# Taxonomy of Applications for Program Obfuscation

## Detailed View

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

# Outline

## 1 Attack classification

- Knowledge Extraction
- Attacks on Integrity
- Tampering Attacks
- Input-Output Attacks

# Outline

## 1 Attack classification

- Knowledge Extraction
- Attacks on Integrity
- Tampering Attacks
- Input-Output Attacks

## 2 Basic Examples

- Cryptosystems
- Program Distribution
- Mobile Agents

# Outline

# Actions against program

## How can adversary act with program?

# Actions against program

## How can adversary act with program?

- ⇨ Study program (extracting knowledge)
- ⇨ Decompose program (reusing parts of it)
- ⇨ Change program behavior (tampering)
- ⇨ Providing wrong inputs and intercepting all connections

# Knowledge Extraction

**What examples of knowledge extraction attacks can you suggest?**

# Knowledge Extraction

**What examples of knowledge extraction attacks can you suggest?**

- ⇨ Viruses recognition
- ⇨ Keys extraction
- ⇨ Vulnerability search
- ⇨ Technology understanding

# Attacks on Integrity

**What examples of decomposition attacks can you suggest?**

# Attacks on Integrity

## What examples of decomposition attacks can you suggest?

- ⇨ Competitor attack
- ⇨ Extraction of signing procedure
- ⇨ Extraction of encoding/decoding procedures
- ⇨ Watermarks search (and deletion)
- ⇨ Patching and adding functionality
- ⇨ Password stealing

# Tampering Attacks

## What examples of tampering can you suggest?

# Tampering Attacks

## What examples of tampering can you suggest?

- ⇨ Constants changing
- ⇨ State changing
- ⇨ Rearranging sequence of procedures evaluation
- ⇨ Deleting any constraints (attacks on demo-versions)

# Input-Output Attacks

**What examples of input-output attacks can you suggest?**

# Input-Output Attacks

## What examples of input-output attacks can you suggest?

- ⇨ Providing inputs on behalf of third party
- ⇨ Delay, delete and change output messages of the program
- ⇨ Force program to output messages we interested in
- ⇨ Input capturing

# New Public-Key Cryptosystems

**General idea**: given a private-key (symmetric) cryptosystem publish obfuscated encryption algorithm $O(E_k)$ as a public key.

Analysis:

- ⇨ We must be sure that key extraction of $O(E_k)$ is computationally hard
- ⇨ Moreover, rewriting $O(E_k)$ to any efficient program computing $D_k$ must be computationally hard
- ⇨ **Conclusion:** starting symmetric cryptosystem should have sufficient difference in encrypting and decrypting algorithms

# DES/AES Obfuscation

**Task:** given a fixed key to write a program computing DES (AES) encoding with this key in a form such that key extraction is computationally hard.

Analysis:

- ⇨ Very good benchmark example of constant hiding
- ⇨ This problem was approached in several papers
- ⇨ No clear straightforward application was presented
- ⇨ No security proof for current obfuscation of block cyphers

# Constructing Homomorphic Encryption

Given good enough obfuscator it's easy to construct a homomorphic encryption.

**Any ideas how to do this?**

# Constructing Homomorphic Encryption

Given good enough obfuscator it's easy to construct a homomorphic encryption.

**Any ideas how to do this?**

**Construction:** as such homomorphic encryption we can take just any public key cryptosystem
Let us consider the following programs $P$ (and $Q$):

> Input: $E(x), E(y)$
> Program algorithm: using private key decrypt $x$ and $y$, compute $x + y$ (respectively $xy$ in $Q$), then encrypt it.
> Output: $E(x+y)$ (respectively, $E(xy)$)

If we are able to obfuscate $P$ and $Q$ in the way that extracting private key and intermediate results ($x$ and $y$) is computationally hard than we are done!

# Linking Problem

The heart of program protection is the following (linking) problem:

Given a program which is a collection of modules (algorithms) and a set of rules in what order these procedures should be evaluated.

**Task:** to protect these rules and make difficult to change them or use only one specific procedure.

**Observation:** this protection is different to black-box security.

# Protection of IF Operator

The protection of IF operator (is something very similar to linking problem):

Given a program which somewhere in its code contains the following construction:

```
If (some condition) then
        do something important
        else do nothing (or some not interesting things)
```

**Adversary attack:** destroy this IF operator i.e. get a program with unconditional important module.

**Observation:** this case is also different to black-box security.

# Examples of Linking Problem/IF Protection

**What examples of linking problem/IF protection can you suggest?**

# Examples of Linking Problem/IF Protection

## What examples of linking problem/IF protection can you suggest?

- License management (shareware) i.e. trial time constraint
- Bounded functionality. E.g. restricted ability to edit/print/reuse something or producing only part of required work.
- Bounded number of runs
- Restrictions on input
- Password access to some additional functionalities

# Network Monitoring Systems

First interesting example of mobile agent needed protection is network monitoring and management systems.

**We have:** a huge network (consisting of **nodes**), and a monitoring agent installed on each node.

Some observations:

- ⇨ Agents interacts with their hosts
- ⇨ Agents interacts with central (the only trusted) node. We can call it control center.
- ⇨ We can't protect agents against just deleting (uninstalling them)
- ⇨ We want to protect the "state" of agents and their proper execution

# Buying Agent

Another important example is buying agent.

**What do we have:** a set of “sellers” with installed buying agents. These agents have a task to purchase a specific good if some conditions (usually on price) holds

Aspects:

- ⇨ Buying agents have keys to the credit card or electronic money.
- ⇨ Adversary is always able to delete an agent.
- ⇨ Agents owner wants to prevent key’s extraction and changing conditions of purchase or even buying wrong good.

# Ideas of Applications

## Your ideas of applications?

# Ideas of Applications

## Your ideas of applications?

- ⇨ Diversity producing
- ⇨ Random generators for cryptosystems
- ⇨ Guaranteed slowdown
- ⇨ “Evil application”: viruses hiding

# Connections with other topics

## Other research topics interested in obfuscation include:

- ⇨ Digital Watermarks

  **Obfuscation task:** to make watermark search and deletion more difficult.

- ⇨ Steganography

  **Obfuscation task:** to make steganography algorithms unclear and hence increase lifetime of information hiding algorithms.

- ⇨ Tamper resistant devices

  **Obfuscation task:** software implementation of such devices would be much easier to distribute than any hardware solutions.

# Summary

- ⇨ We provide some classification of attacks (knowledge extracting, tampering, decomposition and input-output attacks)
- ⇨ We study some applications in cryptosystem design, program constraints protection and mobile agent technology.
- ⇨ We mention some connections to other research fields (steganography, watermarking, tamper-resistant devices).

# Summary

- We provide some classification of attacks (knowledge extracting, tampering, decomposition and input-output attacks)
- We study some applications in cryptosystem design, program constraints protection and mobile agent technology.
- We mention some connections to other research fields (steganography, watermarking, tamper-resistant devices).

Question Time!

# Not Covered by the Talk

- Shareware protection schemes
- Pdf protection
- Multiple modification of the program
- Obfuscating of key generator algorithm
- Additional functionality hiding
- Unrecognizable properties
- Digital Rights Management
- Sampling algorithms [barak]
- Simulation of Clipper Chip [Murayama]
- iTunes
- Windows RMS LockBox
- Xbox
- Search Engine Algorithm Hiding
- Internet cafe (trusted user, but untrusted software and operating system)

# What to read?

On this topic (applications of obfuscation) there is no single good reference.

On the other hand any paper on obfuscation provide vogue description of some applications.