, and Wendy

Prisoners' Problem

Cryptography

Steganography

Covertexts





Choice of Covertex

payload 010010 + 100101001001 covertext

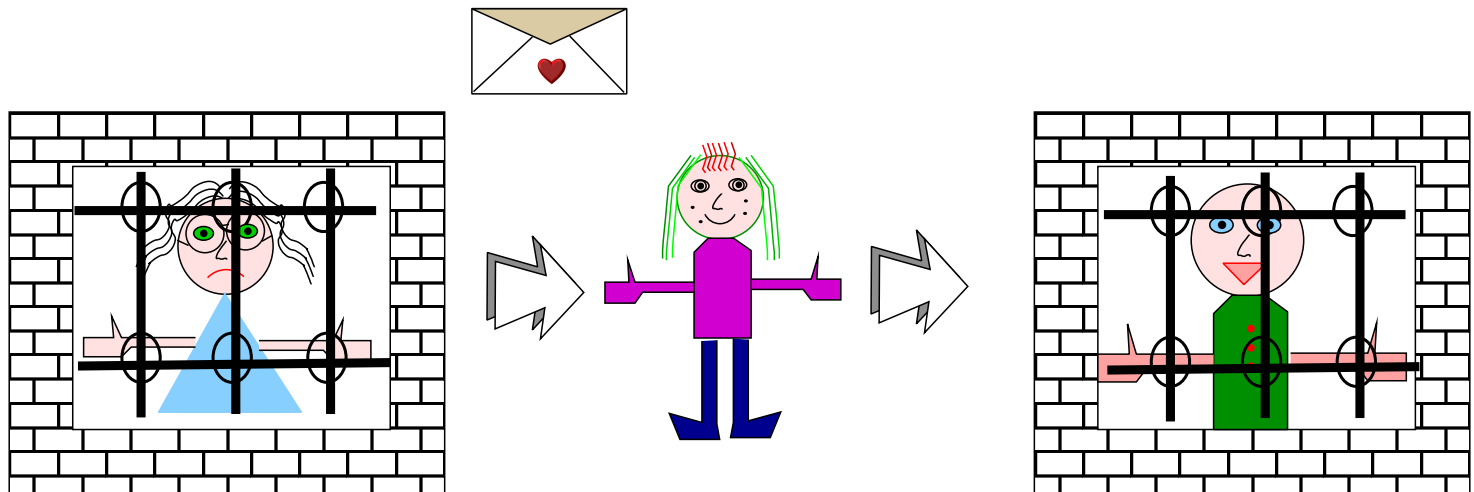
Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

Covertex



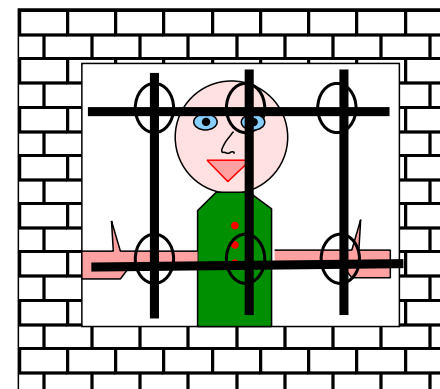
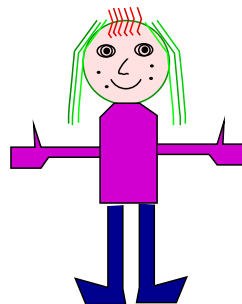
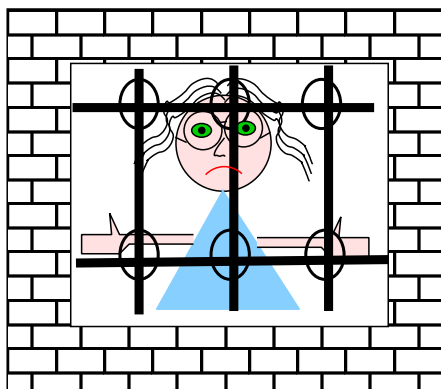


Choice of Covertext

payload 010010 + covertext 100101001001

\Downarrow

stegotext 100010010101001001



Alice, Bob, and
Wendy

Prisoners' Problem

Cryptography

Steganography

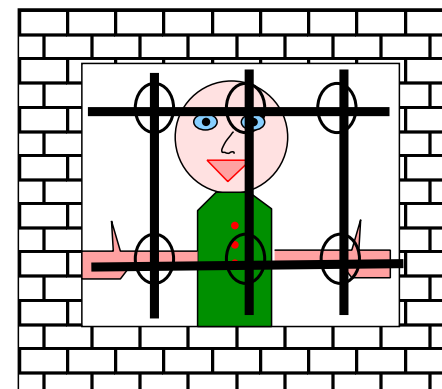
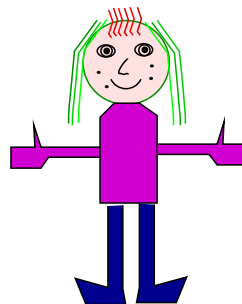
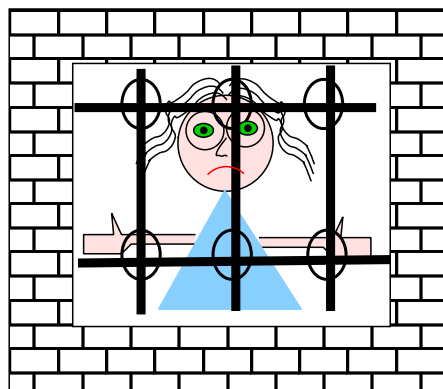
Coverttexts



Choice of Covertext

010010

↗
payload



Alice, Bob, and
Wendy

Prisoners' Problem

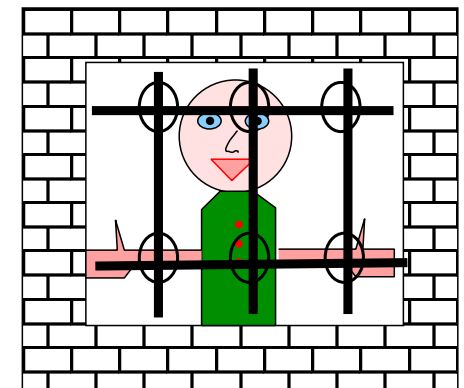
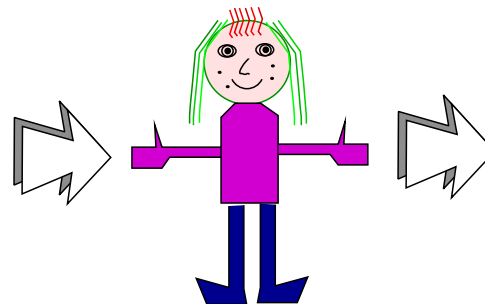
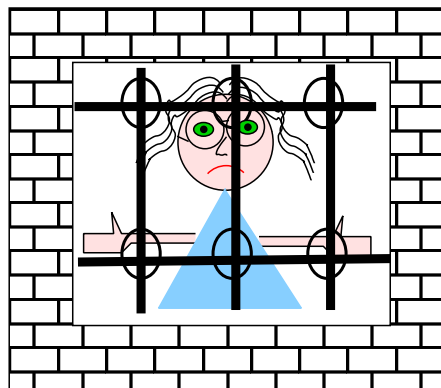
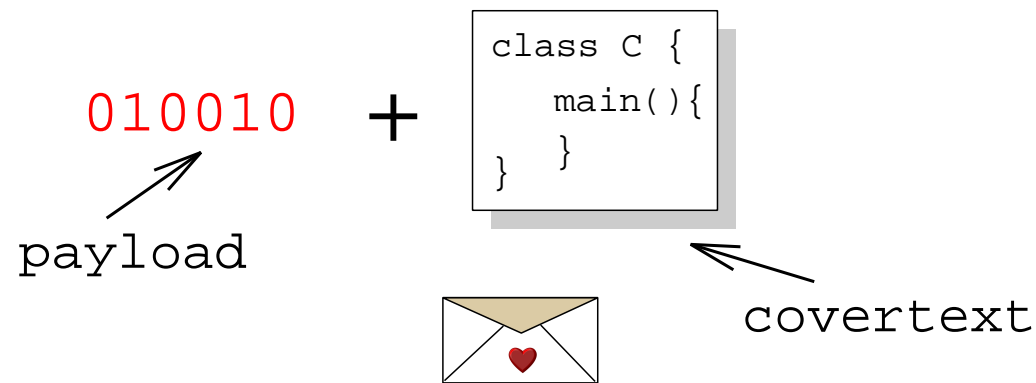
Cryptography

Steganography

Covertexts



Choice of Covertex

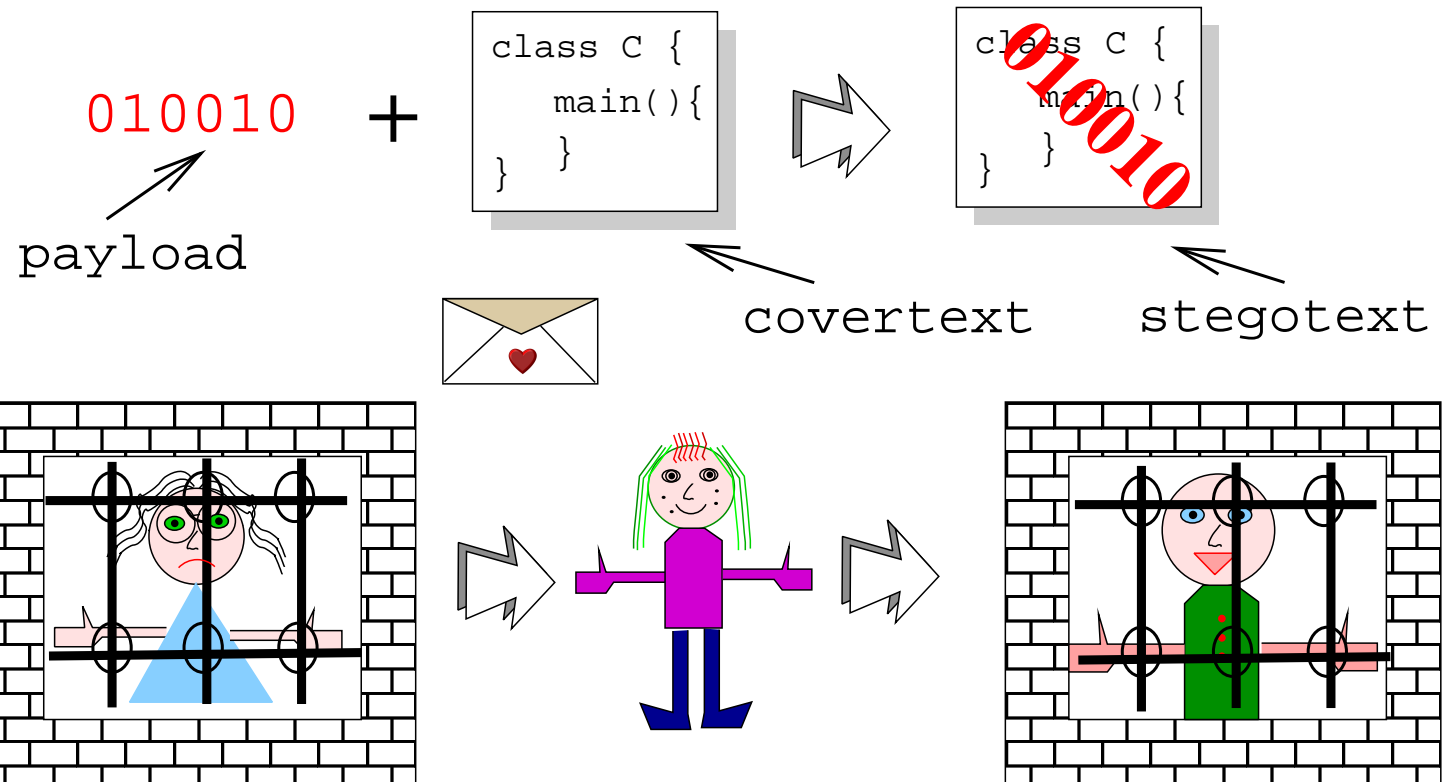


Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Covertexs



Choice of Covertext

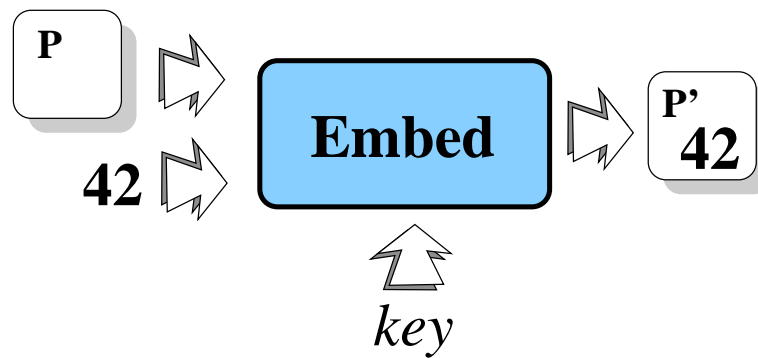


Alice, Bob, and
Wendy

Prisoners' Problem
Cryptography
Steganography
Coverttexts

Software Watermarking

Software
Watermarking

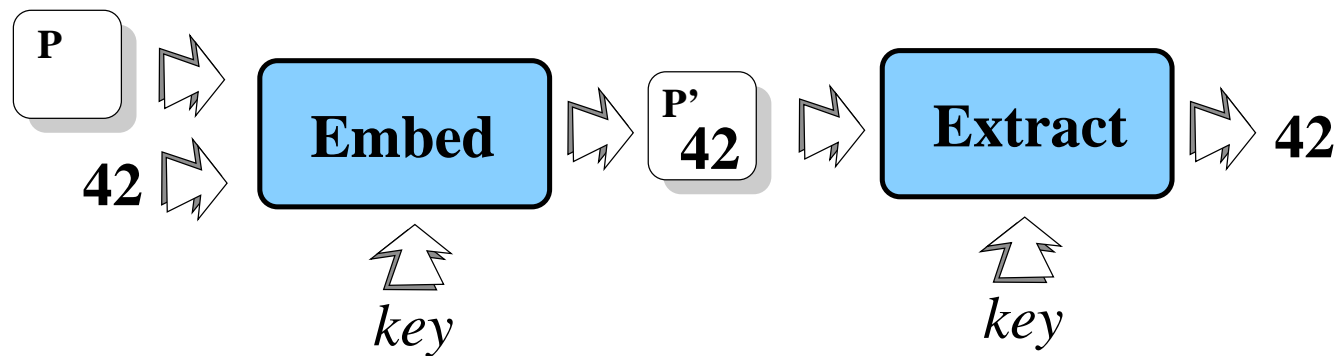


Embed an integer W in program P such that

- W is resilient against automated attacks
- W is stealthy
- W is large (high bitrate)
- the overhead (space and time) is low

Software Watermarking

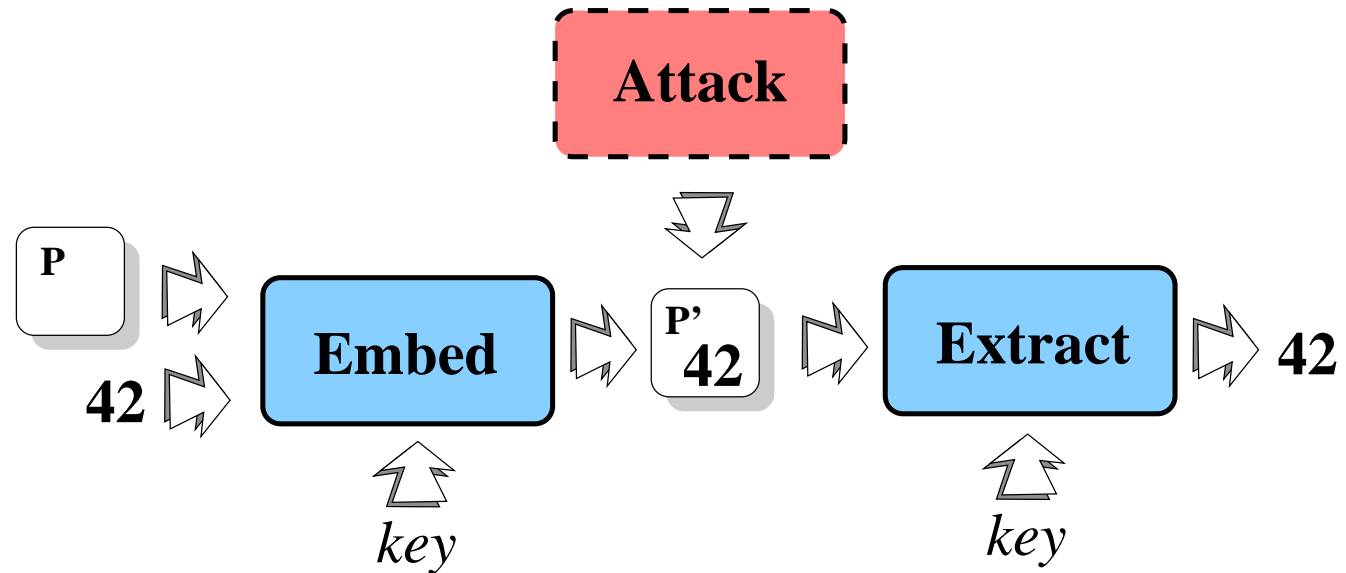
Software
Watermarking



Embed an integer W in program P such that

- W is resilient against automated attacks
- W is stealthy
- W is large (high bitrate)
- the overhead (space and time) is low

Software Watermarking



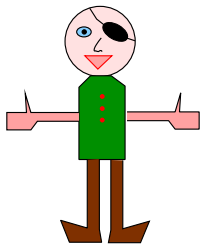
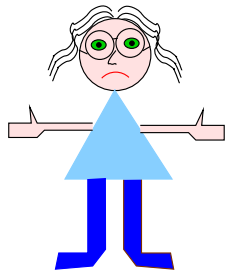
Embed an integer W in program P such that

- W is resilient against automated attacks
- W is stealthy
- W is large (high bitrate)
- the overhead (space and time) is low

Attacks on Software Watermarks

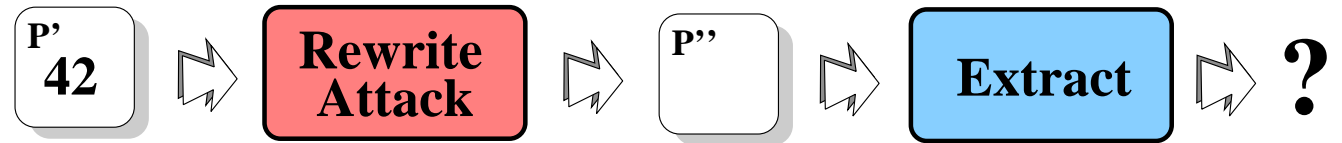


Software
Watermarking

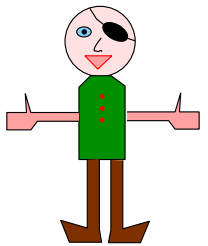
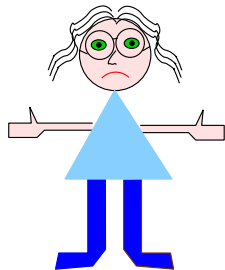




Attacks on Software Watermarks

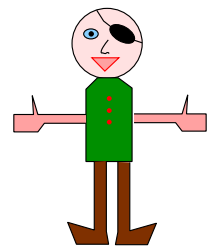
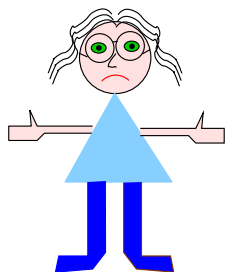
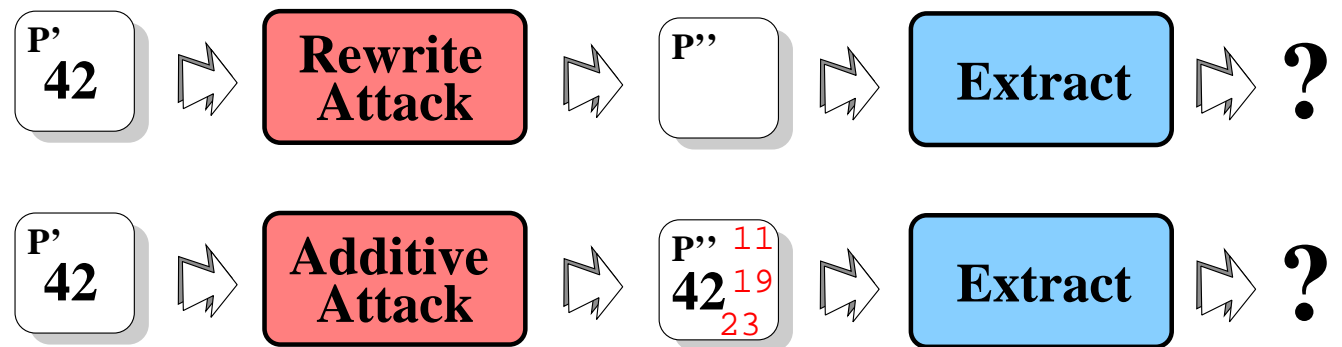


Software
Watermarking





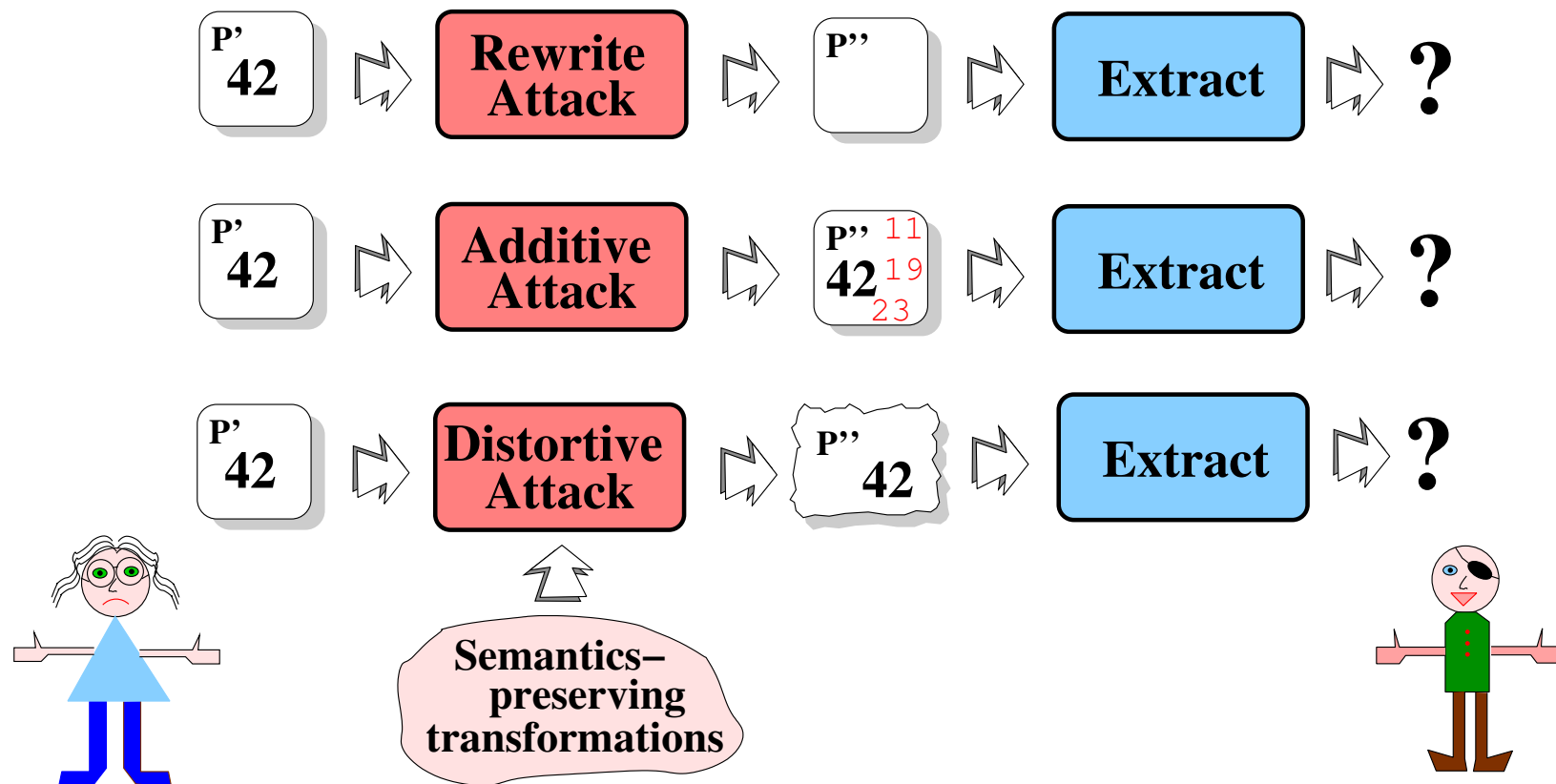
Attacks on Software Watermarks





Attacks on Software Watermarks

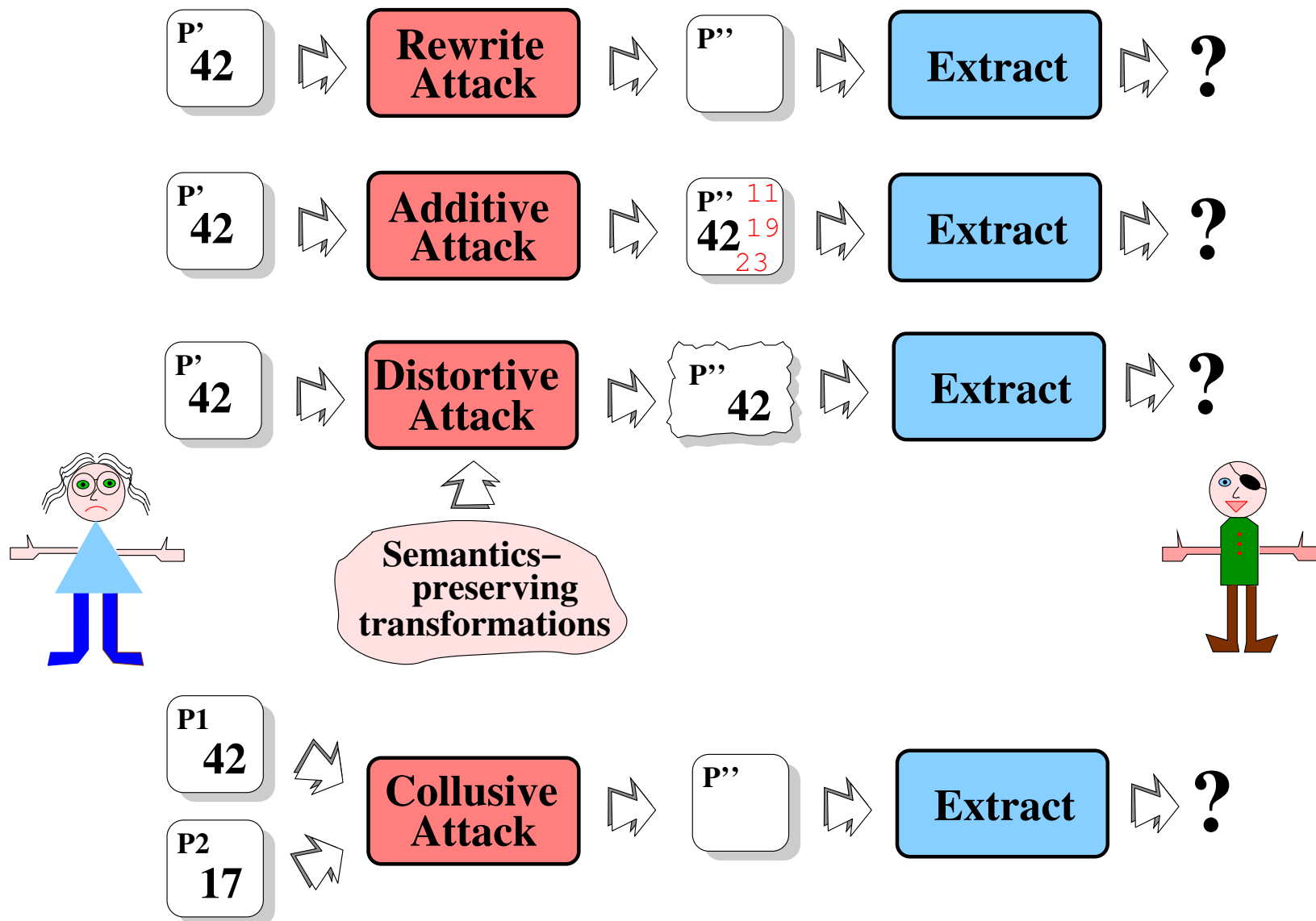
Software
Watermarking





Attacks on Software Watermarks

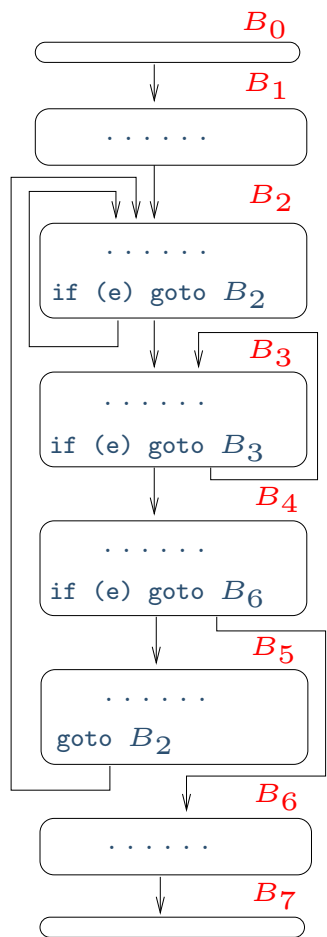
Software
Watermarking





— Davidson & Myhrvold

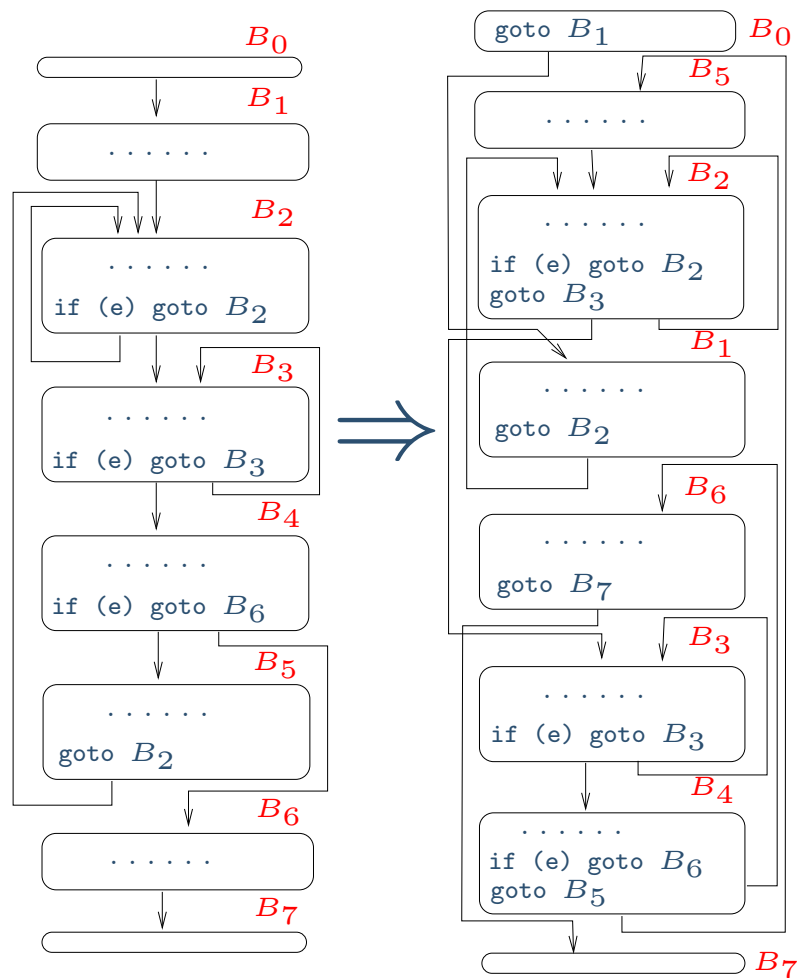
Static Watermarks





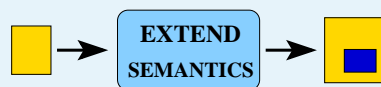
— Davidson & Myhrvold

Static Watermarks



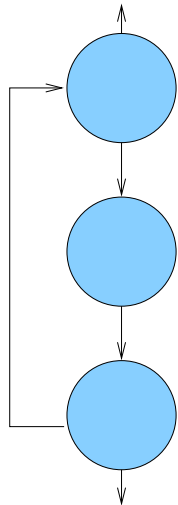
The watermark is encoded in the basic block sequence $\langle B_5, B_2, B_1, B_6, B_3, B_4 \rangle$.

US Patent 5,559,884, 1996, Microsoft

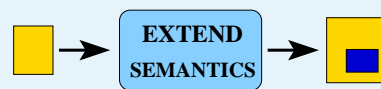


— Venkatesan et al.

Program CFG

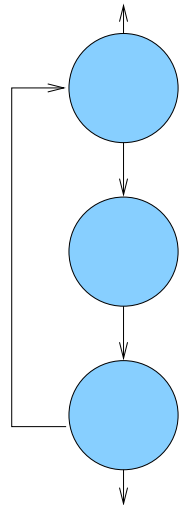


Static Watermarks

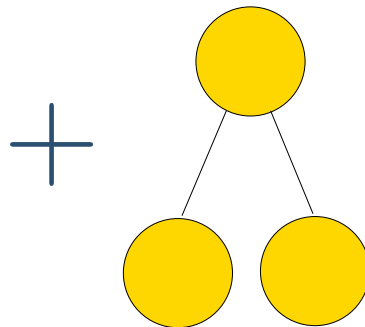


— Venkatesan et al.

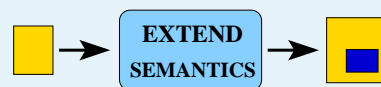
Program CFG



Watermark CFG



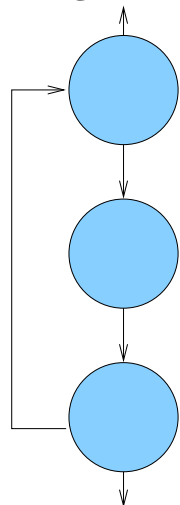
Static Watermarks



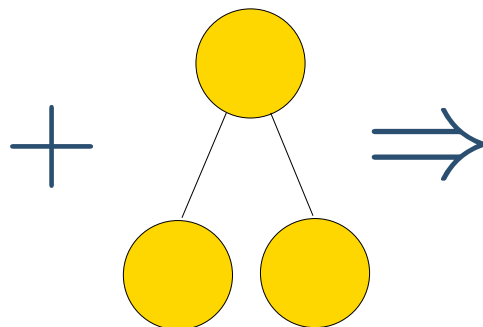
— Venkatesan et al.

Static Watermarks

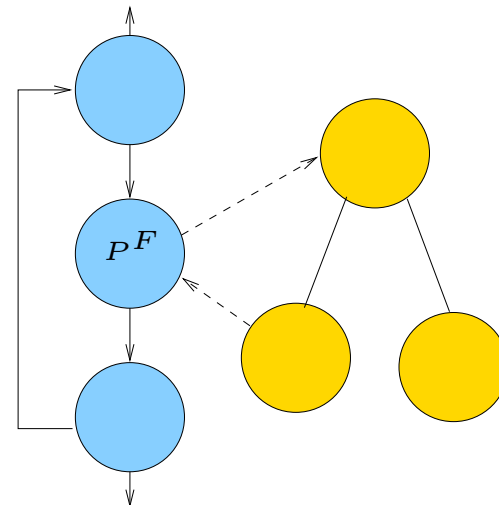
Program CFG



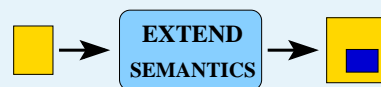
Watermark CFG



Marked Program



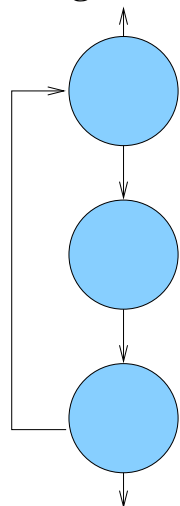
- **Bogus branches** tie the watermark CFG to the program.



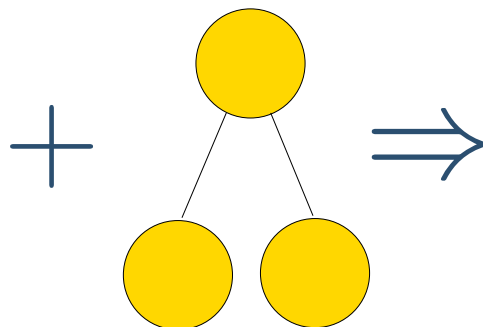
— Venkatesan et al.

Static Watermarks

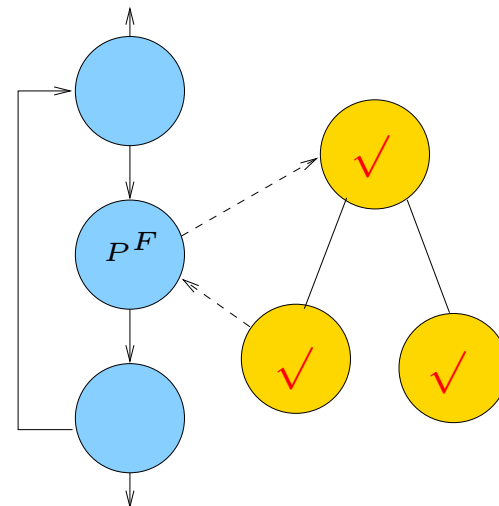
Program CFG



Watermark CFG



Marked Program

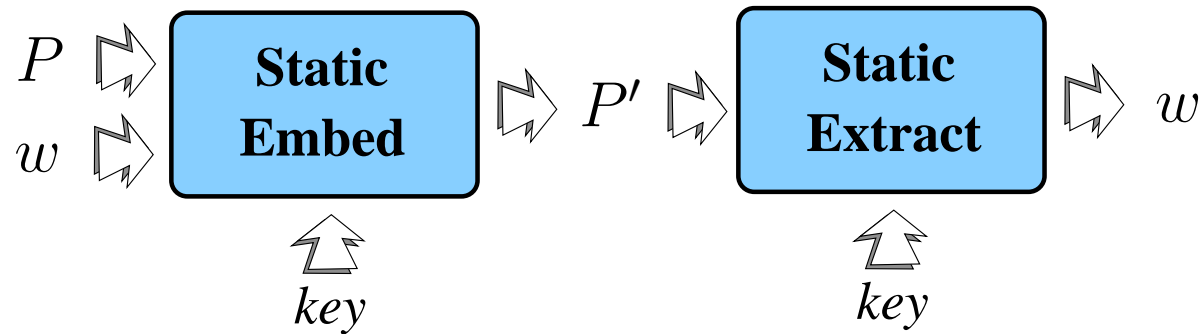


- **Bogus branches** tie the watermark CFG to the program.
- Basic blocks are **marked** so the watermark graph can be found.

Venkatesan et al., 4th Information Hiding Workshop, IHW'01

Collberg et al., 6th Information Hiding Workshop, IHW'04

Static vs. Dynamic Watermarking

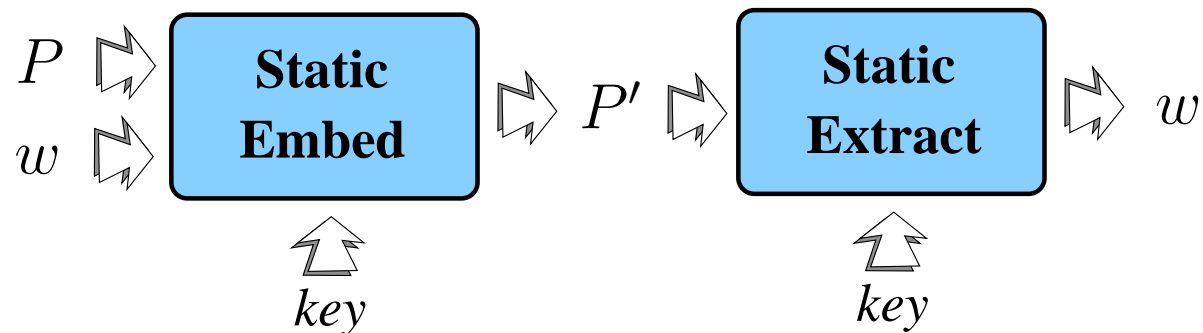


- **Static** algorithms are vulnerable to semantics-preserving code transformations.

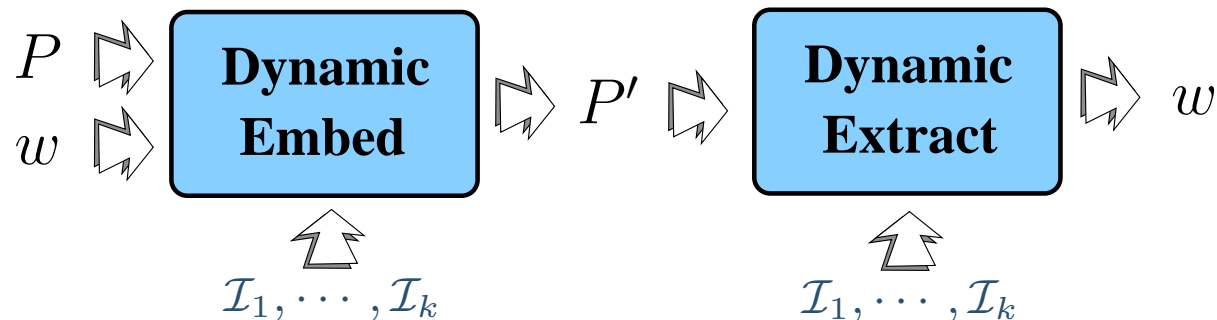
- Dynamic Watermarks
- CT
- Problems
- Increasing bit-rate
- Graph Attacks
- Graph Encoding
- Bogus fields
- Alias analysis
- Global roots
- Unstealthy nodes
- Weak cuts
- Collusion
- Problems
- PBW
- NT



Static vs. Dynamic Watermarking



- **Static** algorithms are vulnerable to semantics-preserving code transformations.



- **Dynamic** algorithms extract the mark from the state of the program when run on a secret key input sequence.

Dynamic Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

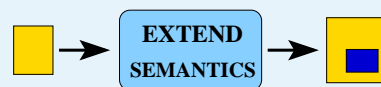
Weak cuts

Collusion

Problems

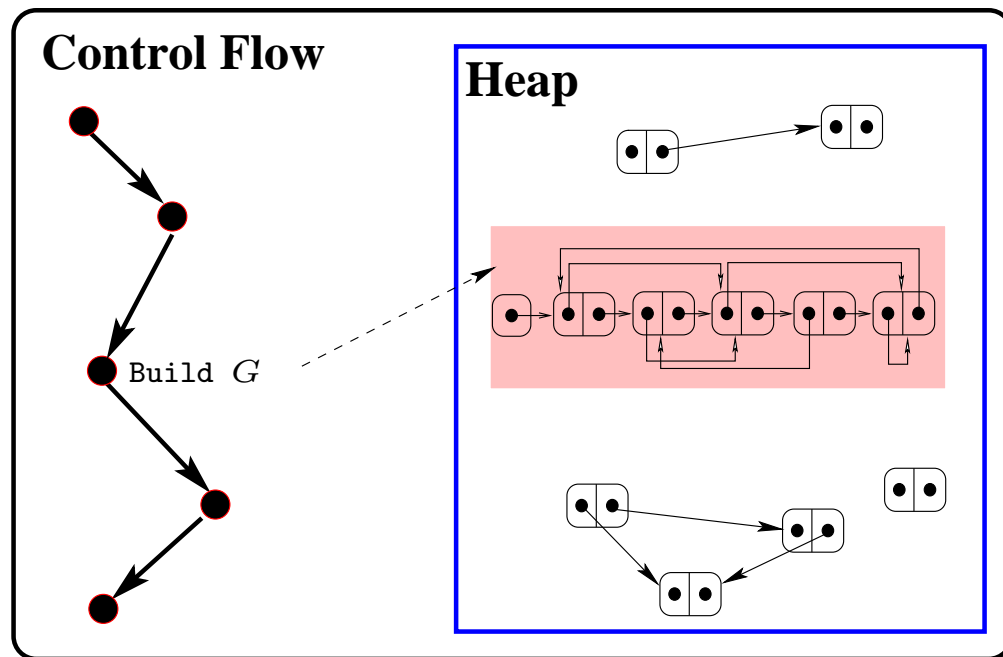
PBW

NT



— Graph-Based Watermark

$\mathcal{I}_1, \dots, \mathcal{I}_k$



$\Rightarrow n$

- The watermark is embedded in the topology of a dynamic graph structure, built at runtime but only for the special input sequence $\mathcal{I}_1, \dots, \mathcal{I}_k$.
- Why? **Shape-analysis** is hard.

Collberg & Thomborson, ACM POPL'99

Dynamic Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example

```
public class Main {  
    public static void main(String args[]){  
        System.out.println("Hello_" + args[0]);  
    }  
}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT

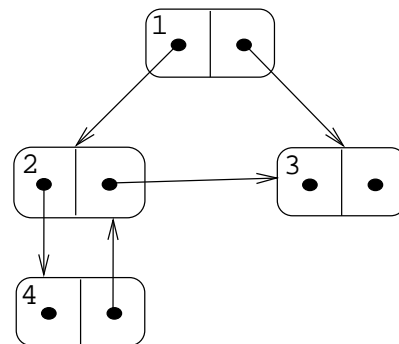


CT — Example

```
public class Main {  
    public static void main(String args[]){  
        System.out.println("Hello_" + args[0]);  
    }  
}
```



$W = 42 \rightarrow$



Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example...

```
public class Node {public Node left , right ;}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example...

```
public class Node {public Node left , right ;}
```

+

```
public class Main {  
  
    public static void main(String args []) {  
        System.out.println(" Hello_" + args [0]);  
  
    }  
}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example...

```
public class Node {public Node left , right ;}
```

+

```
public class Main {  
  
    public static void main(String args[]){  
        System.out.println("Hello_" + args[0]);  
  
        Node n4=new Node();  
        Node n2=new Node();  
        n2.left=n4; n4.right=n2;  
        Node n3=new Node();  
        n2.right=n3;  
        Node n1=new Node();  
        n1.left=n2; n1.right=n3;  
  
    }  
}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example...

```
public class Node {public Node left , right ;}
```

+

```
public class Main {  
  
    public static void main(String args[]){  
        System.out.println("Hello_" + args[0]);  
        if (i.equals("World")) {  
            Node n4=new Node();  
            Node n2=new Node();  
            n2.left=n4; n4.right=n2;  
            Node n3=new Node();  
            n2.right=n3;  
            Node n1=new Node();  
            n1.left=n2; n1.right=n3;  
        }  
    }  
}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



CT — Example...

```
public class Node {public Node left , right ;}
```

+

```
public class Main {  
    public static Node root;  
    public static void main( String args [] ){  
        System.out.println( " Hello _ " + args [0] );  
        if ( i.equals( " World " ) ) {  
            Node n4=new Node();  
            Node n2=new Node();  
            n2.left=n4; n4.right=n2;  
            Node n3=new Node();  
            n2.right=n3;  
            Node n1=new Node(); root=n1;  
            n1.left=n2; n1.right=n3;  
        }  
    }  
}
```

Dynamic
Watermarks

CT

Problems
Increasing bit-rate
Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?
5. global variables?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?
5. global variables?
6. introducing extra unstealthy classes?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?
5. global variables?
6. introducing extra unstealthy classes?
7. weak cuts?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?
5. global variables?
6. introducing extra unstealthy classes?
7. weak cuts?
8. static collusive attacks?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems

Algorithm looks easy enough on paper, but there are lots of problems!

How do we avoid

1. huge (unstealthy) graphs?
2. attacks by small graph perturbations?
3. bogus field addition?
4. attacks by advanced alias analysis?
5. global variables?
6. introducing extra unstealthy classes?
7. weak cuts?
8. static collusive attacks?
9. dynamic collusive attacks?

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

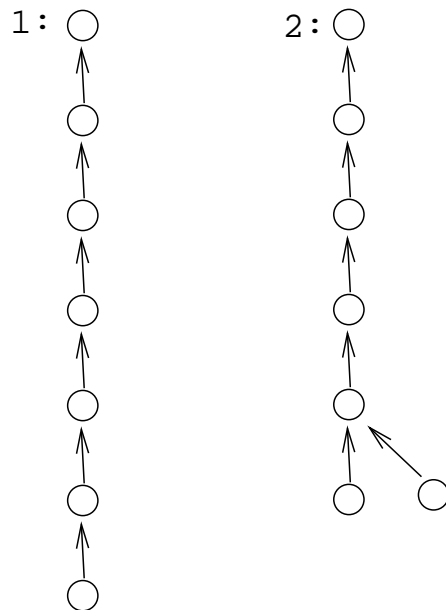
NT



- n is represented by the *index* of the the graph G in some enumeration.
- We must, efficiently, be able to
 1. given n , generate the n :th graph,
 2. given G , find G 's index n .
- Oriented parent-pointer trees \Rightarrow bit-rate: 1.56 bits per word.



Avoiding huge graphs — Enumeration Encoding



- n is represented by the *index* of the the graph G in some enumeration.
- We must, efficiently, be able to
 1. given n , generate the n :th graph,
 2. given G , find G 's index n .
- Oriented parent-pointer trees \Rightarrow bit-rate: 1.56 bits per word.

Dynamic
Watermarks

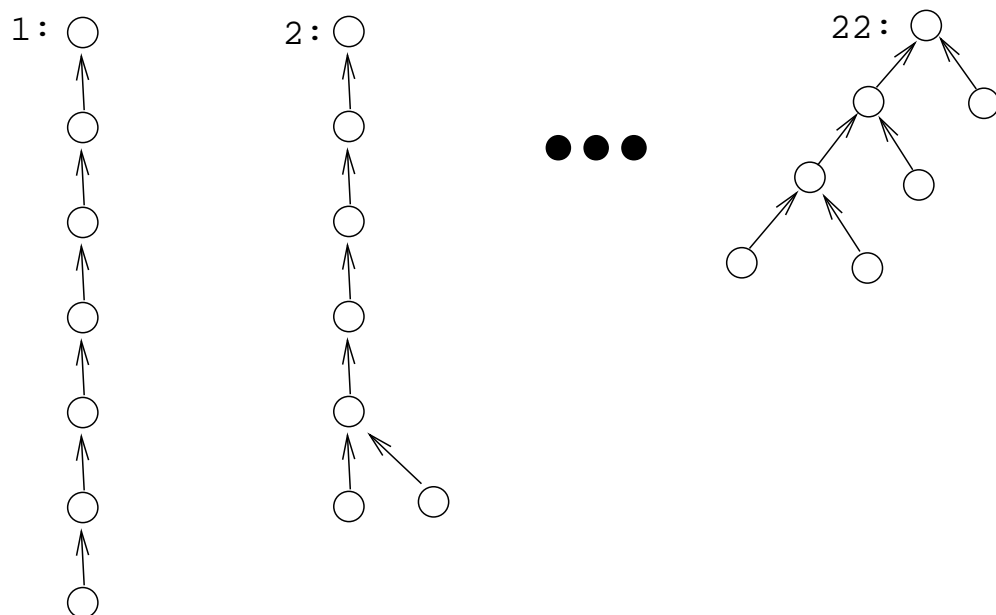
CT
Problems

Increasing bit-rate

Graph Attacks
Graph Encoding
Bogus fields
Alias analysis
Global roots
Unstealthy nodes
Weak cuts
Collusion
Problems
PBW
NT



Avoiding huge graphs — Enumeration Encoding



- n is represented by the *index* of the the graph G in some enumeration.
- We must, efficiently, be able to
 1. given n , generate the n :th graph,
 2. given G , find G 's index n .
- Oriented parent-pointer trees \Rightarrow bit-rate: 1.56 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

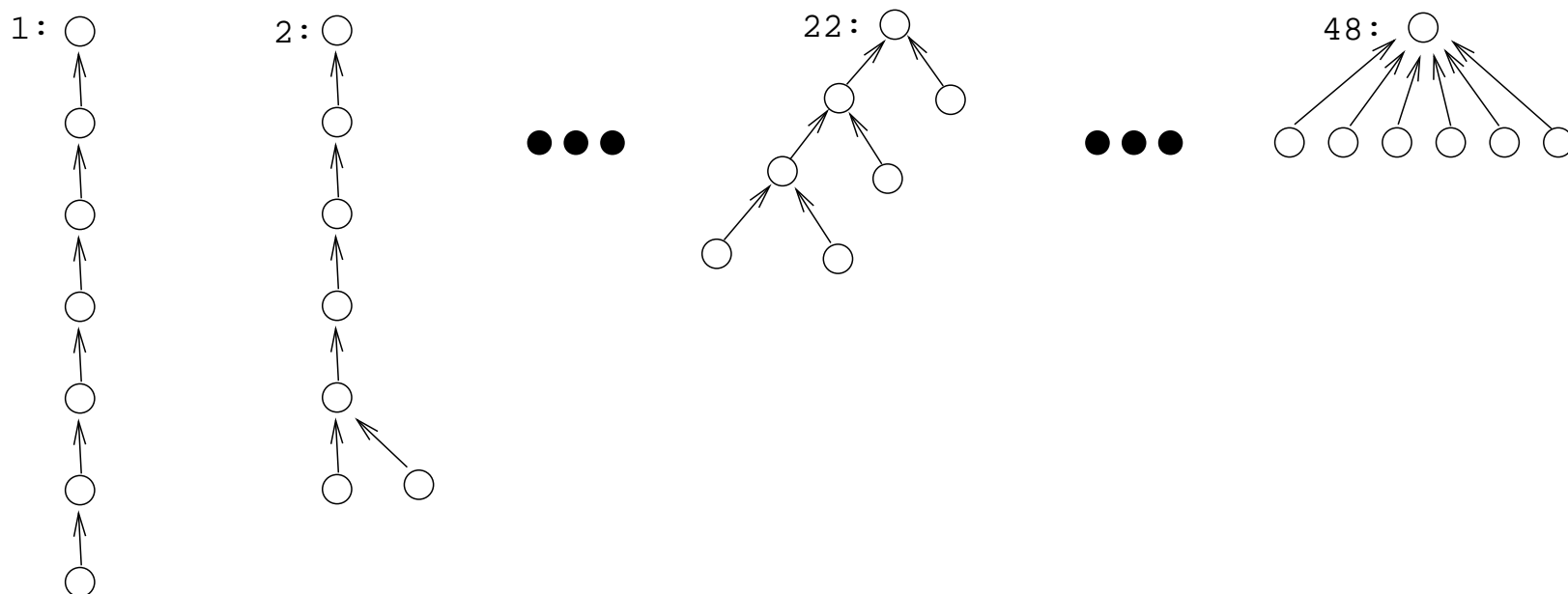
Problems

PBW

NT



Avoiding huge graphs — Enumeration Encoding



- n is represented by the *index* of the the graph G in some enumeration.
- We must, efficiently, be able to
 1. given n , generate the n :th graph,
 2. given G , find G 's index n .
- Oriented parent-pointer trees \Rightarrow bit-rate: 1.56 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

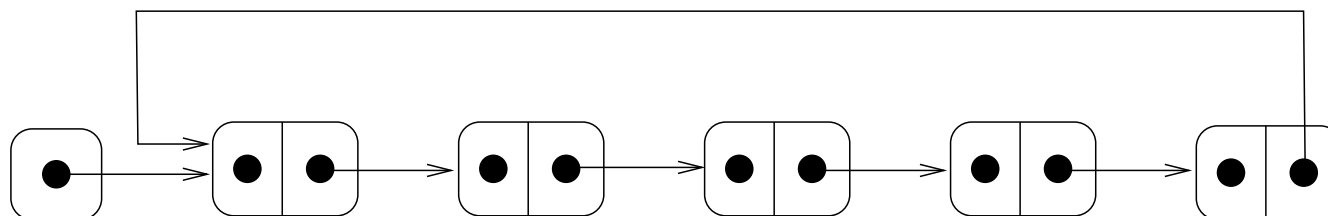
Problems

PBW

NT



Avoiding huge graphs — Radix- k Encoding



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT

$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

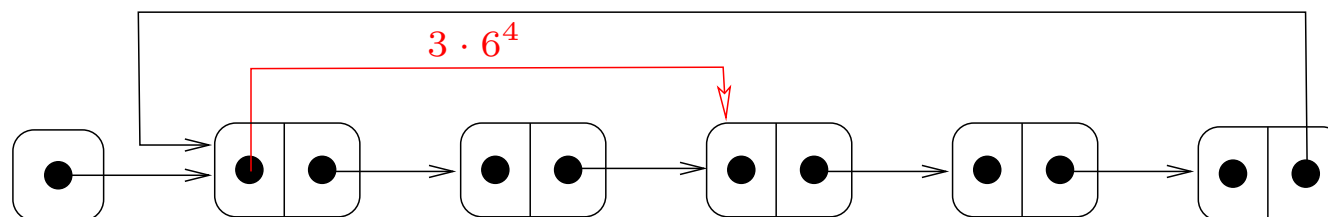
- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|-------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 ... |

- Bit-rate is 4 bits per word.



Avoiding huge graphs — Radix- k Encoding



$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|-------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 ... |

- Bit-rate is 4 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

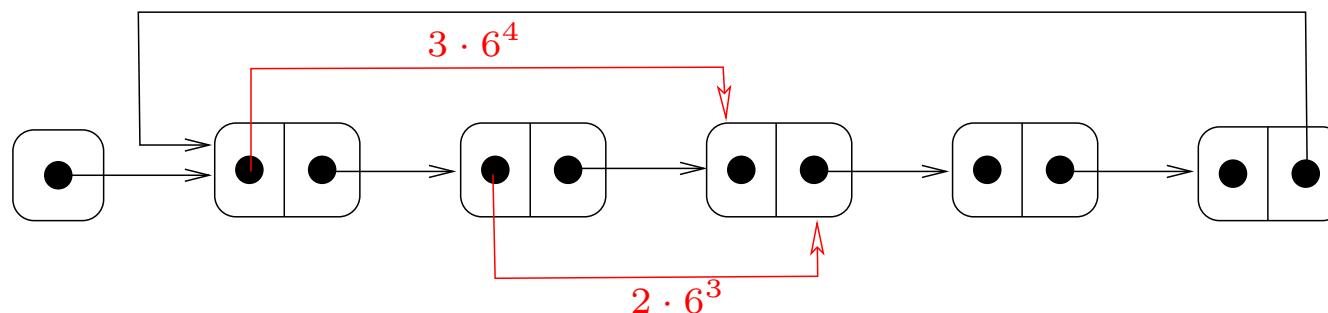
Problems

PBW

NT



Avoiding huge graphs — Radix- k Encoding



$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|-------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 ... |

- Bit-rate is 4 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

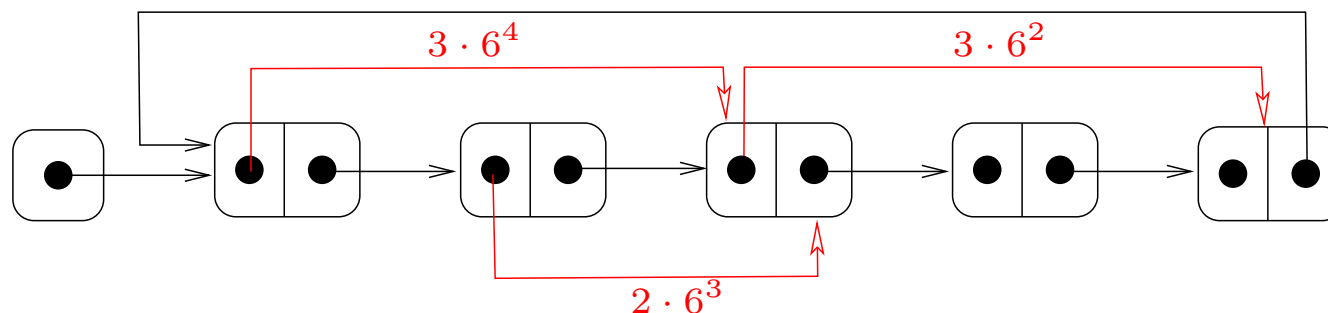
Problems

PBW

NT



Avoiding huge graphs — Radix- k Encoding



$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|---------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 \dots |

- Bit-rate is 4 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

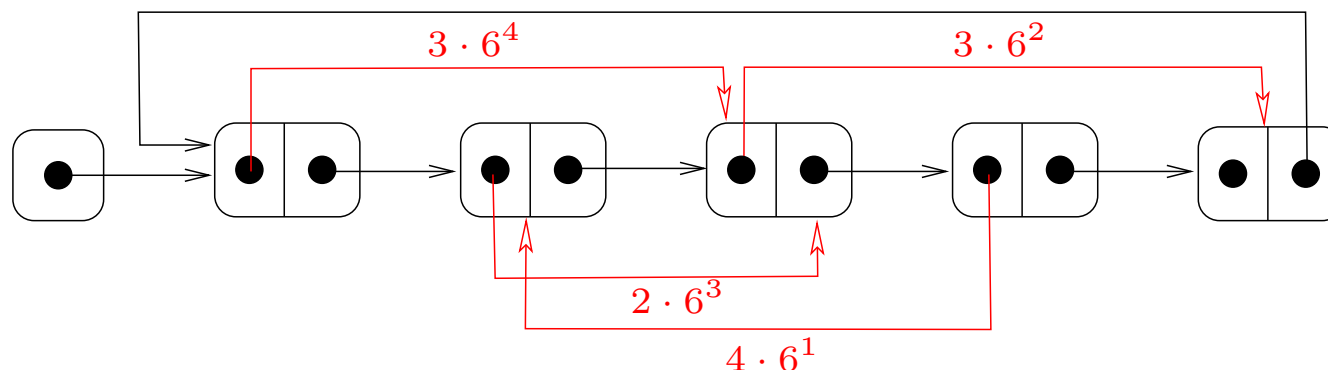
Problems

PBW

NT



Avoiding huge graphs — Radix- k Encoding



$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|-------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 ... |

- Bit-rate is 4 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

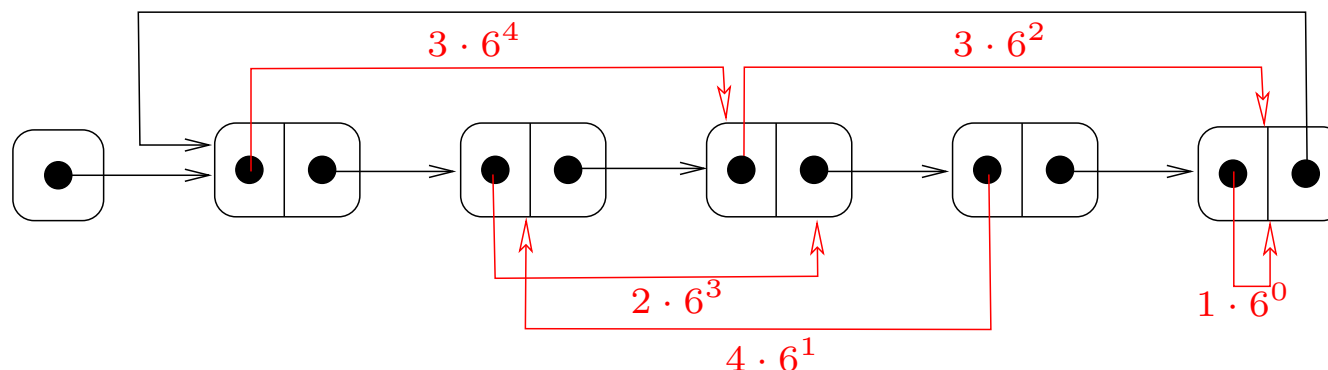
Problems

PBW

NT



Avoiding huge graphs — Radix- k Encoding



$$3 \cdot 6^4 + 2 \cdot 6^3 + 3 \cdot 6^2 + 4 \cdot 6^1 + 1 \cdot 6^0 = 4453$$

- G is a circular linked list where an extra pointer field encodes a base- k digit:

| | | |
|-------------------|---------------|-------|
| null-pointer | \Rightarrow | 0 |
| self-pointer | \Rightarrow | 1 |
| next node pointer | \Rightarrow | 2 ... |

- Bit-rate is 4 bits per word.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

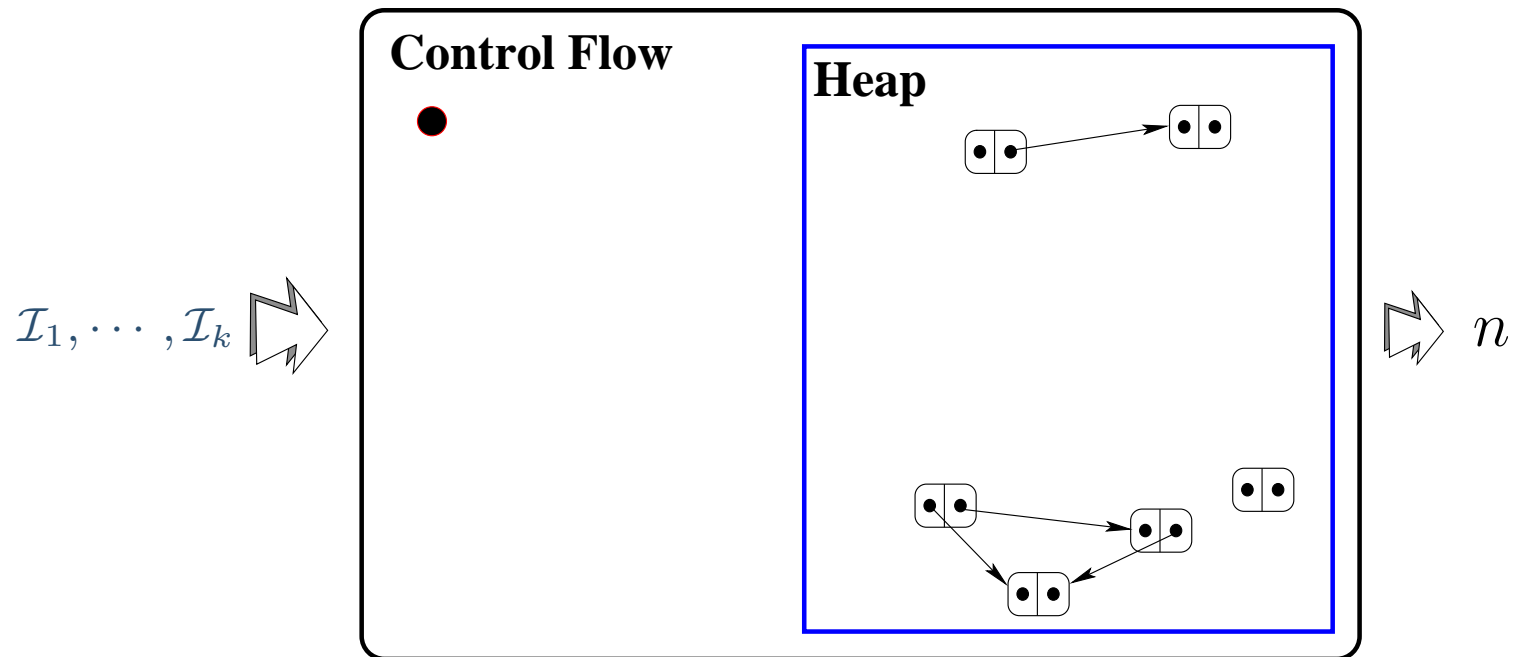
Collusion

Problems

PBW

NT

Avoiding huge graphs — Splitting the graph



- Build the graph in pieces \Rightarrow increase the watermark size without decreasing stealth.



Avoiding huge graphs — Splitting the graph

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

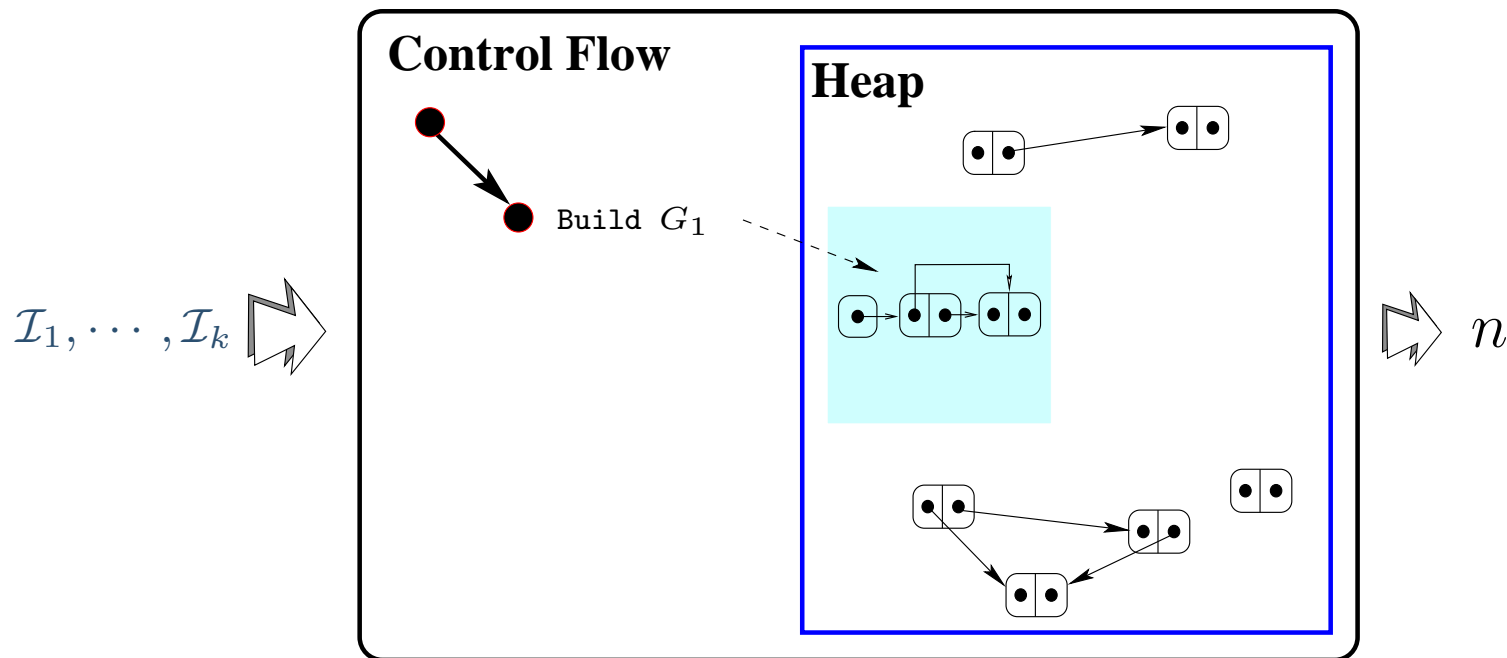
Weak cuts

Collusion

Problems

PBW

NT



- Build the graph in pieces \Rightarrow increase the watermark size without decreasing stealth.



Avoiding huge graphs — Splitting the graph

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

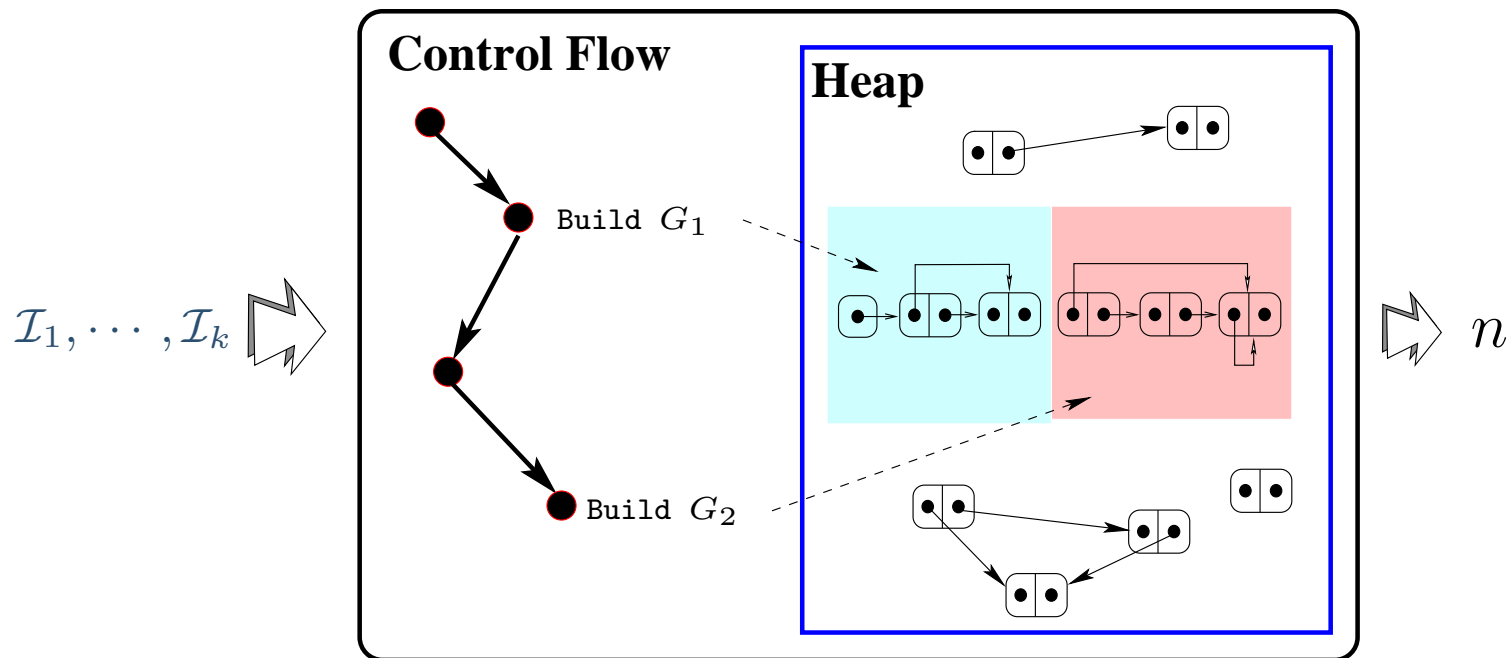
Weak cuts

Collusion

Problems

PBW

NT



- Build the graph in pieces \Rightarrow increase the watermark size without decreasing stealth.



Avoiding huge graphs — Splitting the graph

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

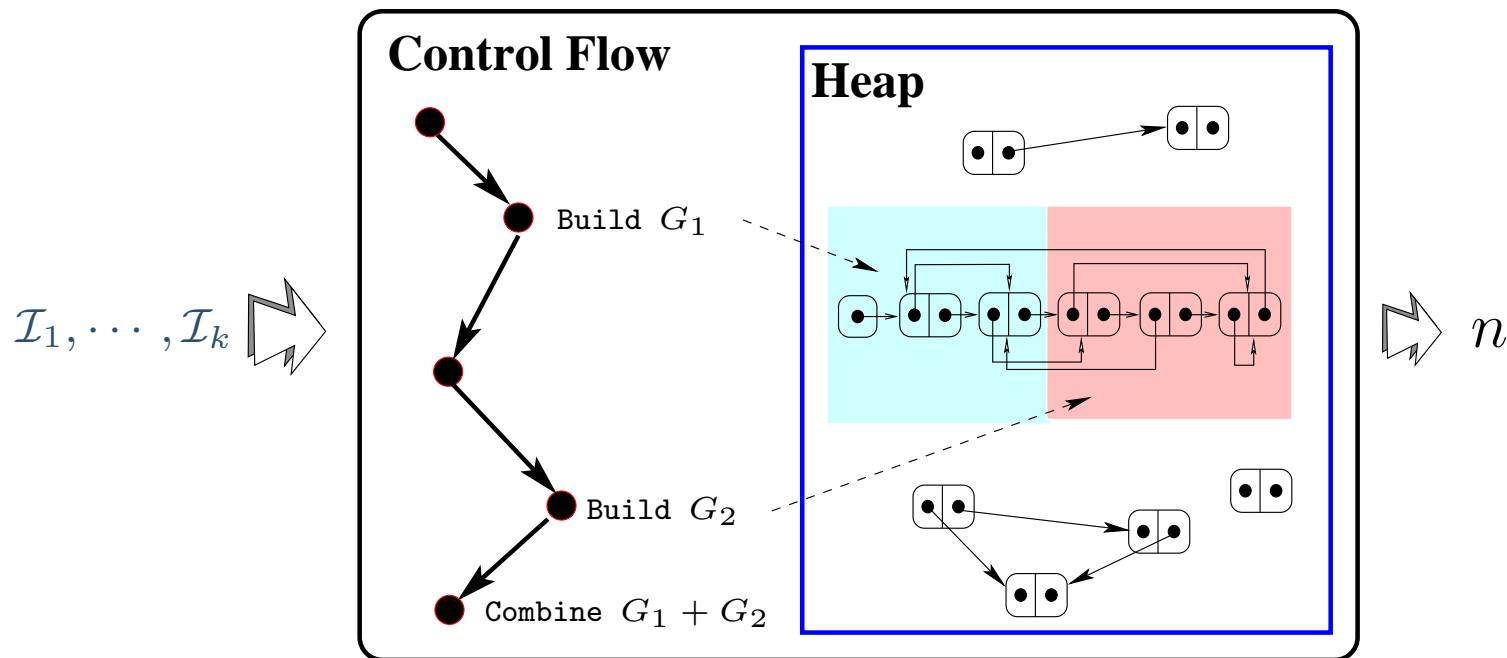
Weak cuts

Collusion

Problems

PBW

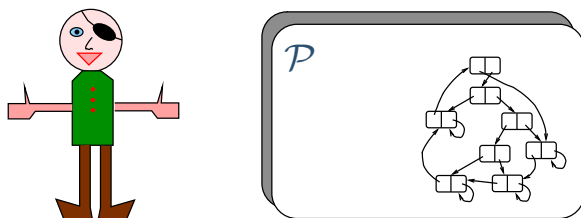
NT



- Build the graph in pieces \Rightarrow increase the watermark size without decreasing stealth.
- More pieces \Rightarrow increase the code necessary to combine pieces!



Avoiding attacks by small graph perturbations



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

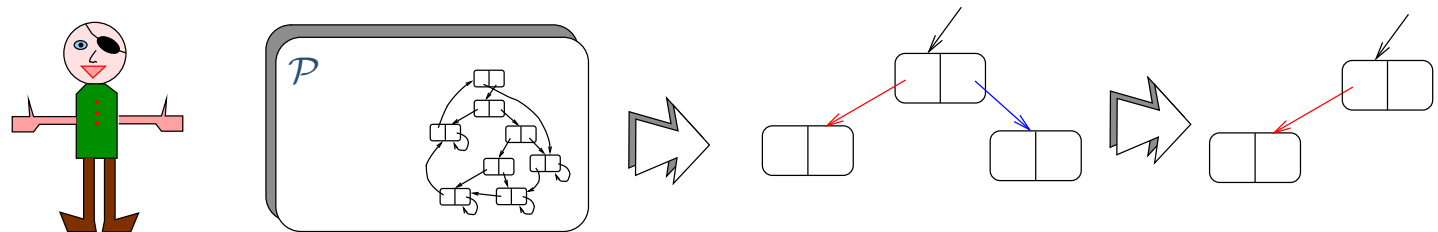
Problems

PBW

NT



Avoiding attacks by small graph perturbations



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

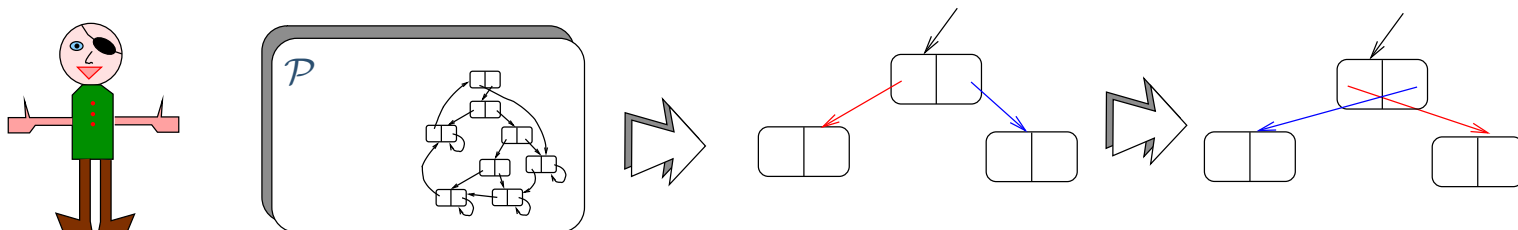
PBW

NT

■ Node deletion



Avoiding attacks by small graph perturbations



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

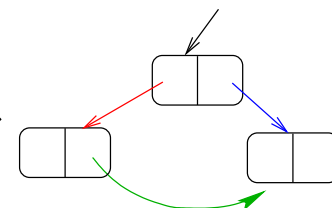
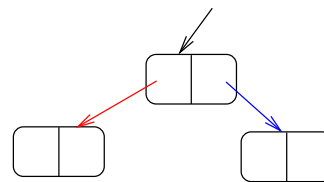
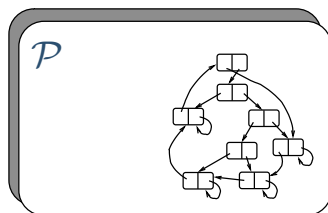
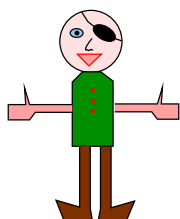
PBW

NT

- Node deletion
- Edge flip



Avoiding attacks by small graph perturbations



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

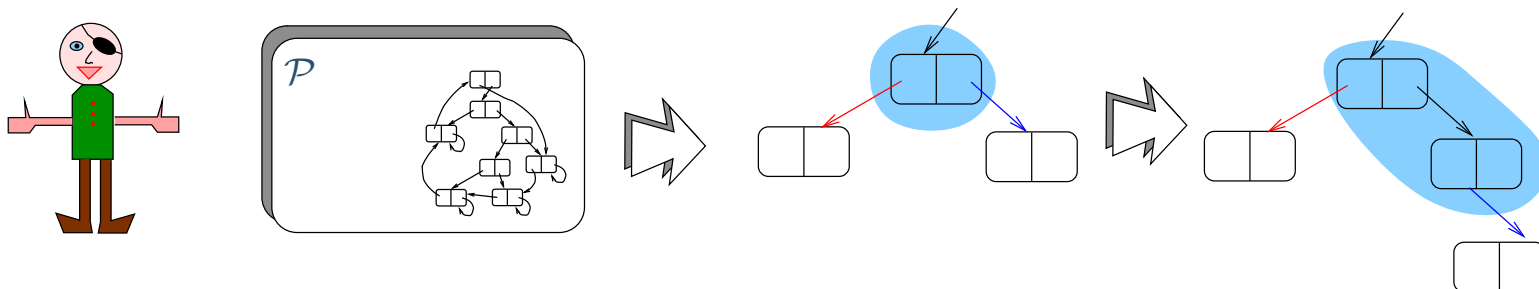
PBW

NT

- Node deletion
- Edge flip
- Edge addition



Avoiding attacks by small graph perturbations



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

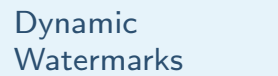
Collusion

Problems

PBW

NT

- Node deletion
- Edge flip
- Edge addition
- Node split

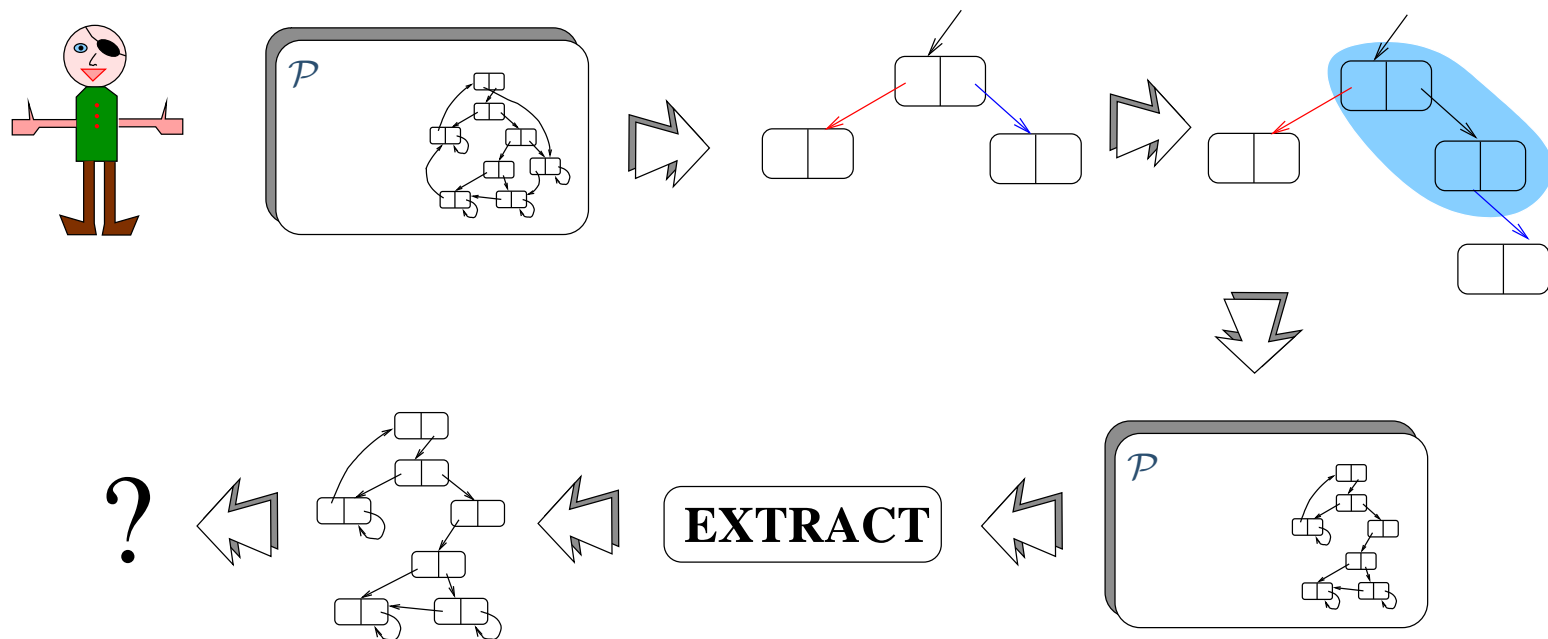


NT

- ## Software Watermarking



Avoiding attacks by small graph perturbations



?

EXTRACT

- Node deletion
- Edge flip
- Edge addition
- Node split

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Error-Correcting Graphs — PPCT

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

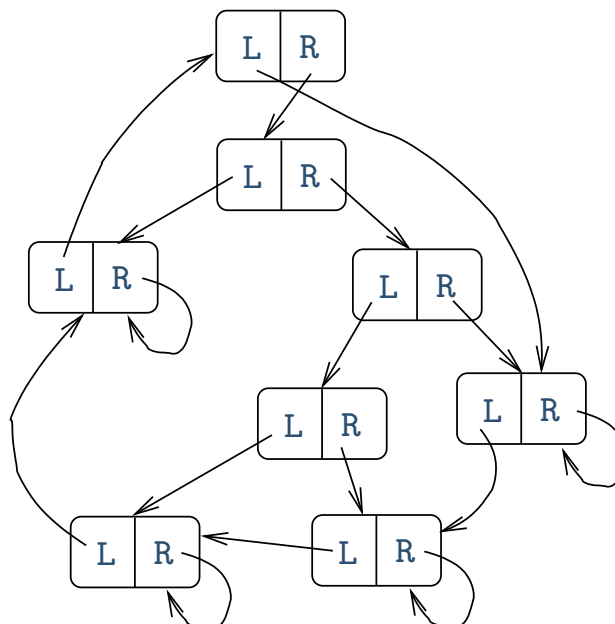
Weak cuts

Collusion

Problems

PBW

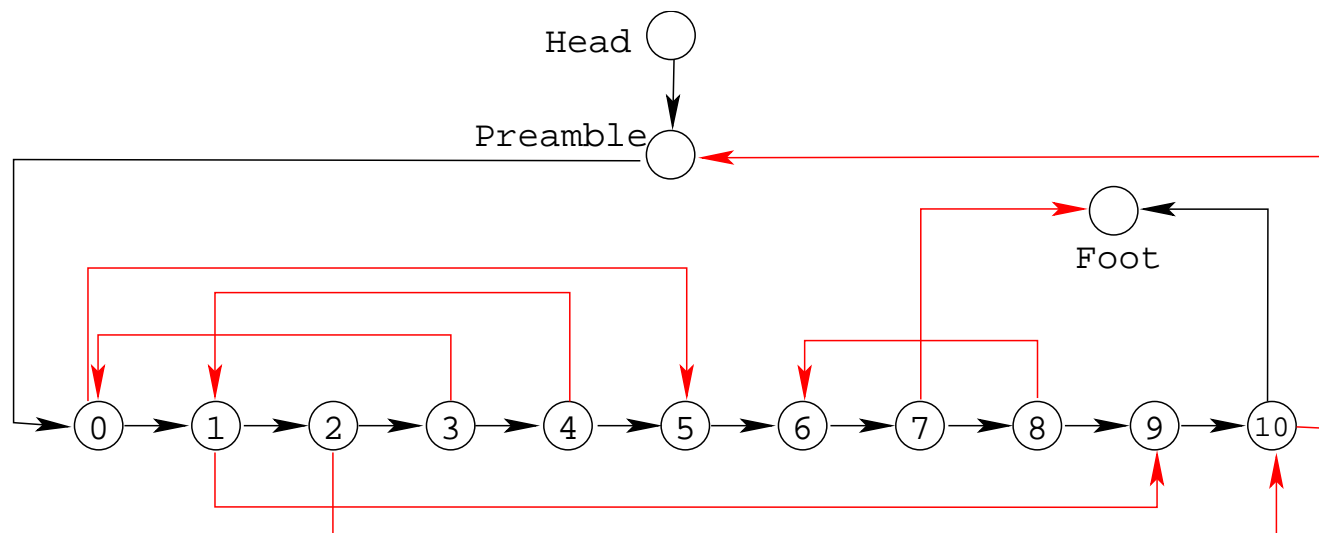
NT



- **Planted Plane Cubic Trees** can detect and correct one occurrence of node deletion or insertion.



Error-Correcting Graphs — RPG

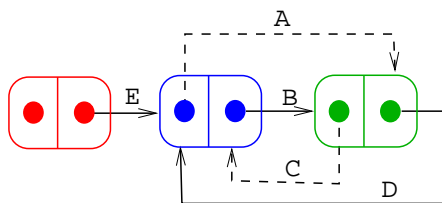


- **Reducible Permutation Graphs** can be decoded and corrected for edge-flips in polynomial time.

Collberg et al., Workshop on Graphs in Computer Science 2003.



Error-Correcting Graphs — Cycled Graphs



- Turn every node into a 3-cycle and every edge into a path of length 3 during embedding.
- During extraction the cycles and paths are contracted back to the original graph.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

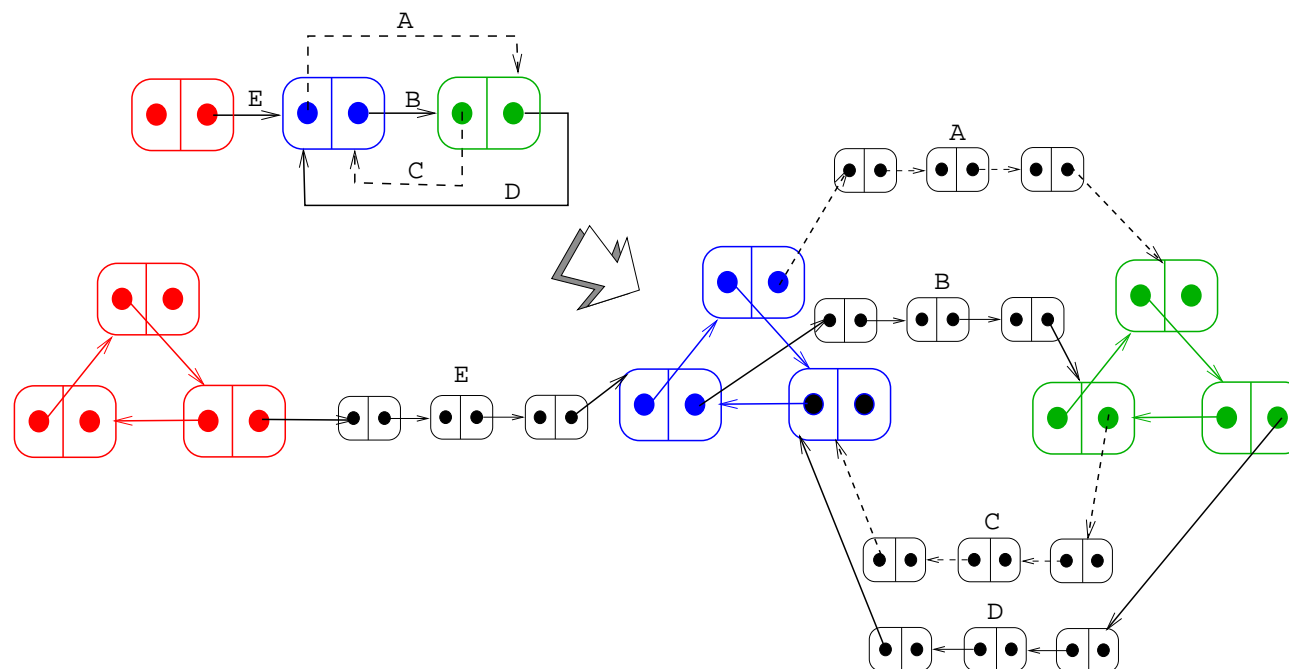
Problems

PBW

NT



Error-Correcting Graphs — Cycled Graphs



- Turn every node into a 3-cycle and every edge into a path of length 3 during embedding.
- During extraction the cycles and paths are contracted back to the original graph.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

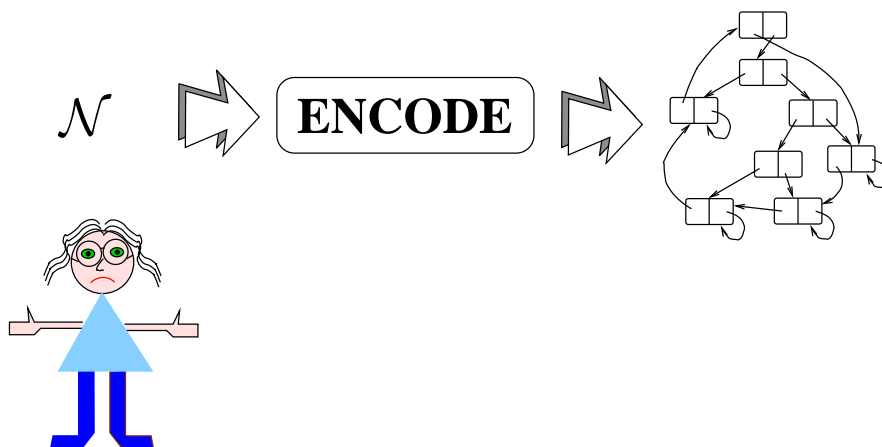
Problems

PBW

NT



Research problem: Graph Encoding Properties



- **ENCODE** should produce **small**, **directed multigraphs** with an **ordering on the outgoing edges** and **low max out-degree**.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

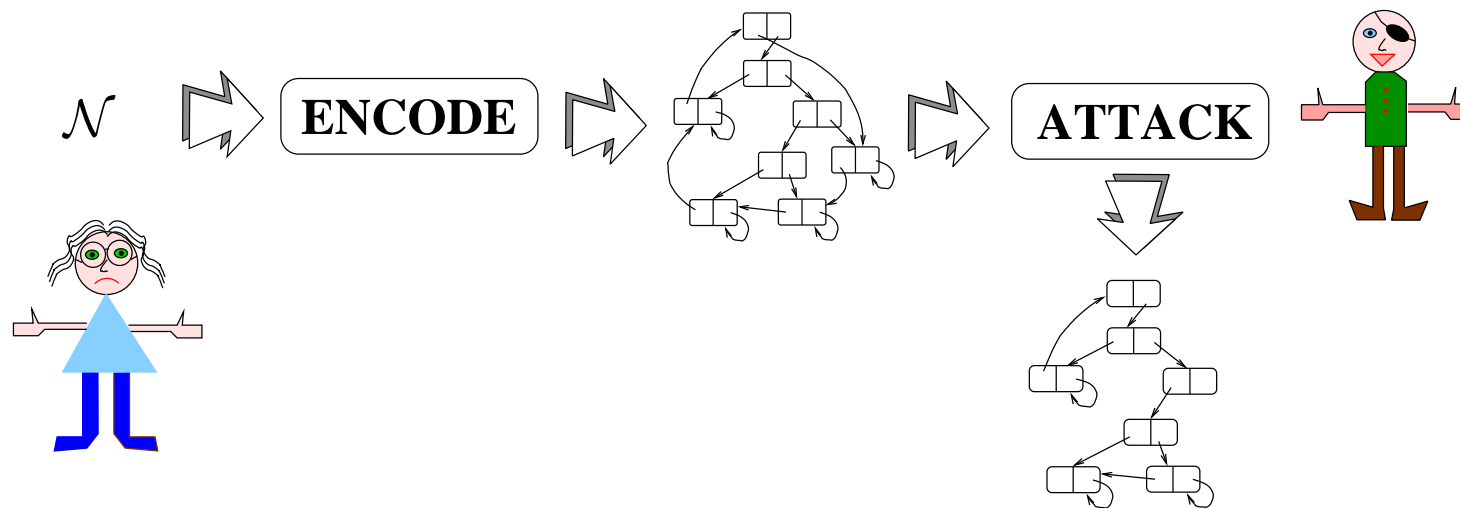
Problems

PBW

NT



Research problem: Graph Encoding Properties



- **ENCODE** should produce **small, directed multigraphs** with an **ordering on the outgoing edges** and **low max out-degree**.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

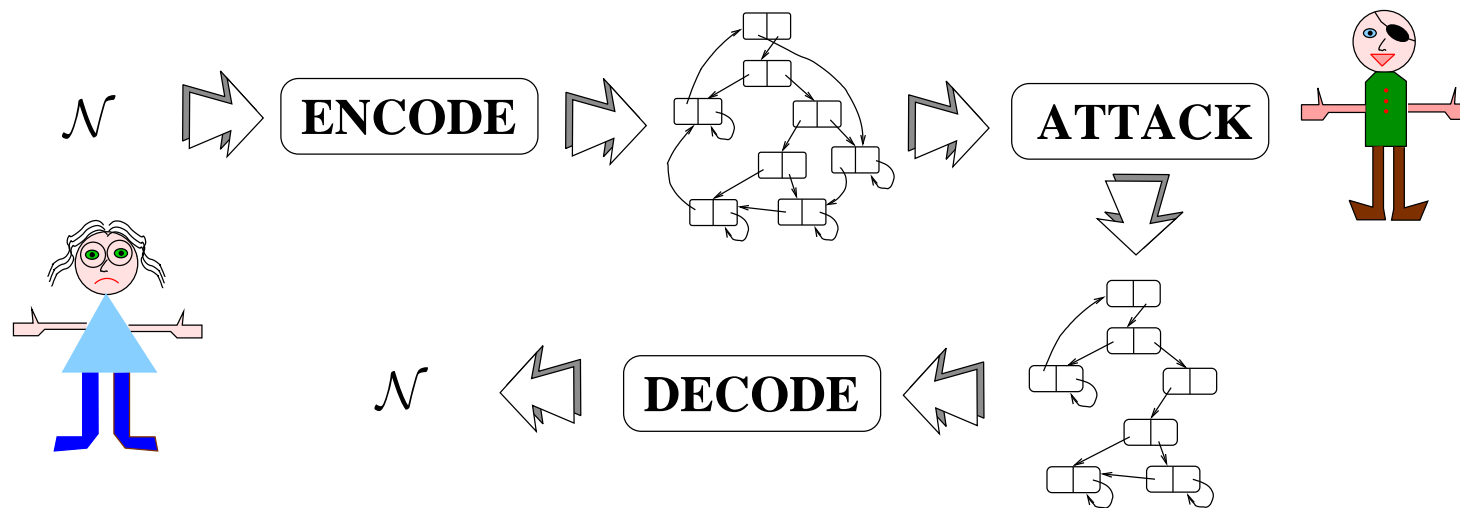
Problems

PBW

NT



Research problem: Graph Encoding Properties



- **ENCODE** should produce **small, directed multigraphs** with an **ordering on the outgoing edges** and **low max out-degree**.
- **DECODE** should be **insensitive to small perturbations** and be **polynomial time efficient**.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Avoiding Bogus Field Addition

- Assume that we have a graph node Node:

```
class Node {  
    public int a;  
    public Node left , right;  
}
```

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Avoiding Bogus Field Addition

- Assume that we have a graph node Node:

```
class Node {  
    public int a;  
    public Node left , right;  
}
```

- Reflection lets us check the integrity of this type at runtime:

```
Field [] F = Node.class.getFields();  
if (F.length != 3) die();  
if (F[1].getType() != Node.class) die();
```

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Avoiding Bogus Field Addition

- Assume that we have a graph node Node:

```
class Node {  
    public int a;  
    public Node left , right;  
}
```

- Reflection lets us check the integrity of this type at runtime:

```
Field [] F = Node.class.getFields();  
if (F.length != 3) die();  
if (F[1].getType() != Node.class) die();
```

- Unfortunately, this type of code is unstealthy.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

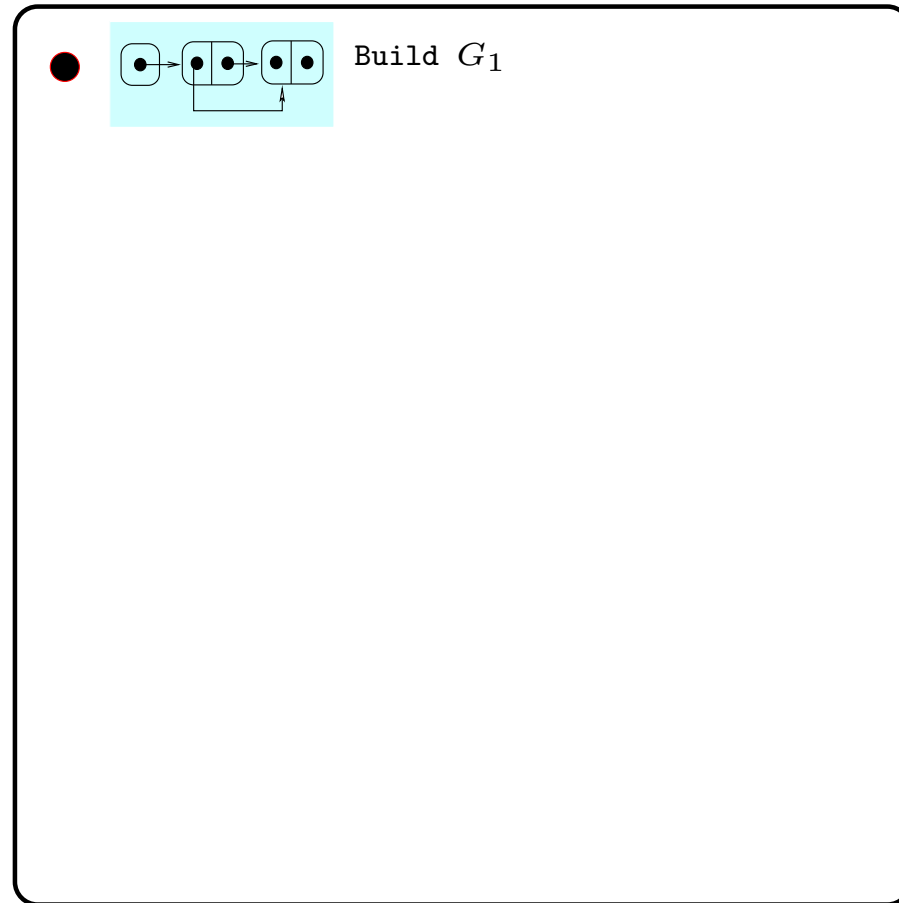
Collusion

Problems

PBW

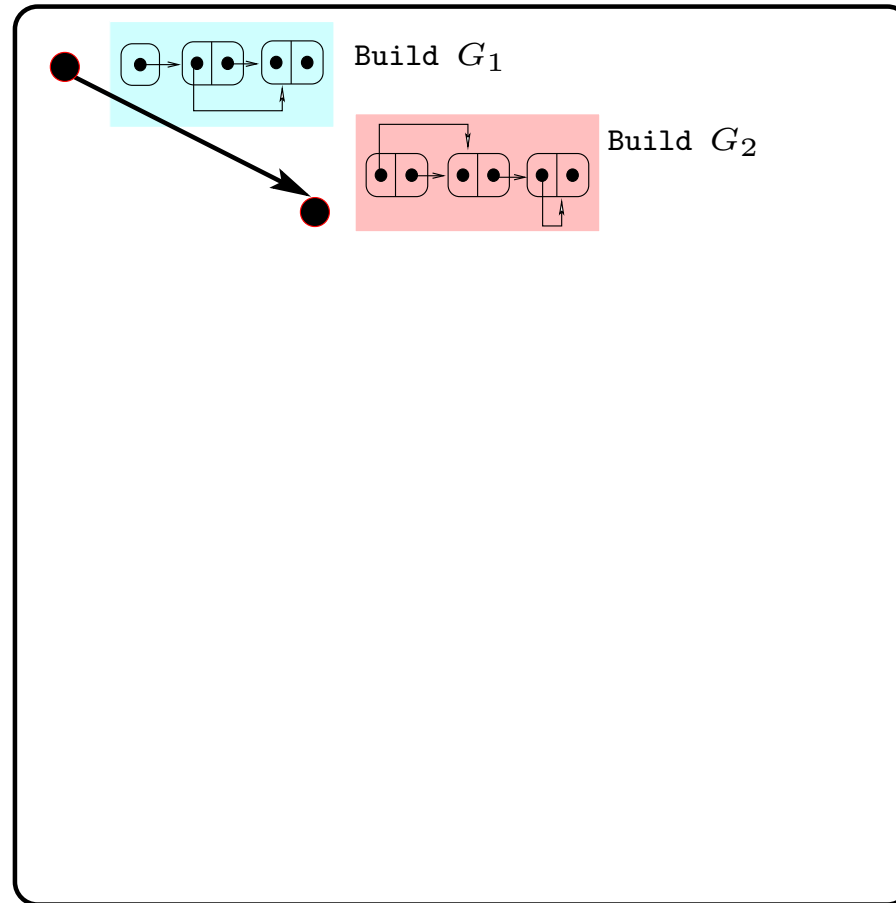
NT

Avoiding Alias Analysis



- Why? Static analysis algorithms fail on dynamically changing structures.

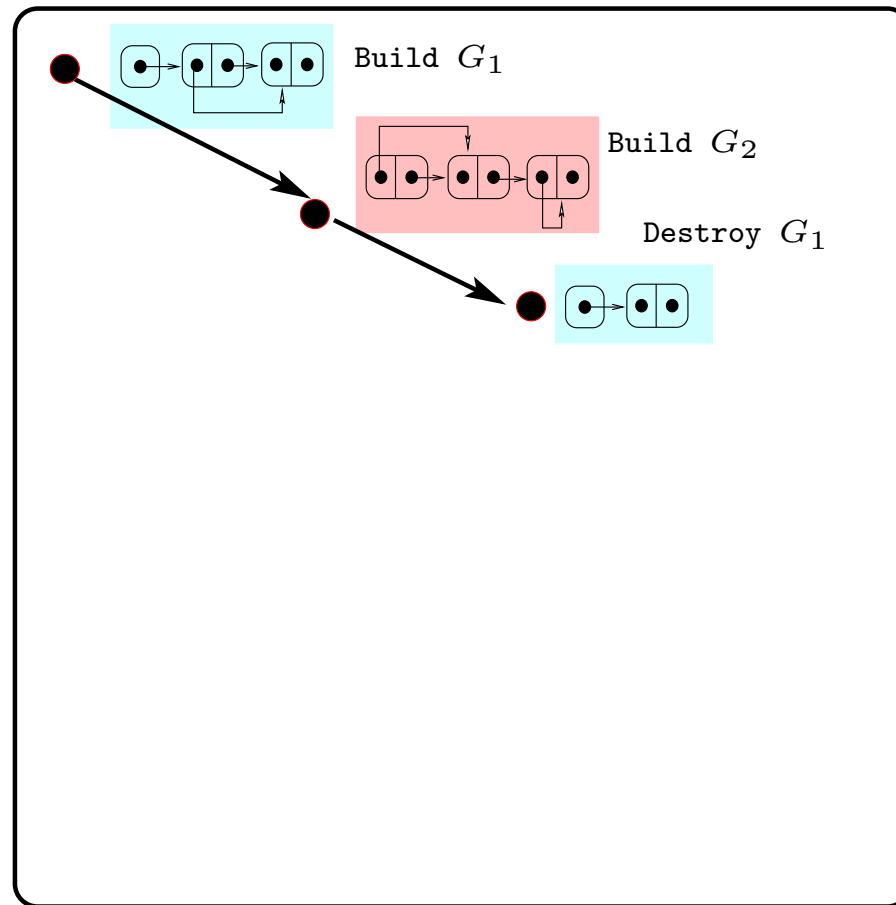
Avoiding Alias Analysis



- Why? Static analysis algorithms fail on dynamically changing structures.

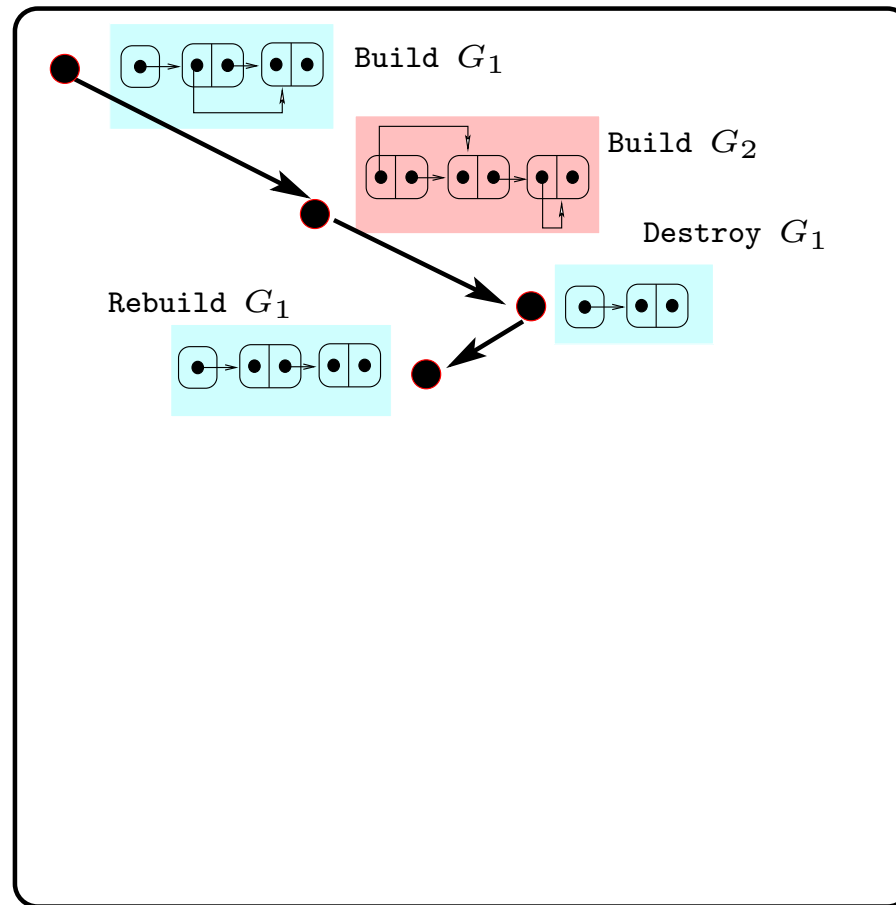


Avoiding Alias Analysis



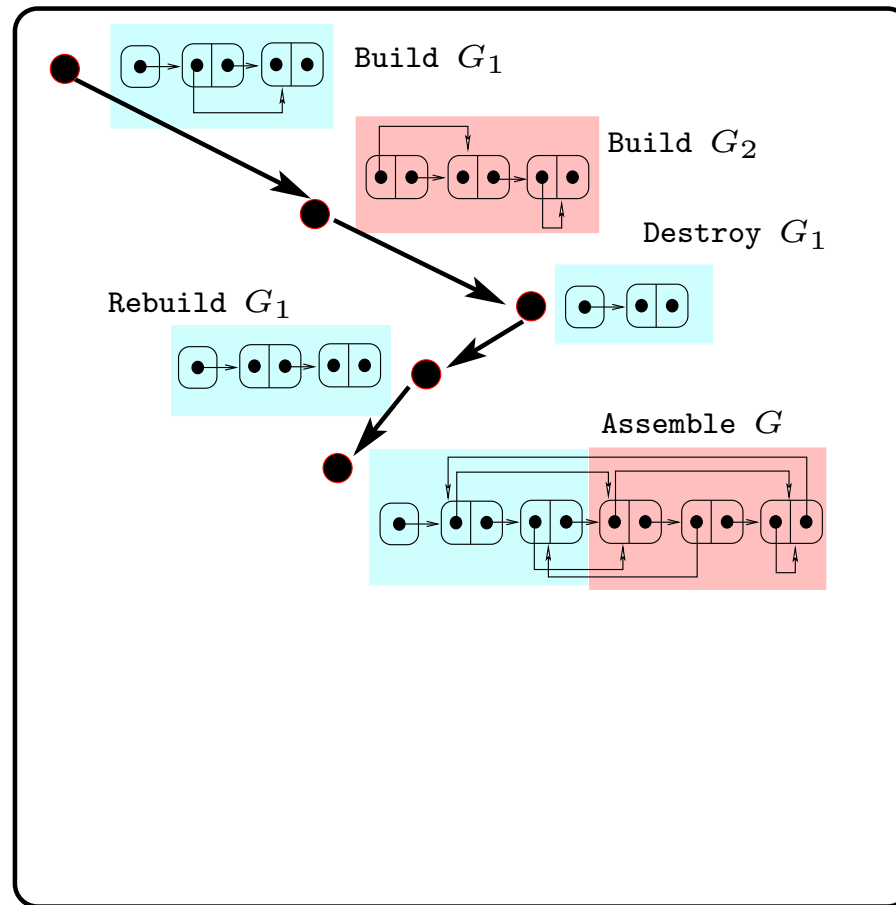
- Why? Static analysis algorithms fail on dynamically changing structures.

Avoiding Alias Analysis



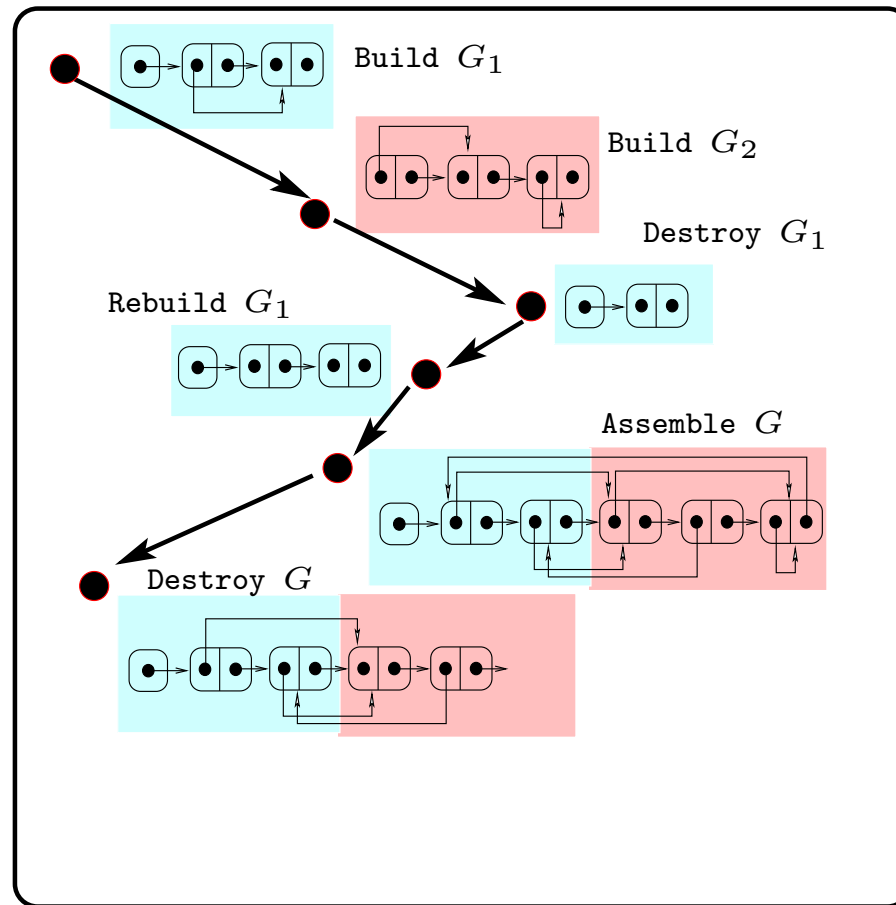
- Why? Static analysis algorithms fail on dynamically changing structures.

Avoiding Alias Analysis



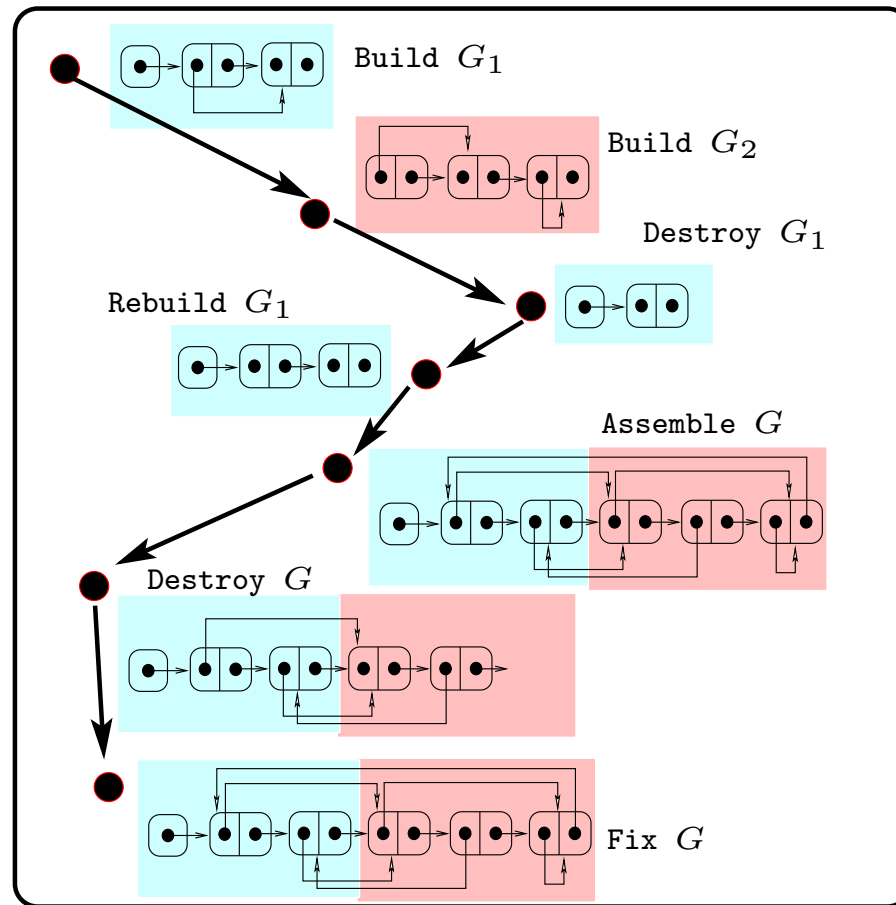
- Why? Static analysis algorithms fail on dynamically changing structures.

Avoiding Alias Analysis



- Why? Static analysis algorithms fail on dynamically changing structures.

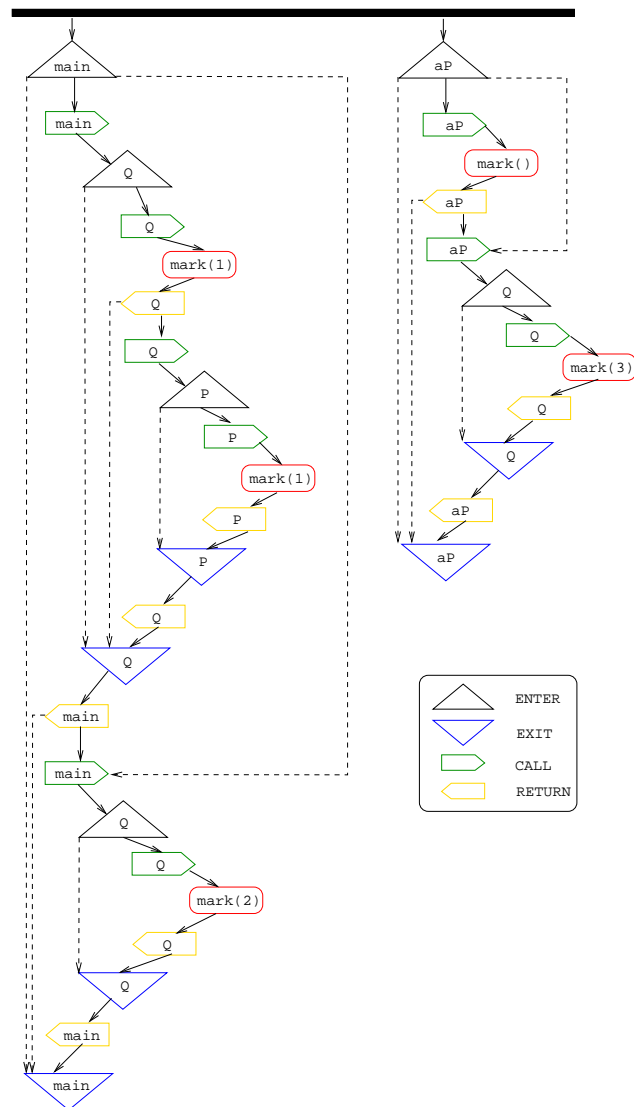
Avoiding Alias Analysis



- Why? Static analysis algorithms fail on dynamically changing structures.



Avoiding Global Variables



```
public class Main {  
    public static Node root;  
  
    public static void main(String args[]){  
        Node n2 = new Node();  
        Node n1 = new Node();  
        n1.left = n2;  n1.right = n2;  
        root = n1;  
    }  
}
```

- We shouldn't store secrets in globals, since there are few of them!
- Analyze the flow to pass secrets along in bogus method arguments.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Avoiding Unstealthy Node Classes

- Don't create unstealthy extra watermark classes:

```
class Node{  
    public Node left , right ;  
}
```

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT

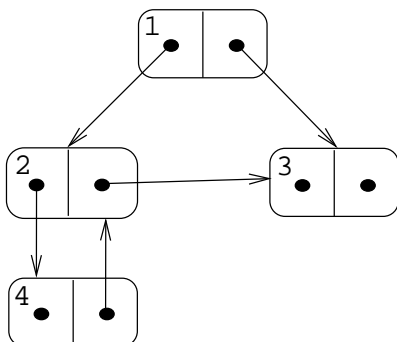


Avoiding Unstealthy Node Classes

- Don't create unstealthy extra watermark classes:

```
class Node{  
    public Node left , right ;  
}
```

- Instead reuse user or standard library classes:



```
LinkedList n4 = new LinkedList();  
n4.add( null );  
LinkedList n2 = new LinkedList();  
n2.add( n4 );  
n4.add( n2 );  
Event n3 = new Event( null , 0 , null );  
n2.add( n3 );  
Object [] n1 = { n2 , n3 };
```

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

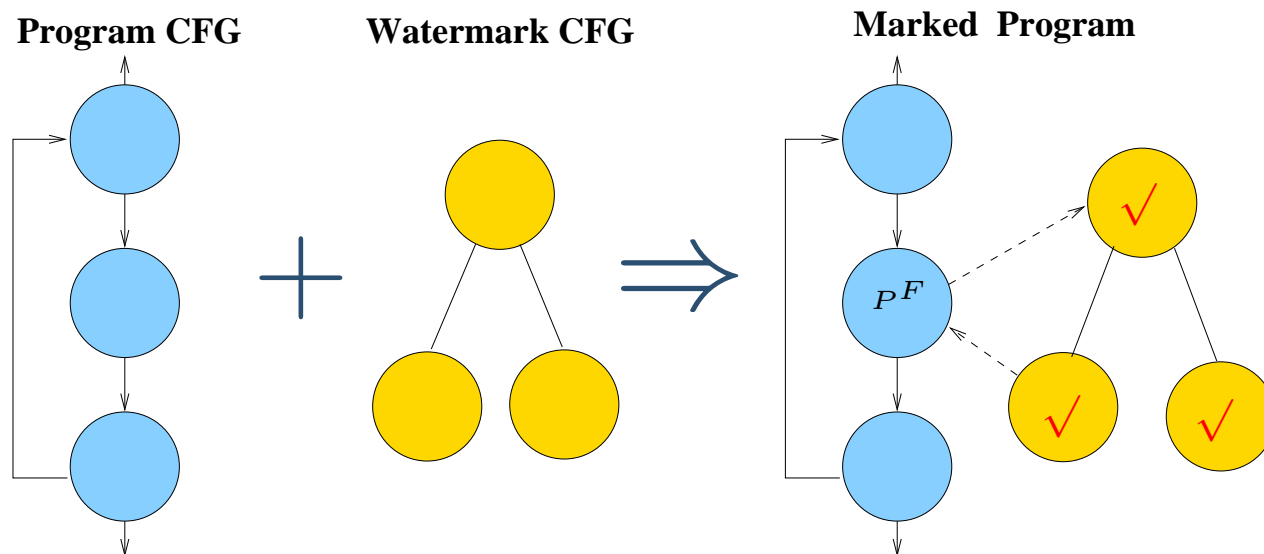
PBW

NT



Avoiding Weak Cuts

- We can avoid weak cuts by adding many bogus jumps across the code, a la Venkatesan:



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

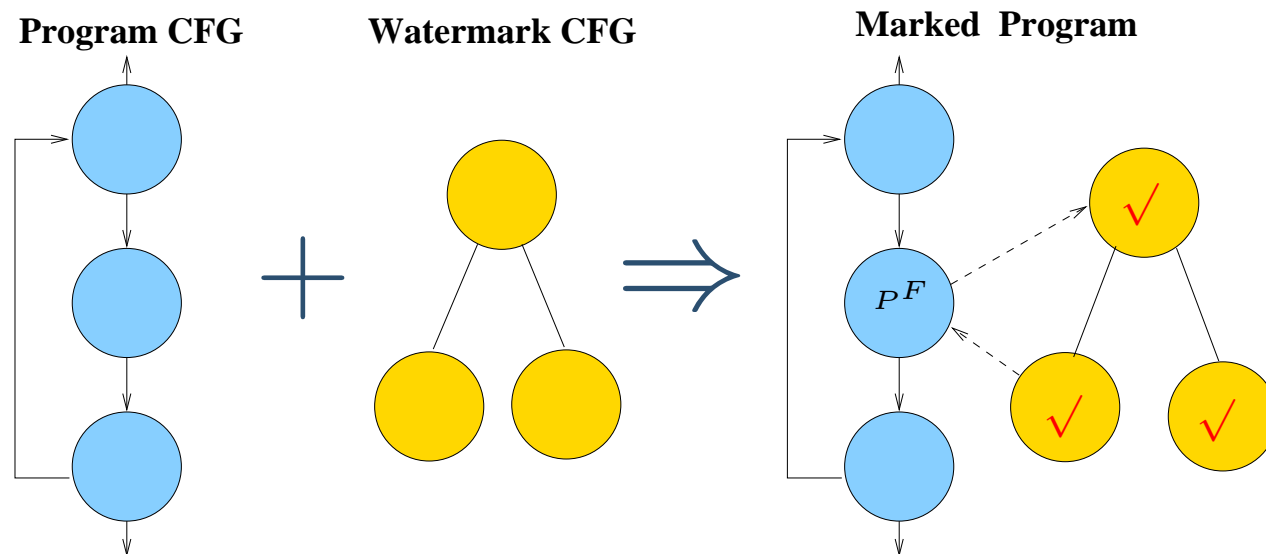
PBW

NT

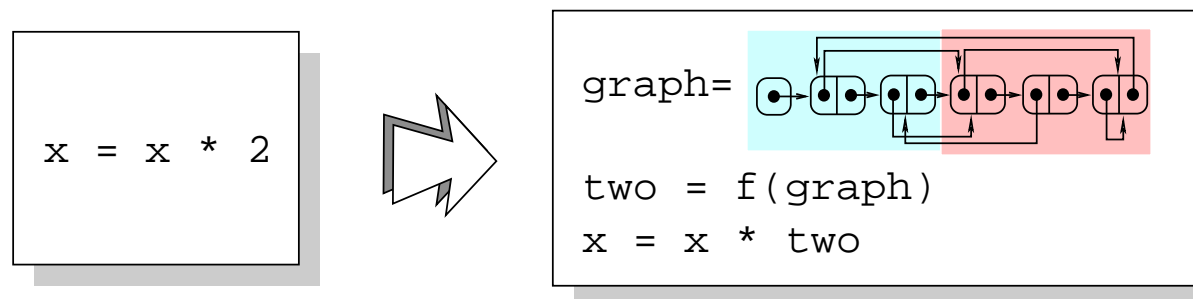


Avoiding Weak Cuts

- We can avoid weak cuts by adding many bogus jumps across the code, a la Venkatesan:



- Compute a value from the graph and use it in the code:





Avoiding static collusive attacks

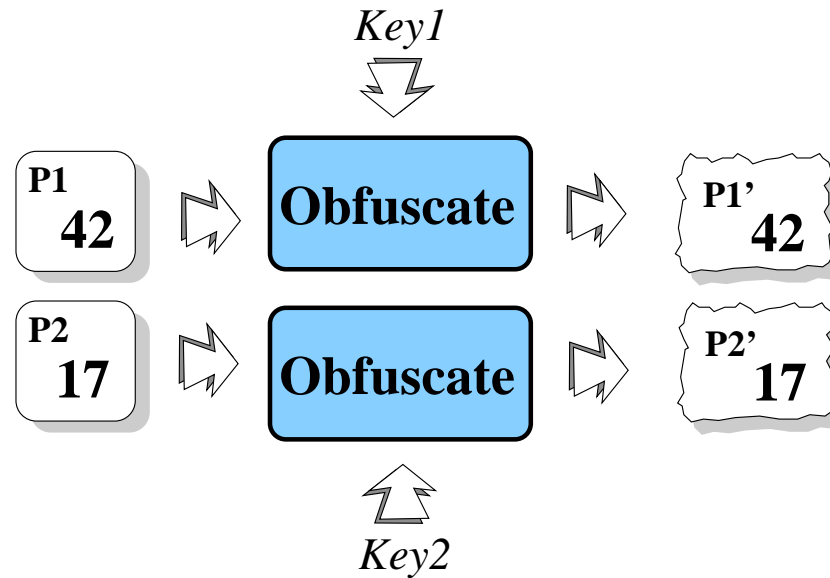
- Dynamic Watermarks
- CT
- Problems
- Increasing bit-rate
- Graph Attacks
- Graph Encoding
- Bogus fields
- Alias analysis
- Global roots
- Unstealthy nodes
- Weak cuts
- Collusion**
- Problems
- PBW
- NT

P1
42

P2
17

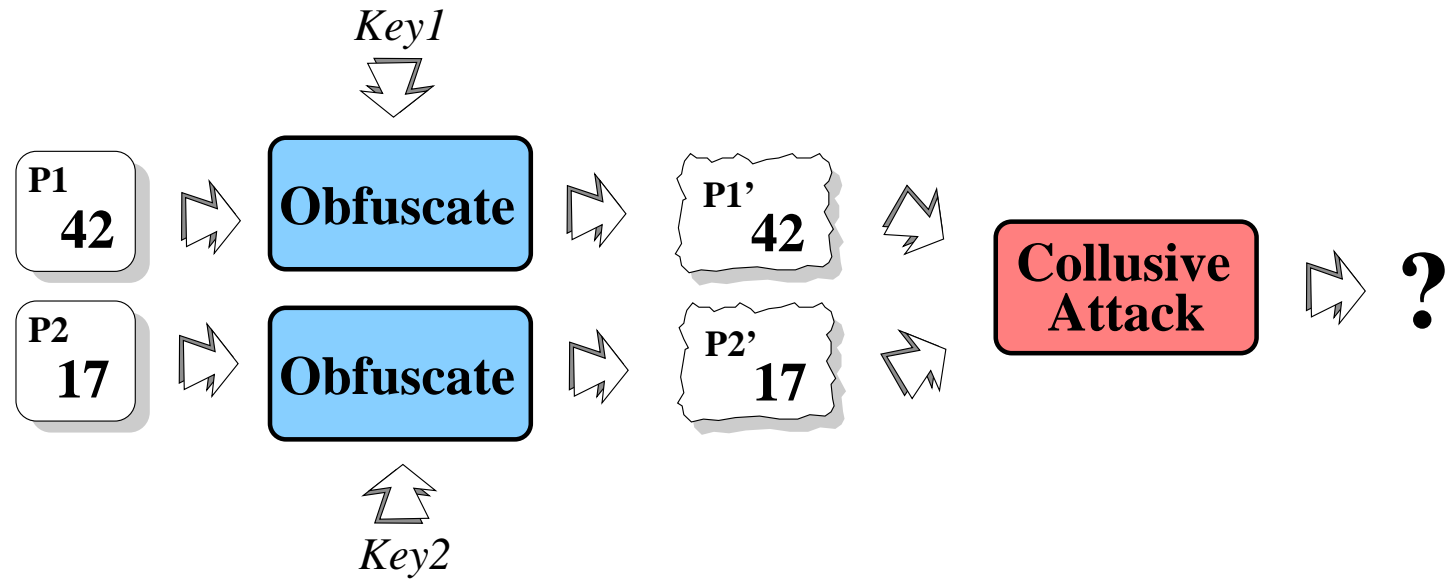
- Obfuscation can be used to protect against static collusive attacks.

Avoiding static collusive attacks



- Obfuscation can be used to protect against static collusive attacks.

Avoiding static collusive attacks



- Obfuscation can be used to protect against static collusive attacks.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

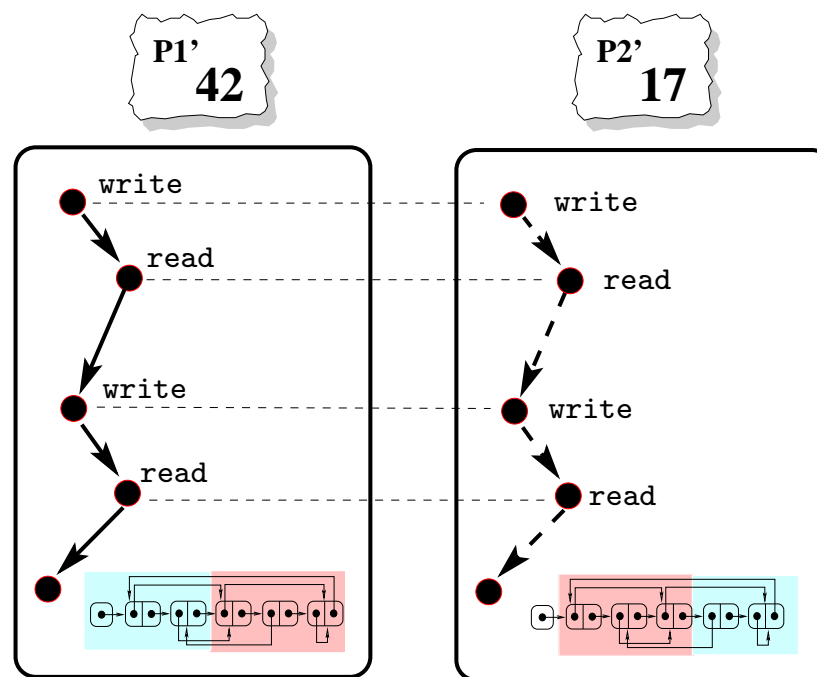
PBW

NT



Avoiding Dynamic Collusive Attacks

- The adversary can run two differently obfuscated versions of two differently fingerprinted programs, lining up code points at system calls:



Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

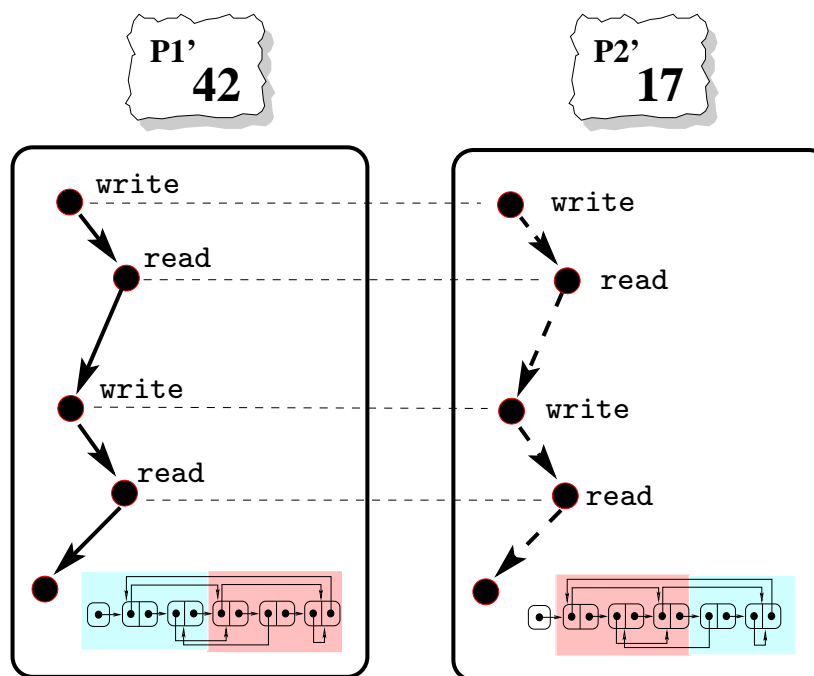
PBW

NT



Avoiding Dynamic Collusive Attacks

- The adversary can run two differently obfuscated versions of two differently fingerprinted programs, lining up code points at system calls:



- Now he only needs to analyze the code between system calls
- much less work!

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

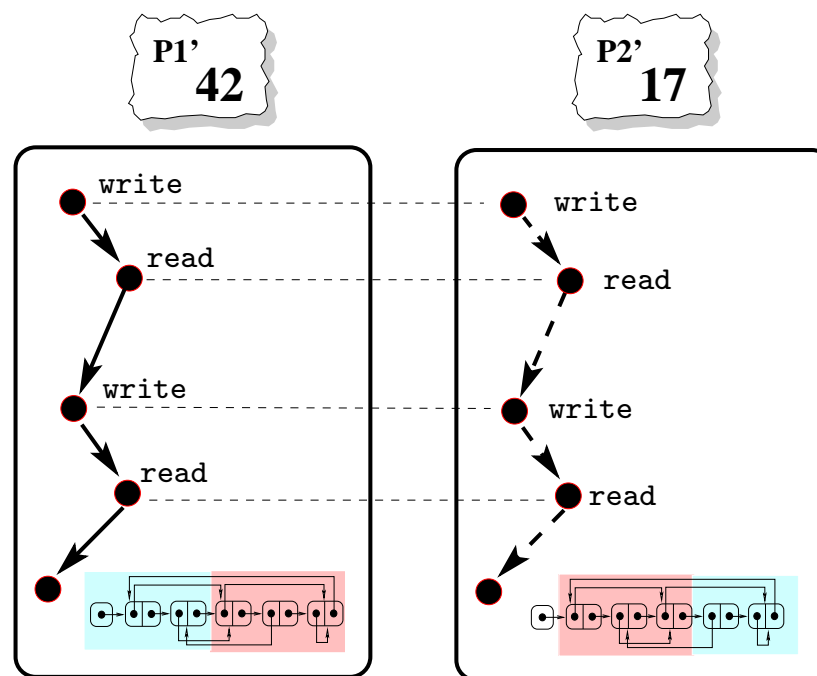
PBW

NT



Avoiding Dynamic Collusive Attacks

- The adversary can run two differently obfuscated versions of two differently fingerprinted programs, lining up code points at system calls:



- Now he only needs to analyze the code between system calls - much less work!
- Hard to insert extra system calls to confuse him!

Dynamic Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)
6. introducing extra unstealthy classes? (**user/lib classes**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)
6. introducing extra unstealthy classes? (**user/lib classes**)
7. weak cuts? (**compute values from graph**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)
6. introducing extra unstealthy classes? (**user/lib classes**)
7. weak cuts? (**compute values from graph**)
8. static collusive attacks? (**obfuscation**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)
6. introducing extra unstealthy classes? (**user/lib classes**)
7. weak cuts? (**compute values from graph**)
8. static collusive attacks? (**obfuscation**)
9. dynamic collusive attacks? (**uhm?**)

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
2. attacks by small graph perturbations? (**ECGs**)
3. bogus field addition? (**reflection**)
4. attacks by alias analysis? (**destructive updates**)
5. global variables? (**method params**)
6. introducing extra unstealthy classes? (**user/lib classes**)
7. weak cuts? (**compute values from graph**)
8. static collusive attacks? (**obfuscation**)
9. dynamic collusive attacks? (**uhm?**)

■ Attention to detail is important!

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
 2. attacks by small graph perturbations? (**ECGs**)
 3. bogus field addition? (**reflection**)
 4. attacks by alias analysis? (**destructive updates**)
 5. global variables? (**method params**)
 6. introducing extra unstealthy classes? (**user/lib classes**)
 7. weak cuts? (**compute values from graph**)
 8. static collusive attacks? (**obfuscation**)
 9. dynamic collusive attacks? (**uhm?**)
- Attention to detail is important!
 - Stealth is hard!

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



Problems Revisited

How do we avoid

1. huge graphs? (**new graph classes**)
 2. attacks by small graph perturbations? (**ECGs**)
 3. bogus field addition? (**reflection**)
 4. attacks by alias analysis? (**destructive updates**)
 5. global variables? (**method params**)
 6. introducing extra unstealthy classes? (**user/lib classes**)
 7. weak cuts? (**compute values from graph**)
 8. static collusive attacks? (**obfuscation**)
 9. dynamic collusive attacks? (**uhm?**)
- Attention to detail is important!
 - Stealth is hard!
 - Dynamic attacks are harder!

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

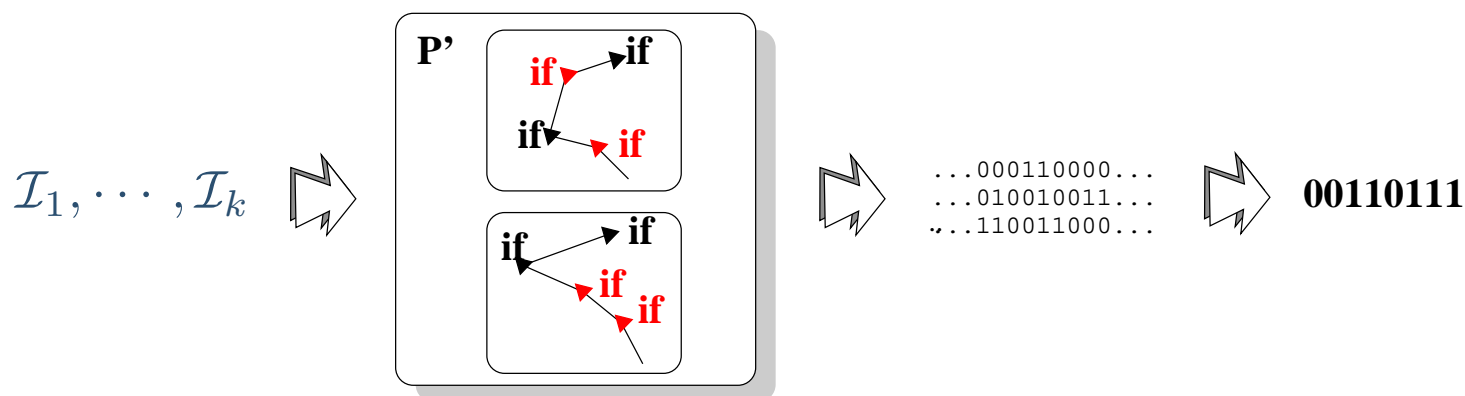
Problems

PBW

NT



— Path-Based Watermarking



- The branches executed for the secret input generate a stream of 0s and 1s from which the watermark is extracted.
- An attacker can easily insert new branches:

Java \Rightarrow Use an Error Correcting Code

x86 \Rightarrow Tamper-proof the branches

Collberg et al., ACM PLDI'04

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

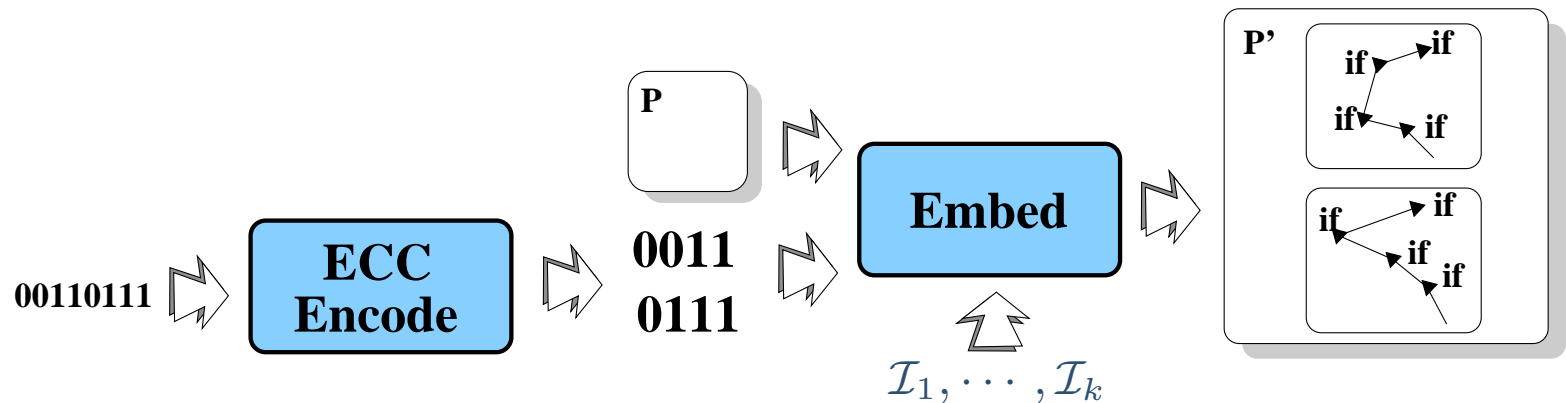
Collusion

Problems

PBW

NT

PBW — Embedding



- The watermark is split into a large number of redundant pieces using an error correcting code.
- Each piece is individually embedded in the program.
- We want to be able to lose some pieces and still recover the watermark.



PBW — Code Generation

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

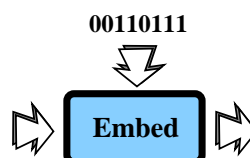
Collusion

Problems

PBW

NT

```
void main() {  
    int a=25,b=10;  
    while((a%b)!=0){  
        int t=b%a;  
        b = a;  
        a = t;  
    }  
    println(b);  
}
```

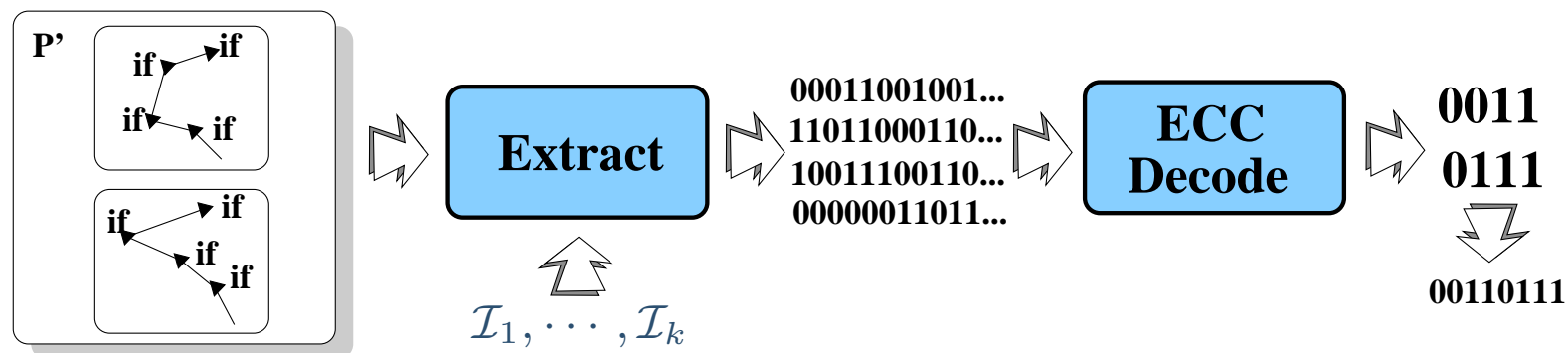


```
void main() {  
    int a=25,b=10;  
    int u=0,x=0x1a,c=8;  
    while((a%b)!=0) {  
        int t = b % a;  
        b = a; a = t;  
        if (t==a) u++;  
        if (t==a) u++;  
        if (a==10) u++;  
        if (a==10) u++;  
    }  
    for(int i=0; i<c;i++,x>>=1)  
        if ((x&1)==1) x|=1;  
    println(b);  
}
```

- Several different types of code can be generated to increase stealth.
- Ensure to protect against simple branch-flips!



PBW — Extraction



- The program is run with the secret input.
- Branches are monitored and a bitstream extracted.
- Using the error correcting code, the watermark pieces are extracted from the bitstream and recombined into the watermark.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



PBW — ECC Encode

$$p_1 = 2, p_2 = 3, p_3 = 5$$

$$W = 17 \Rightarrow \begin{aligned} W &\equiv 5 \pmod{p_1 p_2} \\ W &\equiv 7 \pmod{p_1 p_3} \\ W &\equiv 2 \pmod{p_2 p_3} \end{aligned}$$

- Choose p_1, \dots, p_k pairwise relatively prime, split watermark into $\frac{k(k-1)}{2}$ pieces of the form $W \equiv r \pmod{p_i p_k}$.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



PBW — ECC Encode

$$p_1 = 2, p_2 = 3, p_3 = 5$$

$$\begin{array}{rclclcl} & W & \equiv & 5 \bmod p_1 p_2 & & 5 & = & 5 \\ W = 17 \Rightarrow & W & \equiv & 7 \bmod p_1 p_3 & \Rightarrow & p_1 p_2 + 7 & = & 13 \\ & W & \equiv & 2 \bmod p_2 p_3 & & p_1 p_2 + p_1 p_3 + 2 & = & 18 \end{array}$$

- Choose p_1, \dots, p_k pairwise relatively prime, split watermark into $\frac{k(k-1)}{2}$ pieces of the form $W \equiv r \bmod p_i p_k$.
- Use an enumeration scheme to turn these into integers, run through a block-cipher, embed into program.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



PBW — ECC Encode

$$p_1 = 2, p_2 = 3, p_3 = 5$$

$$W = 17 \Rightarrow \begin{array}{l} W \equiv 5 \pmod{p_1 p_2} \\ W \equiv 7 \pmod{p_1 p_3} \\ W \equiv 2 \pmod{p_2 p_3} \end{array} \Rightarrow \begin{array}{l} 5 \\ p_1 p_2 + 7 \\ p_1 p_2 + p_1 p_3 + 2 \end{array} = \begin{array}{l} 5 \\ 13 \\ 18 \end{array} \Rightarrow \begin{array}{c} \overbrace{11 \dots 01}^{64} \\ \overbrace{01 \dots 11}^{64} \\ \overbrace{10 \dots 00}^{64} \end{array}$$

- Choose p_1, \dots, p_k pairwise relatively prime, split watermark into $\frac{k(k-1)}{2}$ pieces of the form $W \equiv r \pmod{p_i p_k}$.
- Use an enumeration scheme to turn these into integers, run through a block-cipher, embed into program.
- The Generalized Chinese Remainder Theorem allows W to be reconstructed from $\lceil \frac{k}{2} \rceil$ pieces.

| |
|---------------------|
| Dynamic Watermarks |
| CT |
| Problems |
| Increasing bit-rate |
| Graph Attacks |
| Graph Encoding |
| Bogus fields |
| Alias analysis |
| Global roots |
| Unstealthy nodes |
| Weak cuts |
| Collusion |
| Problems |
| PBW |
| NT |



PBW — ECC Decode

- Slide a 64-bit window across the bitstream. Throw out those that don't meet randomness criteria.

...1011100110100000000000011010010010...

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



PBW — ECC Decode

- Slide a 64-bit window across the bitstream. Throw out those that don't meet randomness criteria.

... 1011100110100000000000011010010010...

- Reconstruct $W \equiv r \bmod p_i p_k$ by inverting enumeration scheme.

$$\begin{array}{l} 11 \dots 01 \\ 01 \dots 11 \\ 10 \dots 00 \\ 10 \dots 11 \end{array} \Rightarrow \begin{array}{c} 5 \\ 13 \\ 17 \\ 0 \end{array} \Rightarrow \begin{array}{l} W \equiv 5 \bmod p_1 p_2 \\ W \equiv 7 \bmod p_1 p_3 \\ W \equiv 1 \bmod p_2 p_3 \\ W \equiv 0 \bmod p_1 p_2 \end{array}$$

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

PBW

NT



PBW — ECC Decode

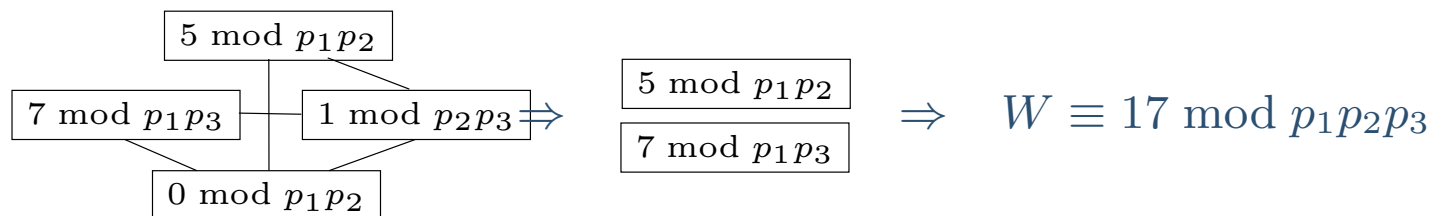
- Slide a 64-bit window across the bitstream. Throw out those that don't meet randomness criteria.

... 1011100110 0000000000 011010010010...

- Reconstruct $W \equiv r \bmod p_i p_k$ by inverting enumeration scheme.

$$\begin{array}{l} 11 \dots 01 \\ 01 \dots 11 \\ 10 \dots 00 \\ 10 \dots 11 \end{array} \Rightarrow \begin{array}{c} 5 \\ 13 \\ 17 \\ 0 \end{array} \Rightarrow \begin{array}{l} W \equiv 5 \bmod p_1 p_2 \\ W \equiv 7 \bmod p_1 p_3 \\ W \equiv 1 \bmod p_2 p_3 \\ W \equiv 0 \bmod p_1 p_2 \end{array}$$

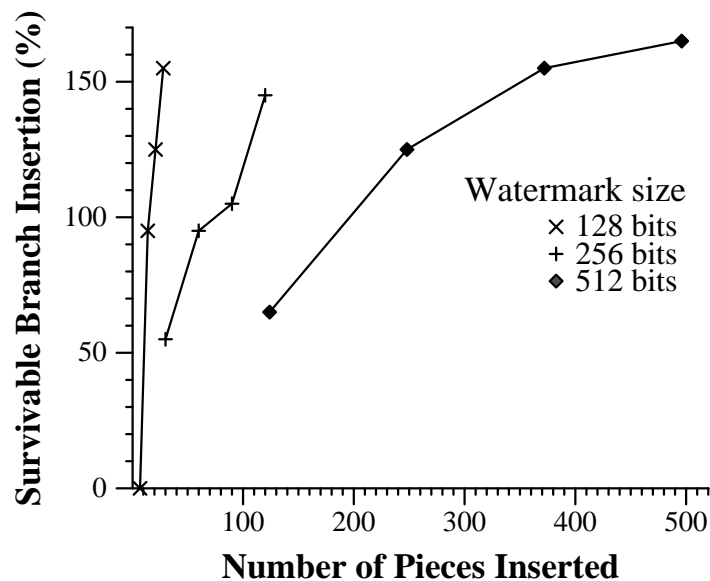
- Build a graph of statements inconsistent wrt to GCRT. Compute “most consistent” subgraph.





PBW — Adding Branches Attack

- Attack model: the attacker randomly adds bogus conditional branches to the program.
- The more pieces we add, the more pieces an attacker has to destroy in order to destroy the watermark



- With a 256-bit mark and 100 pieces, the attacker needs to double the number of branch instructions in the program in order to destroy the mark.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

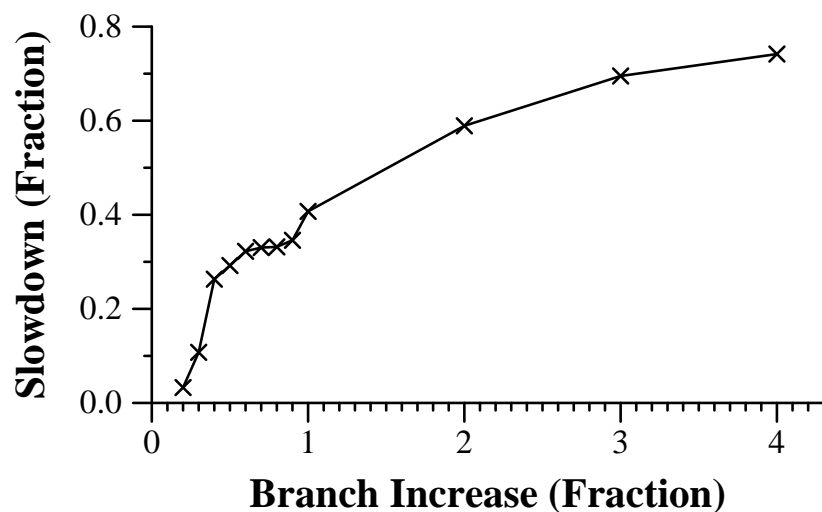
PBW

NT



PBW — Adding Branches Attack

- How much does CaffeineMark slow down wrt the number of branches the attacker added?



- By doubling the number of branches, the attacker slows down the program by about 40%.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

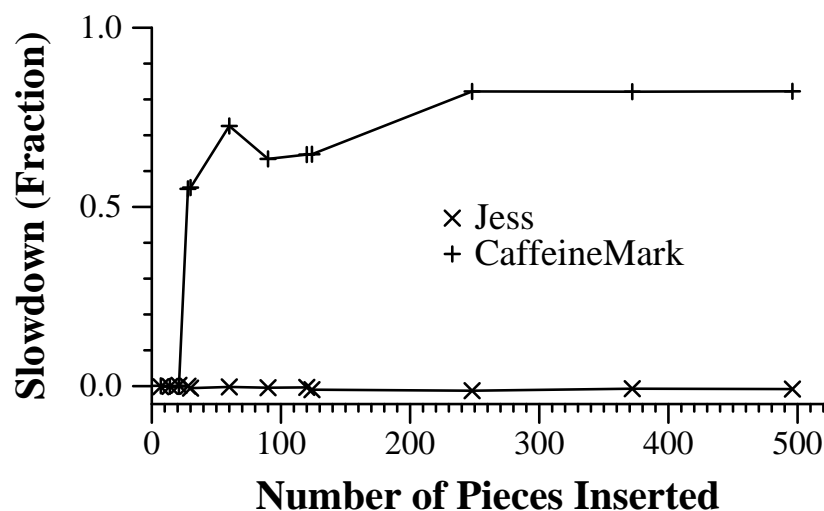
PBW

NT



PBW — Time Overhead

- How does the program slow down as the number of watermark pieces is increased?
- The more pieces we insert, the more pieces the attacker needs to destroy.



- For Jess we avoid the hotspots, so slowdown is negligible.
- For CaffeineMark we can't avoid the hotspots, so slowdown is $> 50\%$.

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

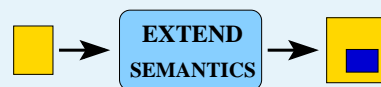
Weak cuts

Collusion

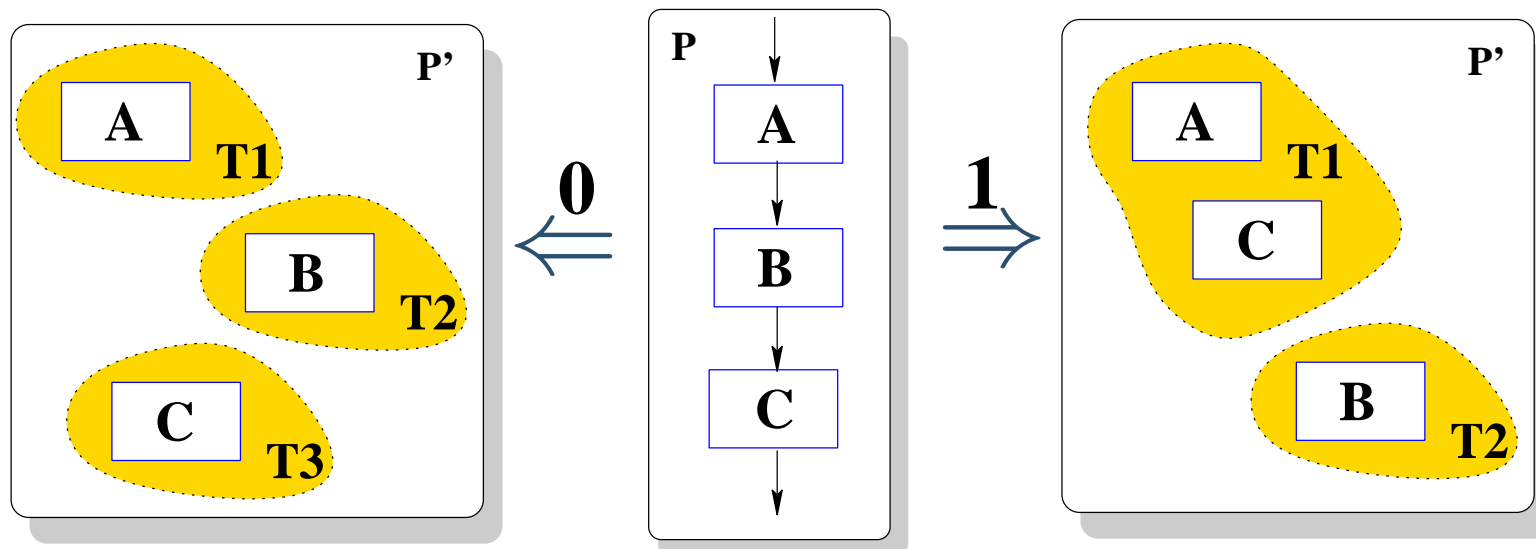
Problems

PBW

NT



— Thread-Based Watermark



- Embed mark in which threads execute which basic blocks.
- Can have huge performance degradation.
- Why? **Parallelism-analysis** is hard.

Nagra & Thomborson, 6th Information Hiding Workshop, IHW'04

Dynamic
Watermarks

CT

Problems

Increasing bit-rate

Graph Attacks

Graph Encoding

Bogus fields

Alias analysis

Global roots

Unstealthy nodes

Weak cuts

Collusion

Problems

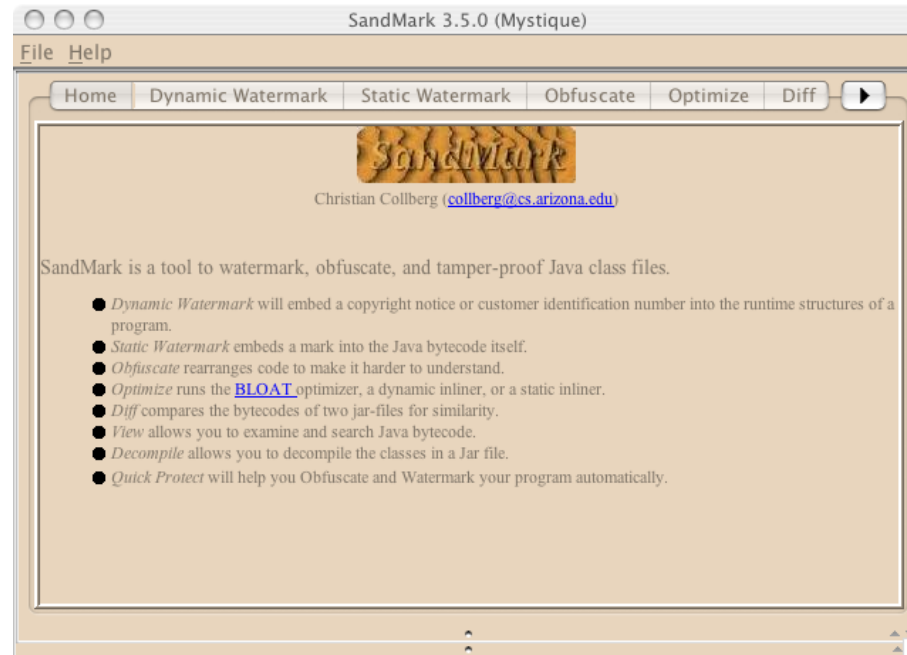
PBW

NT

SANDMARK — A Software Protection Tool

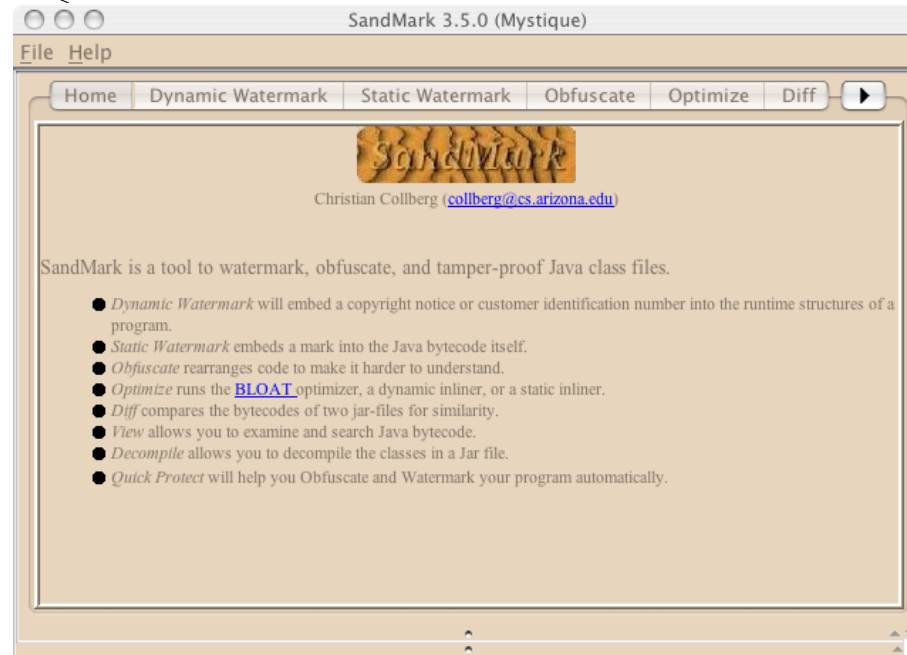


SandMark Birthmarking



SANDMARK — A Software Protection Tool

33 Obfuscation Algorithms



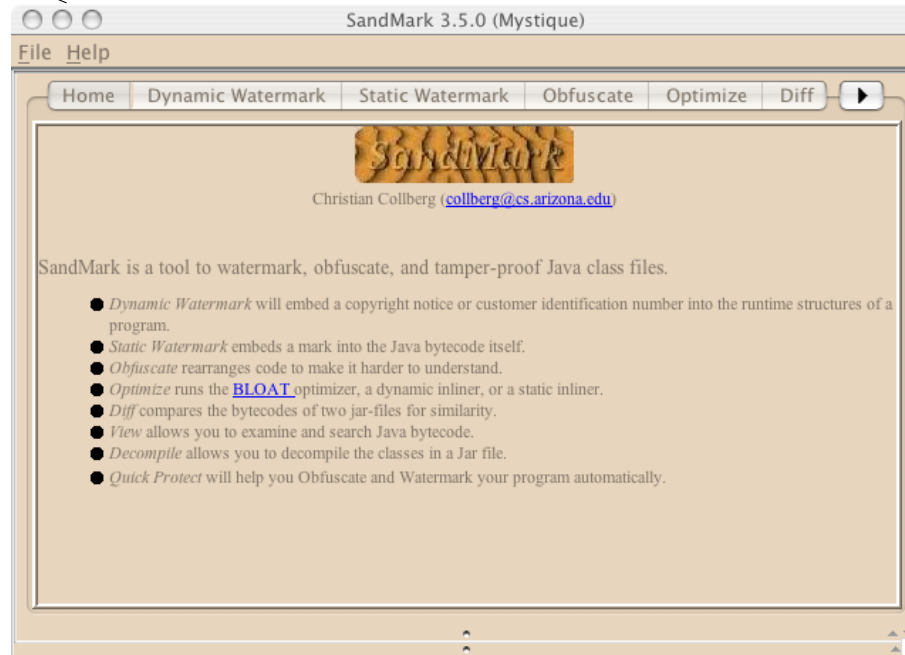
SandMark
Birthmarking

SANDMARK — A Software Protection Tool

33 Obfuscation Algorithms

16 Watermarking Algorithms

SandMark
Birthmarking



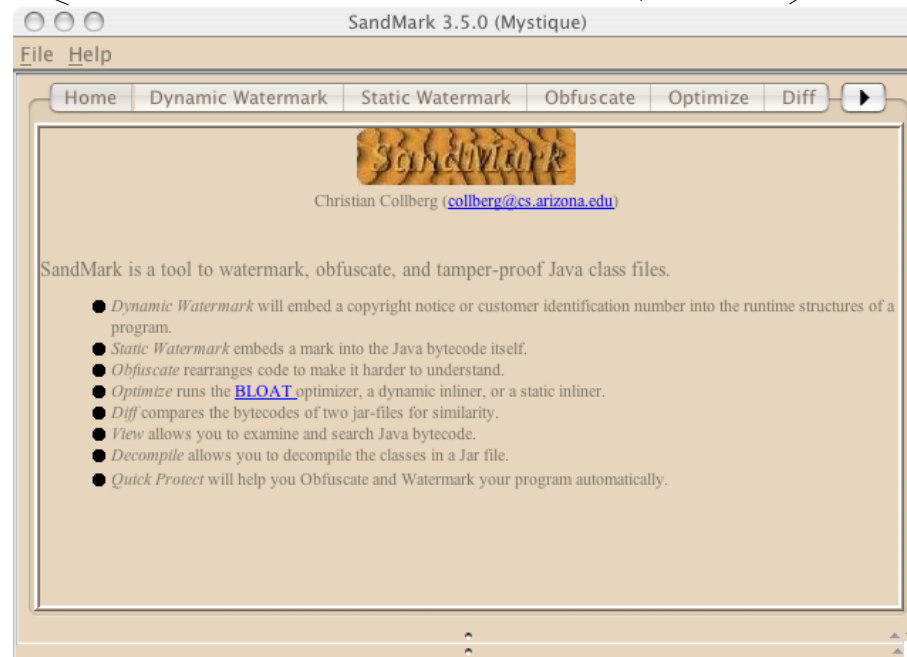
SANDMARK — A Software Protection Tool

33 Obfuscation Algorithms

16 Watermarking Algorithms

6 Birthmarking Algorithms

SandMark
Birthmarking



SANDMARK — A Software Protection Tool

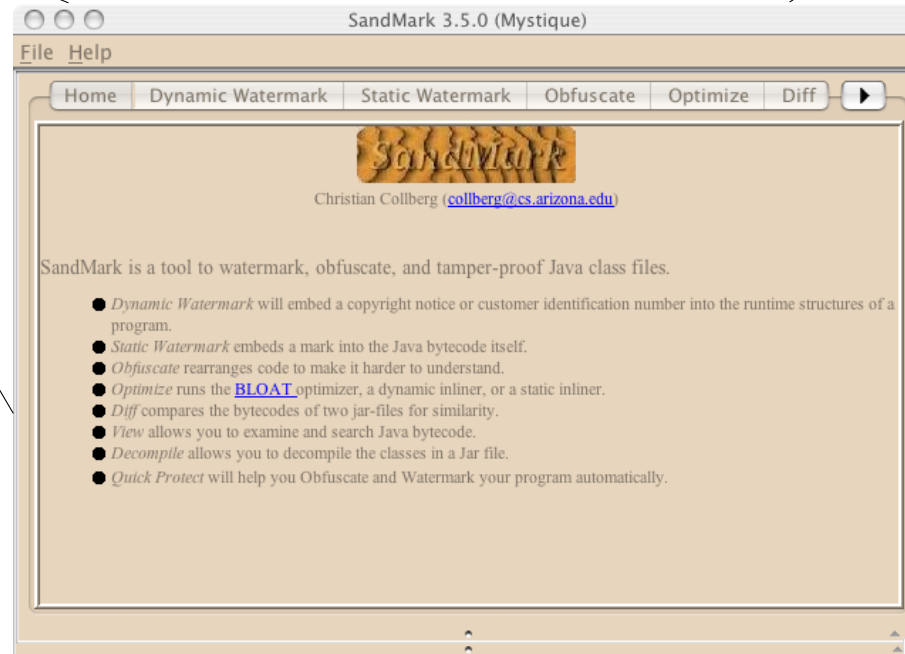


33 Obfuscation Algorithms

16 Watermarking Algorithms

6 Birthmarking Algorithms

6 bytecode
diff Algorithms



SandMark
Birthmarking



SANDMARK — A Software Protection Tool

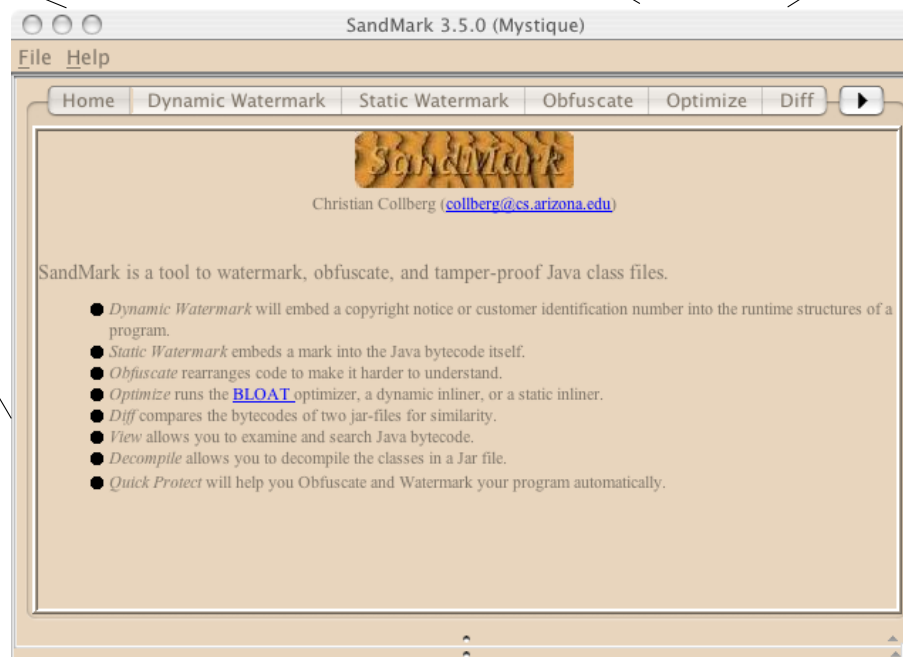
33 Obfuscation Algorithms

16 Watermarking Algorithms

6 Birthmarking Algorithms

6 bytecode
diff Algorithms

bytecode
visualization
tools



SandMark
Birthmarking



SANDMARK — A Software Protection Tool

33 Obfuscation Algorithms

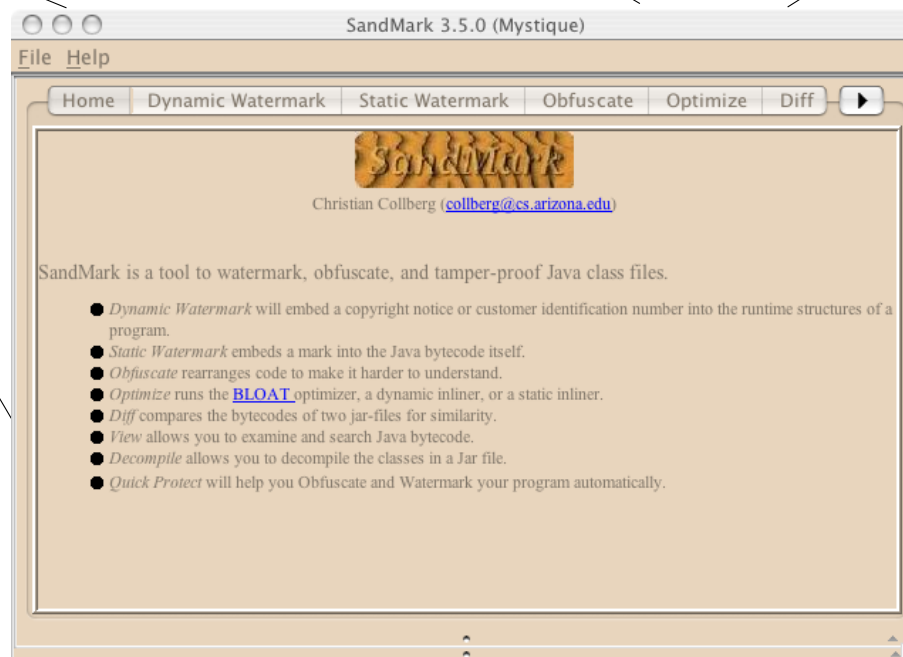
16 Watermarking Algorithms

6 Birthmarking Algorithms

6 bytecode
diff Algorithms

bytecode
visualization
tools

6 Software Complexity
metrics



SandMark
Birthmarking



SANDMARK — A Software Protection Tool

33 Obfuscation Algorithms

16 Watermarking Algorithms

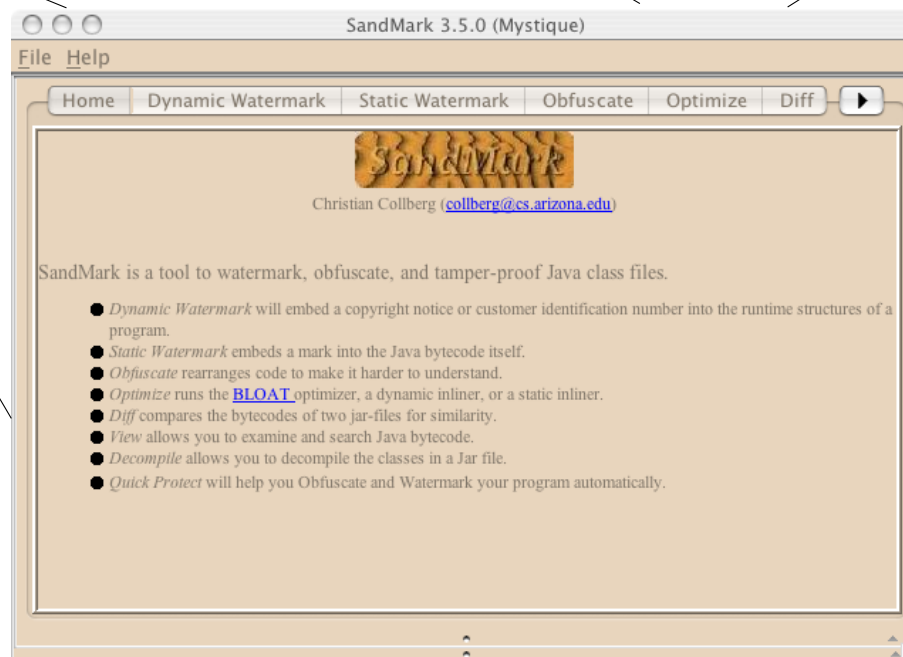
6 Birthmarking Algorithms

6 bytecode
diff Algorithms

bytecode
visualization
tools

6 Software Complexity
metrics

Large toolbox (graphs,
static analysis,etc)

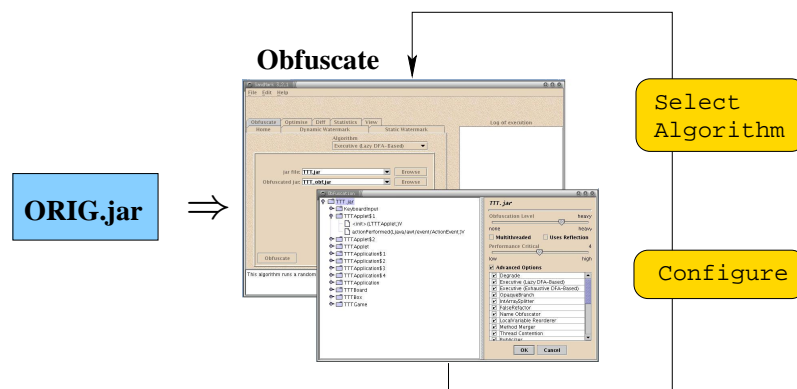


SandMark
Birthmarking



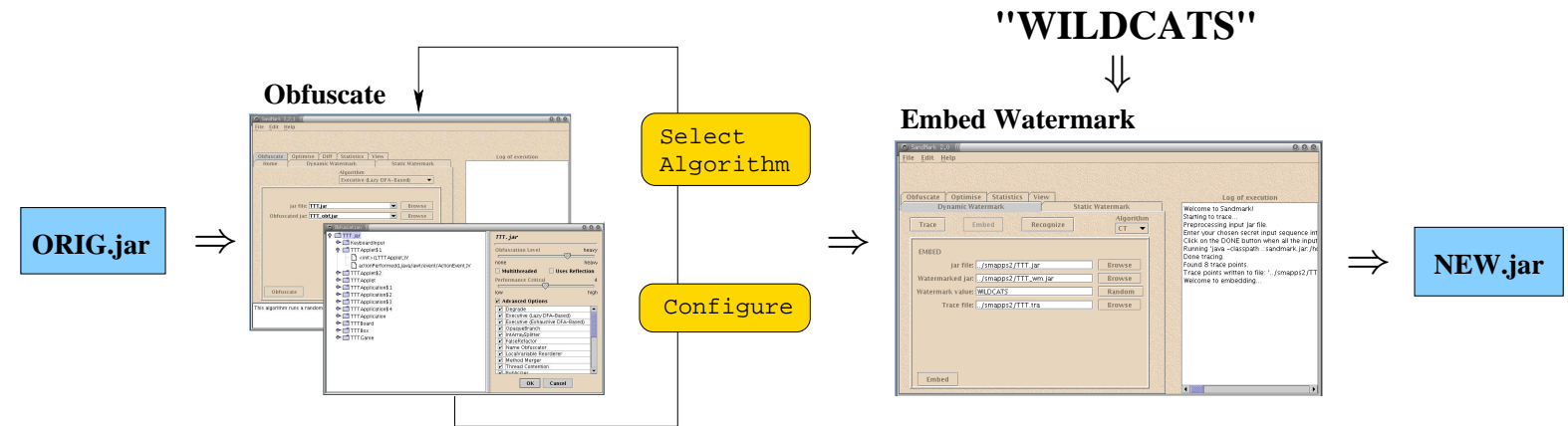
A Session with SANDMARK

SandMark
Birthmarking



- We obfuscate to protect against attacks by
 1. reverse engineering
 2. collusive de-watermarking

A Session with SANDMARK



- We obfuscate to protect against attacks by
 1. reverse engineering
 2. collusive de-watermarking



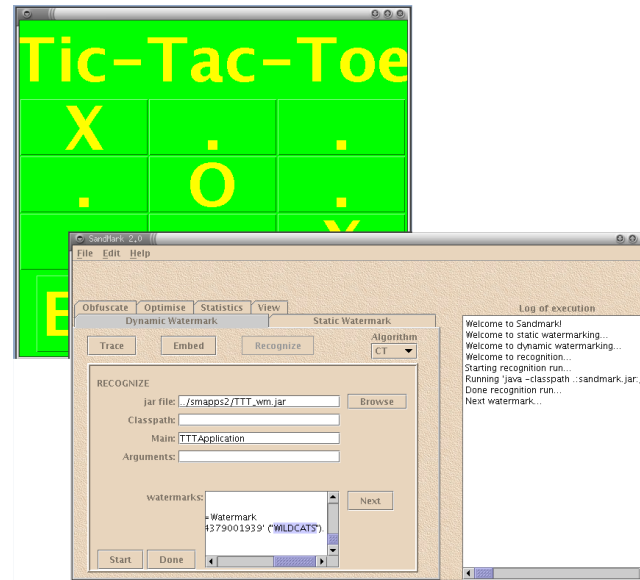
A Session with SANDMARK

SandMark
Birthmarking

NEW.jar



Recognize Watermark



- We extract the watermark to prove ownership.



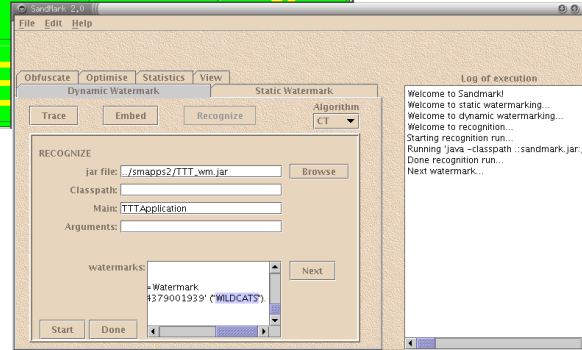
A Session with SANDMARK

SandMark
Birthmarking

NEW.jar



Recognize Watermark



- We extract the watermark to prove ownership.



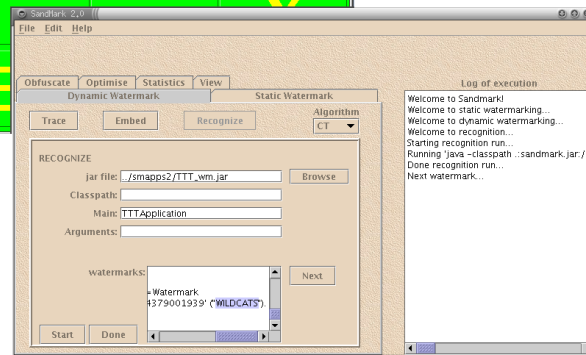
A Session with SANDMARK

SandMark
Birthmarking

NEW.jar



Recognize Watermark

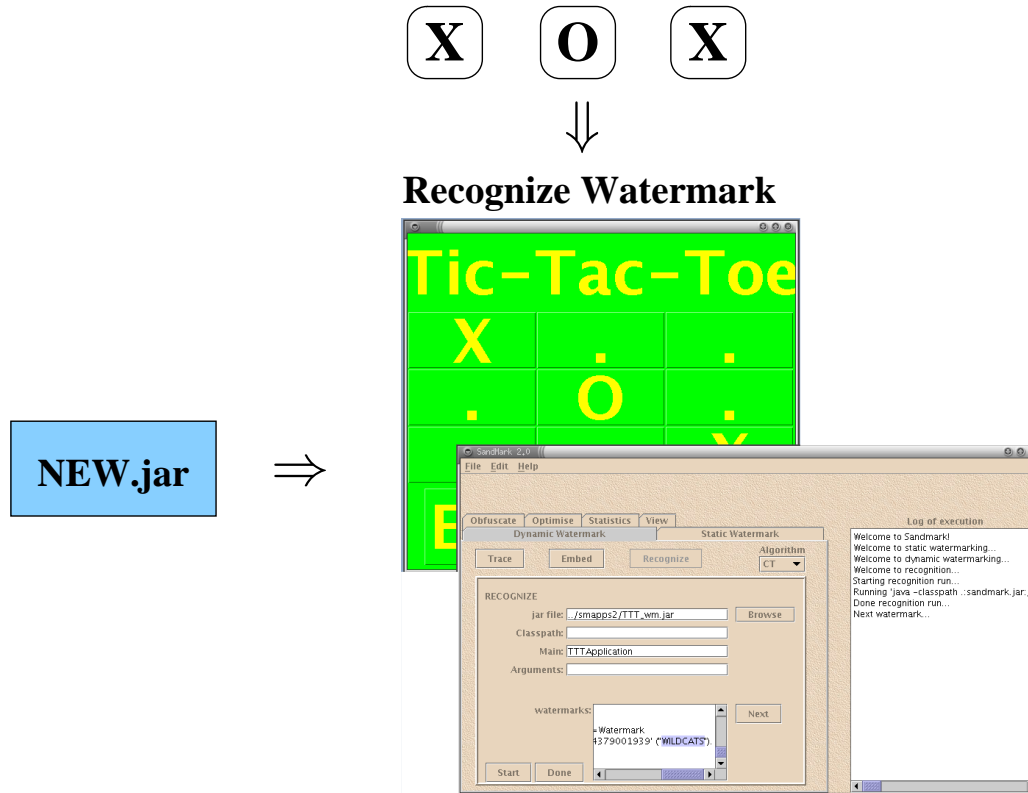


- We extract the watermark to prove ownership.



A Session with SANDMARK

SandMark
Birthmarking



- We extract the watermark to prove ownership.



A Session with SANDMARK

SandMark
Birthmarking

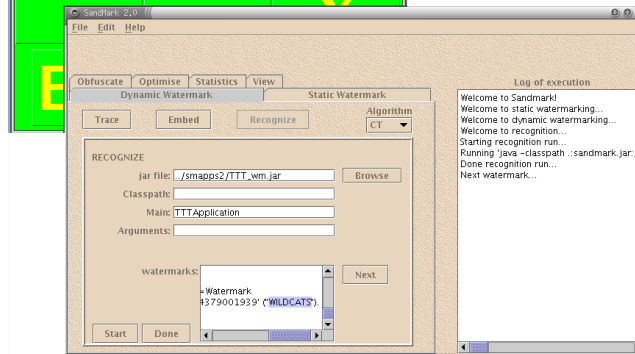
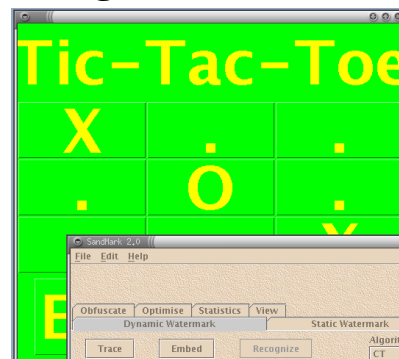
NEW.jar



X O X



Recognize Watermark



⇒ "WILDCATS"

- We extract the watermark to prove ownership.

A Session with SANDMARK

SandMark
Birthmarking

NEW.jar



Compare Bytecodes

Using Trivial Diff... 100% Minimum % 0.0 Trivial Diff Diff Stop

100.0% Main max Main min

100.0% Main Part Main Part

0.0% Main min Main min

Compute Static Statistics

Class: TTTApplication\$4

| Class name | Number of methods | Number of public methods | Method name | Call count | Method size (in bytes) | Thrower of catches | Exceptions |
|----------------|-------------------|--------------------------|-------------|------------|------------------------|--------------------|------------|
| TTTApplet | 8 | 2 | | | | | |
| TTTApplet\$1 | 3 | 3 | | | | | |
| TTTApplet\$2 | 2 | 2 | | | | | |
| TTTApplet\$3 | 2 | 2 | | | | | |
| TTTApplet\$4 | 2 | 2 | | | | | |
| TTTApplet\$5 | 2 | 2 | | | | | |
| TTTApplet\$6 | 2 | 2 | | | | | |
| TTTApplet\$7 | 2 | 2 | | | | | |
| TTTApplet\$8 | 2 | 2 | | | | | |
| TTTApplet\$9 | 2 | 2 | | | | | |
| TTTApplet\$10 | 2 | 2 | | | | | |
| TTTApplet\$11 | 2 | 2 | | | | | |
| TTTApplet\$12 | 2 | 2 | | | | | |
| TTTApplet\$13 | 2 | 2 | | | | | |
| TTTApplet\$14 | 2 | 2 | | | | | |
| TTTApplet\$15 | 2 | 2 | | | | | |
| TTTApplet\$16 | 2 | 2 | | | | | |
| TTTApplet\$17 | 2 | 2 | | | | | |
| TTTApplet\$18 | 2 | 2 | | | | | |
| TTTApplet\$19 | 2 | 2 | | | | | |
| TTTApplet\$20 | 2 | 2 | | | | | |
| TTTApplet\$21 | 2 | 2 | | | | | |
| TTTApplet\$22 | 2 | 2 | | | | | |
| TTTApplet\$23 | 2 | 2 | | | | | |
| TTTApplet\$24 | 2 | 2 | | | | | |
| TTTApplet\$25 | 2 | 2 | | | | | |
| TTTApplet\$26 | 2 | 2 | | | | | |
| TTTApplet\$27 | 2 | 2 | | | | | |
| TTTApplet\$28 | 2 | 2 | | | | | |
| TTTApplet\$29 | 2 | 2 | | | | | |
| TTTApplet\$30 | 2 | 2 | | | | | |
| TTTApplet\$31 | 2 | 2 | | | | | |
| TTTApplet\$32 | 2 | 2 | | | | | |
| TTTApplet\$33 | 2 | 2 | | | | | |
| TTTApplet\$34 | 2 | 2 | | | | | |
| TTTApplet\$35 | 2 | 2 | | | | | |
| TTTApplet\$36 | 2 | 2 | | | | | |
| TTTApplet\$37 | 2 | 2 | | | | | |
| TTTApplet\$38 | 2 | 2 | | | | | |
| TTTApplet\$39 | 2 | 2 | | | | | |
| TTTApplet\$40 | 2 | 2 | | | | | |
| TTTApplet\$41 | 2 | 2 | | | | | |
| TTTApplet\$42 | 2 | 2 | | | | | |
| TTTApplet\$43 | 2 | 2 | | | | | |
| TTTApplet\$44 | 2 | 2 | | | | | |
| TTTApplet\$45 | 2 | 2 | | | | | |
| TTTApplet\$46 | 2 | 2 | | | | | |
| TTTApplet\$47 | 2 | 2 | | | | | |
| TTTApplet\$48 | 2 | 2 | | | | | |
| TTTApplet\$49 | 2 | 2 | | | | | |
| TTTApplet\$50 | 2 | 2 | | | | | |
| TTTApplet\$51 | 2 | 2 | | | | | |
| TTTApplet\$52 | 2 | 2 | | | | | |
| TTTApplet\$53 | 2 | 2 | | | | | |
| TTTApplet\$54 | 2 | 2 | | | | | |
| TTTApplet\$55 | 2 | 2 | | | | | |
| TTTApplet\$56 | 2 | 2 | | | | | |
| TTTApplet\$57 | 2 | 2 | | | | | |
| TTTApplet\$58 | 2 | 2 | | | | | |
| TTTApplet\$59 | 2 | 2 | | | | | |
| TTTApplet\$60 | 2 | 2 | | | | | |
| TTTApplet\$61 | 2 | 2 | | | | | |
| TTTApplet\$62 | 2 | 2 | | | | | |
| TTTApplet\$63 | 2 | 2 | | | | | |
| TTTApplet\$64 | 2 | 2 | | | | | |
| TTTApplet\$65 | 2 | 2 | | | | | |
| TTTApplet\$66 | 2 | 2 | | | | | |
| TTTApplet\$67 | 2 | 2 | | | | | |
| TTTApplet\$68 | 2 | 2 | | | | | |
| TTTApplet\$69 | 2 | 2 | | | | | |
| TTTApplet\$70 | 2 | 2 | | | | | |
| TTTApplet\$71 | 2 | 2 | | | | | |
| TTTApplet\$72 | 2 | 2 | | | | | |
| TTTApplet\$73 | 2 | 2 | | | | | |
| TTTApplet\$74 | 2 | 2 | | | | | |
| TTTApplet\$75 | 2 | 2 | | | | | |
| TTTApplet\$76 | 2 | 2 | | | | | |
| TTTApplet\$77 | 2 | 2 | | | | | |
| TTTApplet\$78 | 2 | 2 | | | | | |
| TTTApplet\$79 | 2 | 2 | | | | | |
| TTTApplet\$80 | 2 | 2 | | | | | |
| TTTApplet\$81 | 2 | 2 | | | | | |
| TTTApplet\$82 | 2 | 2 | | | | | |
| TTTApplet\$83 | 2 | 2 | | | | | |
| TTTApplet\$84 | 2 | 2 | | | | | |
| TTTApplet\$85 | 2 | 2 | | | | | |
| TTTApplet\$86 | 2 | 2 | | | | | |
| TTTApplet\$87 | 2 | 2 | | | | | |
| TTTApplet\$88 | 2 | 2 | | | | | |
| TTTApplet\$89 | 2 | 2 | | | | | |
| TTTApplet\$90 | 2 | 2 | | | | | |
| TTTApplet\$91 | 2 | 2 | | | | | |
| TTTApplet\$92 | 2 | 2 | | | | | |
| TTTApplet\$93 | 2 | 2 | | | | | |
| TTTApplet\$94 | 2 | 2 | | | | | |
| TTTApplet\$95 | 2 | 2 | | | | | |
| TTTApplet\$96 | 2 | 2 | | | | | |
| TTTApplet\$97 | 2 | 2 | | | | | |
| TTTApplet\$98 | 2 | 2 | | | | | |
| TTTApplet\$99 | 2 | 2 | | | | | |
| TTTApplet\$100 | 2 | 2 | | | | | |
| TTTApplet\$101 | 2 | 2 | | | | | |
| TTTApplet\$102 | 2 | 2 | | | | | |
| TTTApplet\$103 | 2 | 2 | | | | | |
| TTTApplet\$104 | 2 | 2 | | | | | |
| TTTApplet\$105 | 2 | 2 | | | | | |
| TTTApplet\$106 | 2 | 2 | | | | | |
| TTTApplet\$107 | 2 | 2 | | | | | |
| TTTApplet\$108 | 2 | 2 | | | | | |
| TTTApplet\$109 | 2 | 2 | | | | | |
| TTTApplet\$110 | 2 | 2 | | | | | |
| TTTApplet\$111 | 2 | 2 | | | | | |
| TTTApplet\$112 | 2 | 2 | | | | | |
| TTTApplet\$113 | 2 | 2 | | | | | |
| TTTApplet\$114 | 2 | 2 | | | | | |
| TTTApplet\$115 | 2 | 2 | | | | | |
| TTTApplet\$116 | 2 | 2 | | | | | |
| TTTApplet\$117 | 2 | 2 | | | | | |
| TTTApplet\$118 | 2 | 2 | | | | | |
| TTTApplet\$119 | 2 | 2 | | | | | |
| TTTApplet\$120 | 2 | 2 | | | | | |
| TTTApplet\$121 | 2 | 2 | | | | | |
| TTTApplet\$122 | 2 | 2 | | | | | |
| TTTApplet\$123 | 2 | 2 | | | | | |
| TTTApplet\$124 | 2 | 2 | | | | | |
| TTTApplet\$125 | 2 | 2 | | | | | |
| TTTApplet\$126 | 2 | 2 | | | | | |
| TTTApplet\$127 | 2 | 2 | | | | | |
| TTTApplet\$128 | 2 | 2 | | | | | |
| TTTApplet\$129 | 2 | 2 | | | | | |
| TTTApplet\$130 | 2 | 2 | | | | | |
| TTTApplet\$131 | 2 | 2 | | | | | |
| TTTApplet\$132 | 2 | 2 | | | | | |
| TTTApplet\$133 | 2 | 2 | | | | | |
| TTTApplet\$134 | 2 | 2 | | | | | |
| TTTApplet\$135 | 2 | 2 | | | | | |
| TTTApplet\$136 | 2 | 2 | | | | | |
| TTTApplet\$137 | 2 | 2 | | | | | |
| TTTApplet\$138 | 2 | 2 | | | | | |
| TTTApplet\$139 | 2 | 2 | | | | | |
| TTTApplet\$140 | 2 | 2 | | | | | |
| TTTApplet\$141 | 2 | 2 | | | | | |
| TTTApplet\$142 | 2 | 2 | | | | | |
| TTTApplet\$143 | 2 | 2 | | | | | |
| TTTApplet\$144 | 2 | 2 | | | | | |
| TTTApplet\$145 | 2 | 2 | | | | | |
| TTTApplet\$146 | 2 | 2 | | | | | |
| TTTApplet\$147 | 2 | 2 | | | | | |
| TTTApplet\$148 | 2 | 2 | | | | | |
| TTTApplet\$149 | 2 | 2 | | | | | |
| TTTApplet\$150 | 2 | 2 | | | | | |
| TTTApplet\$151 | 2 | 2 | | | | | |
| TTTApplet\$152 | 2 | 2 | | | | | |
| TTTApplet\$153 | 2 | 2 | | | | | |
| TTTApplet\$154 | 2 | 2 | | | | | |
| TTTApplet\$155 | 2 | 2 | | | | | |
| TTTApplet\$156 | 2 | 2 | | | | | |
| TTTApplet\$157 | 2 | 2 | | | | | |
| TTTApplet\$158 | 2 | 2 | | | | | |
| TTTApplet\$159 | 2 | 2 | | | | | |
| TTTApplet\$160 | 2 | 2 | | | | | |
| TTTApplet\$161 | 2 | 2 | | | | | |
| TTTApplet\$162 | 2 | 2 | | | | | |
| TTTApplet\$163 | 2 | 2 | | | | | |
| TTTApplet\$164 | 2 | 2 | | | | | |
| TTTApplet\$165 | 2 | 2 | | | | | |
| TTTApplet\$166 | 2 | 2 | | | | | |
| TTTApplet\$167 | 2 | 2 | | | | | |
| TTTApplet\$168 | 2 | 2 | | | | | |
| TTTApplet\$169 | 2 | 2 | | | | | |
| TTTApplet\$170 | 2 | 2 | | | | | |
| TTTApplet\$171 | 2 | 2 | | | | | |
| TTTApplet\$172 | 2 | 2 | | | | | |
| TTTApplet\$173 | 2 | 2 | | | | | |
| TTTApplet\$174 | 2 | 2 | | | | | |
| TTTApplet\$175 | 2 | 2 | | | | | |
| TTTApplet\$176 | 2 | 2 | | | | | |
| TTTApplet\$177 | 2 | 2 | | | | | |
| TTTApplet\$178 | 2 | 2 | | | | | |
| TTTApplet\$179 | 2 | 2 | | | | | |
| TTTApplet\$180 | 2 | 2 | | | | | |
| TTTApplet\$181 | 2 | 2 | | | | | |
| TTTApplet\$182 | 2 | 2 | | | | | |
| TTTApplet\$183 | 2 | 2 | | | | | |
| TTTApplet\$184 | 2 | 2 | | | | | |
| TTTApplet\$185 | 2 | 2 | | | | | |
| TTTApplet\$186 | 2 | 2 | | | | | |
| TTTApplet\$187 | 2 | 2 | | | | | |
| TTTApplet\$188 | 2 | 2 | | | | | |
| TTTApplet\$189 | 2 | 2 | | | | | |
| TTTApplet\$190 | 2 | 2 | | | | | |
| TTTApplet\$191 | 2 | 2 | | | | | |
| TTTApplet\$192 | 2 | 2 | | | | | |
| TTTApplet\$193 | 2 | 2 | | | | | |
| TTTApplet\$194 | 2 | 2 | | | | | |
| TTTApplet\$195 | 2 | 2 | | | | | |
| TTTApplet\$196 | 2 | 2 | | | | | |
| TTTApplet\$197 | 2 | 2 | | | | | |
| TTTApplet\$198 | 2 | 2 | | | | | |
| TTTApplet\$199 | 2 | 2 | | | | | |
| TTTApplet\$200 | 2 | 2 | | | | | |
| TTTApplet\$201 | 2 | 2 | | | | | |
| TTTApplet\$202 | 2 | 2 | | | | | |
| TTTApplet\$203 | 2 | 2 | | | | | |
| TTTApplet\$204 | 2 | 2 | | | | | |
| TTTApplet\$205 | 2 | 2 | | | | | |
| TTTApplet\$206 | 2 | 2 | | | | | |
| TTTApplet\$207 | 2 | 2 | | | | | |
| TTTApplet\$208 | 2 | 2 | | | | | |
| TTTApplet\$209 | 2 | 2 | | | | | |
| TTTApplet\$210 | 2 | 2 | | | | | |
| TTTApplet\$211 | 2 | 2 | | | | | |
| TTTApplet\$212 | 2 | 2 | | | | | |
| TTTApplet\$213 | 2 | 2 | | | | | |
| TTTApplet\$214 | 2 | 2 | | | | | |
| TTTApplet\$215 | 2 | 2 | | | | | |
| TTTApplet\$216 | 2 | 2 | | | | | |
| TTTApplet\$217 | 2 | 2 | | | | | |
| TTTApplet\$218 | 2 | 2 | | | | | |
| TTTApplet\$219 | 2 | 2 | | | | | |
| TTTApplet\$220 | 2 | 2 | | | | | |
| TTTApplet\$221 | 2 | 2 | | | | | |
| TTTApplet\$222 | 2 | 2 | | | | | |
| TTTApplet\$223 | 2 | 2 | | | | | |
| TTTApplet\$224 | 2 | 2 | | | | | |
| TTTApplet\$225 | 2 | 2 | | | | | |
| TTTApplet\$226 | 2 | 2 | | | | | |
| TTTApplet\$227 | 2 | 2 | | | | | |
| TTTApplet\$228 | 2 | 2 | | | | | |
| TTTApplet\$229 | 2 | 2 | | | | | |
| TTTApplet\$230 | 2 | 2 | | | | | |
| TTTApplet\$231 | 2 | 2 | | | | | |
| TTTApplet\$232 | 2 | 2 | | | | | |
| TTTApplet\$233 | 2 | 2 | | | | | |
| TTTApplet\$234 | 2 | 2 | | | | | |
| TTTApplet\$235 | 2 | 2 | | | | | |
| TTTApplet\$236 | 2 | 2 | | | | | |
| TTTApplet\$237 | 2 | 2 | | | | | |
| TTTApplet\$238 | 2 | 2 | | | | | |
| TTTApplet\$239 | 2 | 2 | | | | | |
| TTTApplet\$240 | 2 | 2 | | | | | |
| TTTApplet\$241 | 2 | 2 | | | | | |
| TTTApplet\$242 | 2 | 2 | | | | | |
| TTTApplet\$243 | 2 | 2 | | | | | |
| TTTApplet\$244 | 2 | 2 | | | | | |
| TTTApplet\$245 | 2 | 2 | | | | | |
| TTTApplet\$246 | 2 | 2 | | | | | |
| TT | | | | | | | |



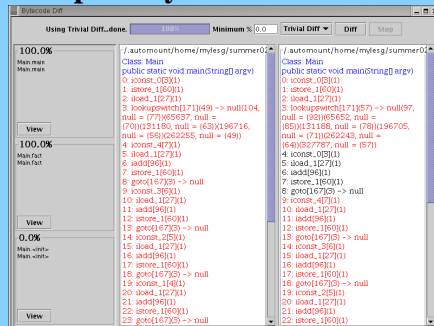
A Session with SANDMARK

SandMark
Birthmarking

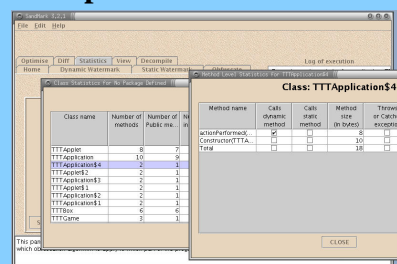
NEW.jar



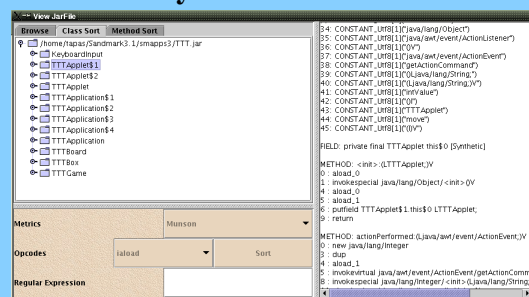
Compare Bytecodes



Compute Static Statistics

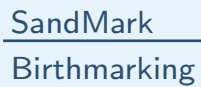


View/Sort Bytecodes



⇒ Watermark located?

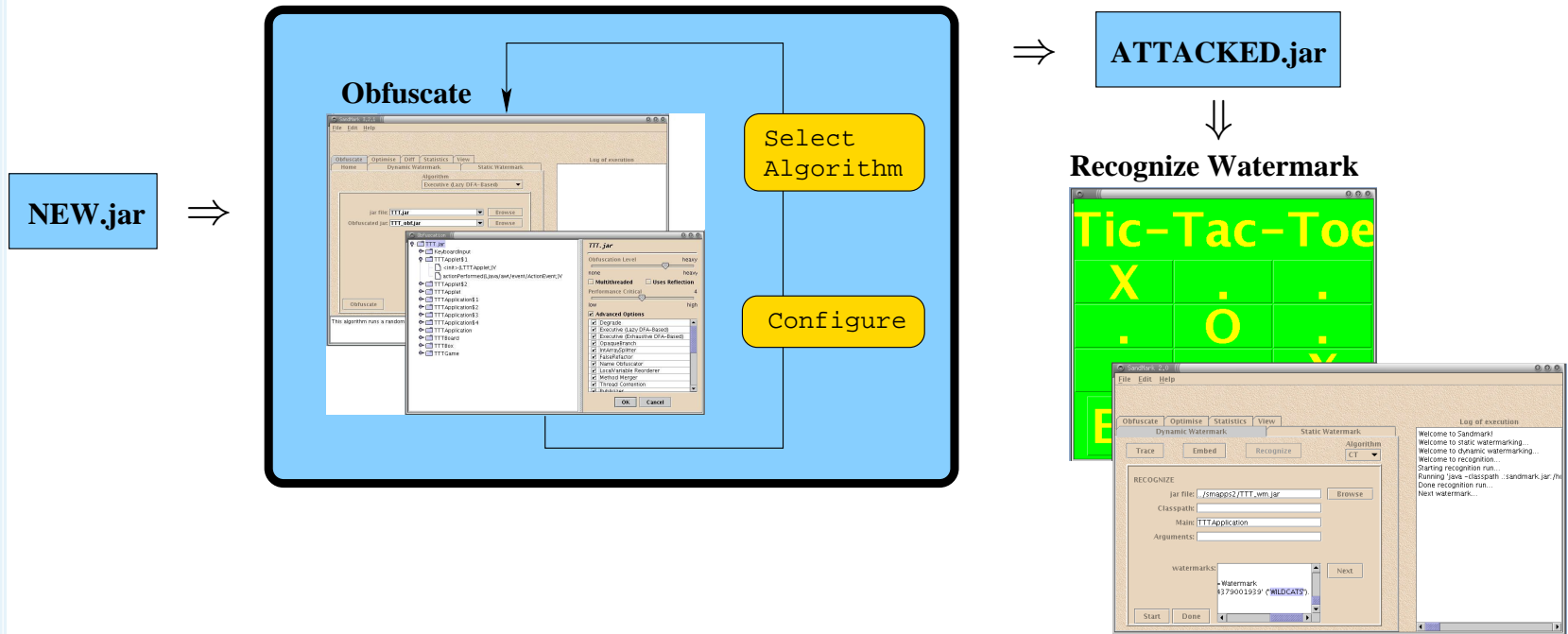
- To simulate a manual attack we examine the obfuscated/watermarked program using various static analysis tools.



- To simulate an **automatic attack** we use SANDMARK's obfuscators (*"SoftStir"*) to attack the watermark.

A Session with SANDMARK

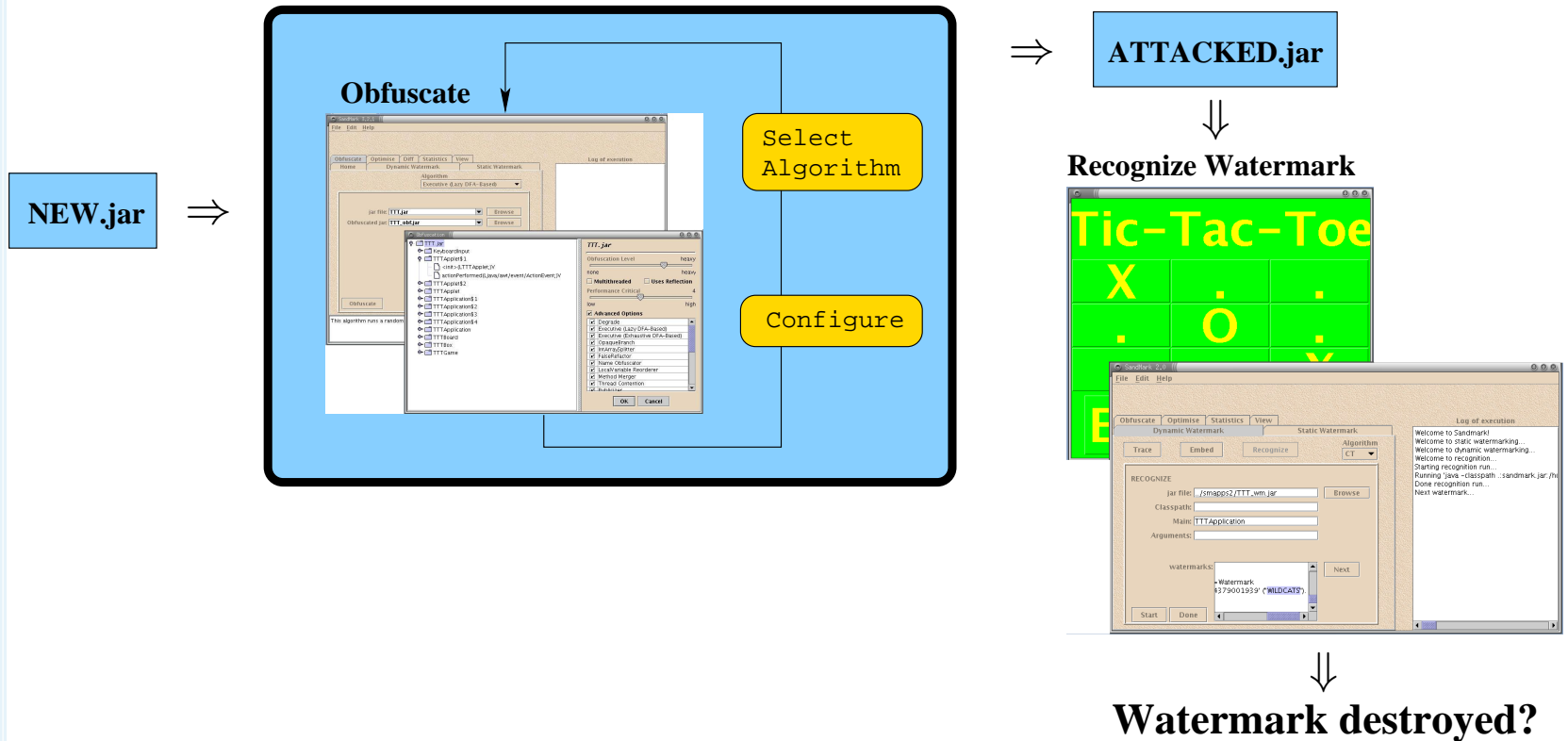
SandMark
Birthmarking



- To simulate an automatic attack we use SANDMARK's obfuscators (*"SoftStir"*) to attack the watermark.

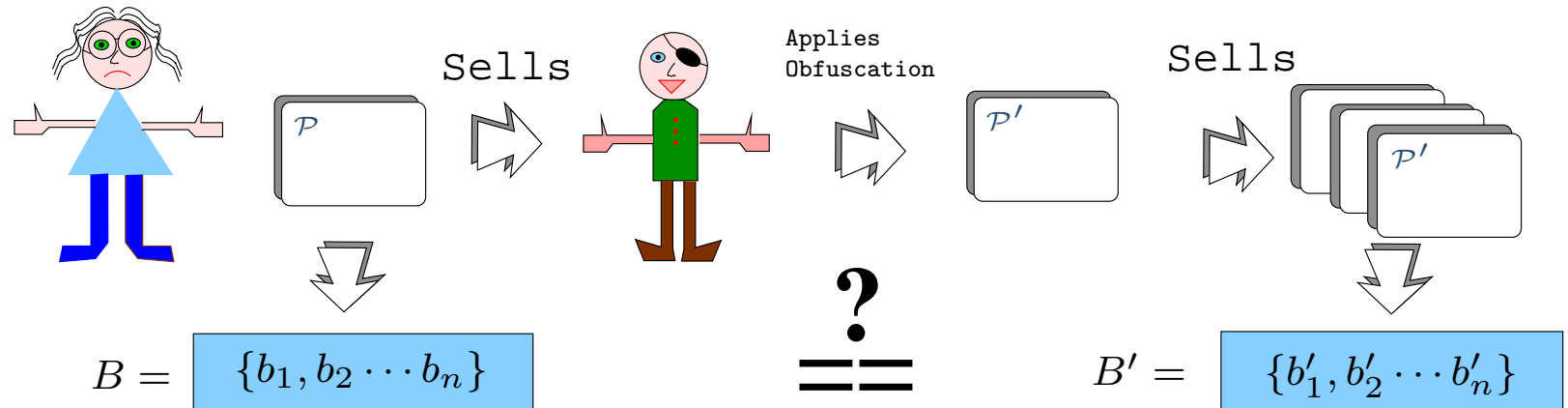
A Session with SANDMARK

SandMark
Birthmarking



- To simulate an **automatic attack** we use SANDMARK's obfuscators (*"SoftStir"*) to attack the watermark.

Work in Progress — Birthmarking



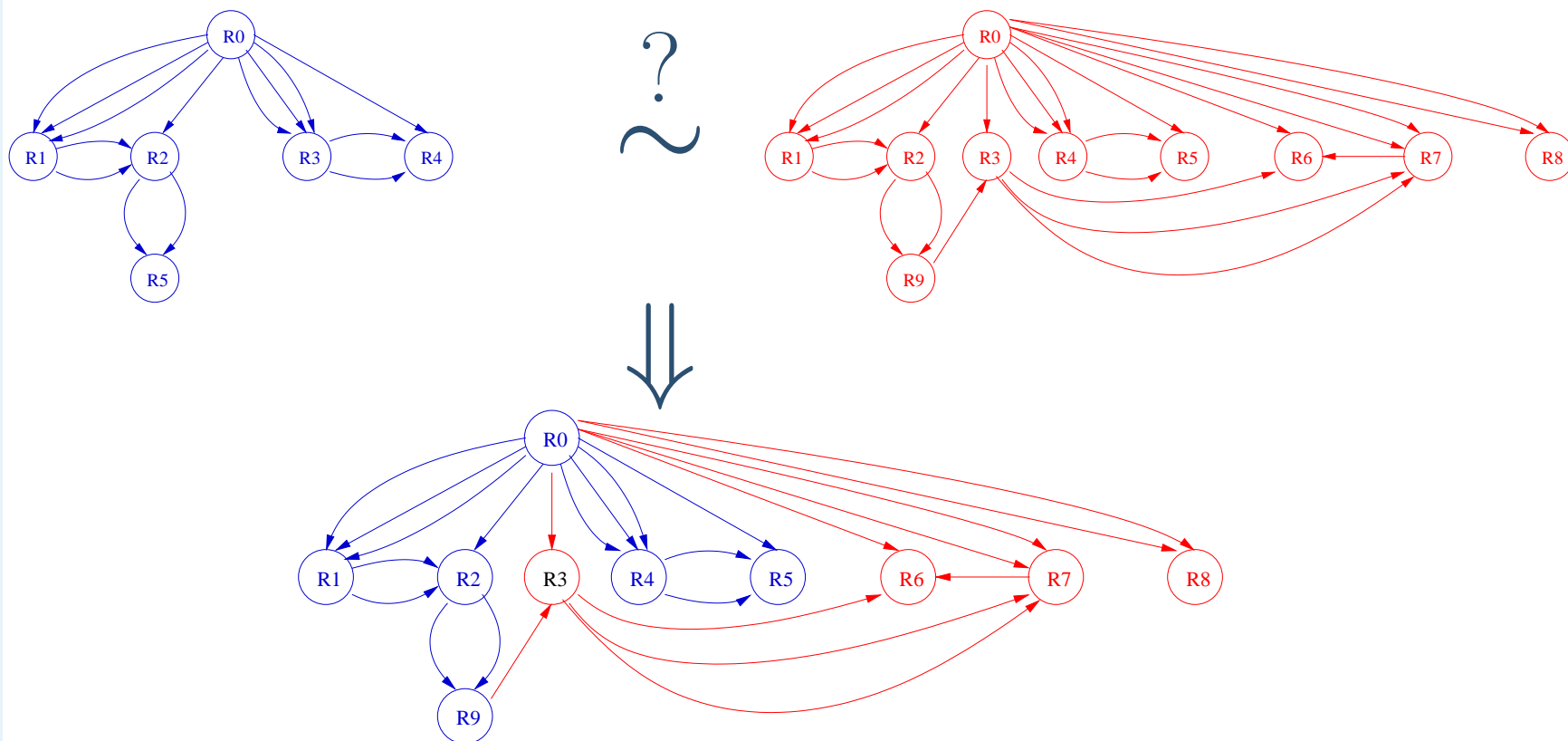
- A **software birthmark** is a set of unique characteristics of a program.
- Birthmarks are used to detect software theft.
- Characteristics are chosen to be invariant under semantics-preserving transformations.

Myles & Collberg, 7th Information Security Conference (ISC'04)



Birthmarking — Whole-Program Paths

- We compute dynamic birthmarks using **Whole-Program Paths** and **graph-similarity metrics**.



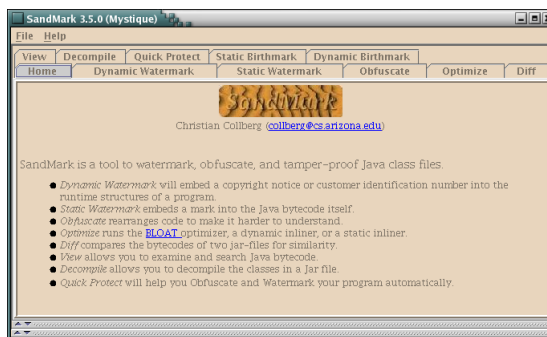


Conclusion

Summary

Conclusion

- Attention to details is important. The attacker will go for the easy target first.
- Implementations that cover *all* the corners are an important part of the evaluation.
- Are dynamic attacks harder to defend against?
- The goal is to make the attacker have to consider the entire program, not just “interesting pieces.” Dynamic attacks are useful to chunk up the program.



- Download from sandmark.cs.arizona.edu.