Different Concepts for Program Obfuscation

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

Outline

1 Applications

- Classical Cryptography
- Software Protection
- Mobile Agents Technology
- Other

Outline

Outline

Applications for Obfuscation

Today: only short overview of applications

In details: Lecture 4 - “Applications for Obfuscation”

Classical Cryptography

What applications in cryptography can we imagine?

Classical Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem $\longrightarrow$ Public key cryptosystem
  It was mentioned even in famous Diffie-Hellman paper.

Classical Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem $\longrightarrow$ Public key cryptosystem
  It was mentioned even in famous Diffie-Hellman paper.
- Homomorphic encoding

Classical Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem $\longrightarrow$ Public key cryptosystem
  It was mentioned even in famous Diffie-Hellman paper.
- Homomorphic encoding
- Random oracles removing

Software Protection

Situation: we distribute (sell) software products.

Question: Threats and applications you see?

Software Protection

Situation: we distribute (sell) software products.

Question: Threats and applications you see?

⇨ Competitors threat (reusing your code)

Software Protection

Situation: we distribute (sell) software products.

Question: Threats and applications you see?

- ⇨ Competitors threat (reusing your code)
- ⇨ Intelligent tampering (changing parameters)

Software Protection

Situation: we distribute (sell) software products.

Question: Threats and applications you see?

- ⇨ Competitors threat (reusing your code)
- ⇨ Intelligent tampering (changing parameters)
- ⇨ Threat of functionality changes (protection demo-versions)

Software Protection

Situation: we distribute (sell) software products.

Question: Threats and applications you see?

- ⇒ Competitors threat (reusing your code)
- ⇒ Intelligent tampering (changing parameters)
- ⇒ Threat of functionality changes (protection demo-versions)
- ⇒ Watermarks protection

Mobile Agents Technology

Situation: we distribute programs for our needs.

Question: Threats and applications you see?

Mobile Agents Technology

Situation: we distribute programs for our needs.

Question: Threats and applications you see?

⇨ Privacy: e.g. internet-distributed computation

Mobile Agents Technology

Situation: we distribute programs for our needs.

Question: Threats and applications you see?

- ⇨ Privacy: e.g. internet-distributed computation
- ⇨ Keys protection: buying agents.

Mobile Agents Technology

Situation: we distribute programs for our needs.

Question: Threats and applications you see?

- Privacy: e.g. internet-distributed computation
- Keys protection: buying agents.
- Intelligent tampering

Other applications

Question: More applications?

Other applications

Question: More applications?

Yes!

Other applications

Question: More applications?

Yes!

- Virus development

Other applications

Question: More applications?

Yes!

- Virus development
- Watermark attacks

An Obfuscator

In details: Lecture 2 - “Obfuscating transformations”

An Obfuscator

In details: Lecture 2 - “Obfuscating transformations”

- Functionality preserving
- Increase of code size, time & space requirements are restricted (usually by constant factor)
- Obfuscated program is not readable (not understandable)

Classification of obfuscating transformations

What can we obfuscate in the program?

Classification of obfuscating transformations

What can we obfuscate in the program?

- Layout transformations

  Change formatting information

Classification of obfuscating transformations

What can we obfuscate in the program?

- Layout transformations

  Change formatting information

- Control flow transformations

  Alter control program and computation

Classification of obfuscating transformations

What can we obfuscate in the program?

- Layout transformations
  Change formatting information
- Control flow transformations
  Alter control program and computation
- Aggregation transformation
  Refactor program using aggregation methods

Classification of obfuscating transformations

What can we obfuscate in the program?

- Layout transformations
  Change formatting information
- Control flow transformations
  Alter control program and computation
- Aggregation transformation
  Refactor program using aggregation methods
- Data transformations
  Use information encoding

Quality of Obfuscation

How good our obfuscation is?

Quality of Obfuscation

How good our obfuscation is?

Strength can be measured by:

- Potency

  $\frac{E(P')}{E(P)} - 1$
- Resilience

  Trivial, weak, strong, full, one-way
- Cost

  Free, cheap, costly, expensive
- Stealthy

Ideal case

What do we want to get?

Ideal case

What do we want to get?

```
mysterious.c

int mysterious(imt x, in

{
  int z;
  z=x+y;
  returnz z;
}
```

Ideal case

What do we want to get?

Very limited information:

- ⇨ input-output behavior
- ⇨ running time

Ana and BAna

We are interested in 2 types of polynomial-time analyzers:

- Ana is a source-code analyzer that can read the program.

$$Ana(P)$$

- BAna is a black-box analyzer that only queries the program as an oracle.

$$BAna^{P}(time(P))$$

Ana and BAna

We are interested in 2 types of polynomial-time analyzers:

- Ana is a source-code analyzer that can read the program.

$$Ana(P)$$

- BAna is a black-box analyzer that only queries the program as an oracle.

$$BAna^P(time(P))$$

Black-Box security

Ana can't get more information than BAna could

Property Hiding

How to formalize property hiding?

Property Hiding

How to formalize property hiding?

Instance: two families of programs Π_1 and Π_2

Adversary task: given a program $P \in \Pi_1 \cup \Pi_2$ to decide whether $P \in \Pi_1$ or $P \in \Pi_2$.

Property Hiding

How to formalize property hiding?

Instance: two families of programs Π_1 and Π_2

Adversary task: given a program $P \in \Pi_1 \cup \Pi_2$ to decide whether $P \in \Pi_1$ or $P \in \Pi_2$.

Desirable protection: make adversary task as difficult as well-known computationally hard problem is.

Constant Hiding

How to formalize constant hiding?

Constant Hiding

How to formalize constant hiding?

Instance: family of programs

$$\Pi = \{P | P \; computes \; f(s,x); \; s \in S\}$$

Adversary task: given a program $P \in \Pi$ to compute parameter s.

Constant Hiding

How to formalize constant hiding?

Instance: family of programs

$$\Pi = \{P | P \text{ } computes \text{ } f(s, x); \; s \in S\}$$

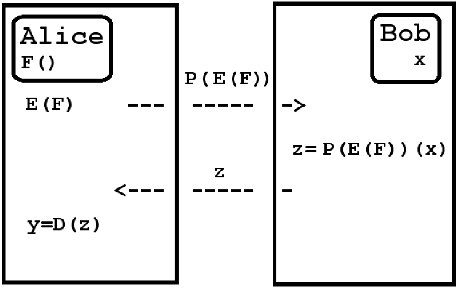
Adversary task: given a program $P \in \Pi$ to compute parameter s.

Desirable protection: make adversary task as difficult as well-known computationally hard problem is.

Encrypted Computation

More details: Lecture 5 - “Basic Complexity Results”

What is encrypted computation?



Basic task: keep *F* unknown to Bob.

Extendings of Encrypted Computation

Additional tasks of encrypted computation model:

- Move difficult computations to Bob

  D is easier than F

- Reduce communication complexity

  In the case $sizeof((F(x)) \ll sizeof(x)$. Example: x is database

- Keep x secret from Alice

Currently studied representations

Obfuscating techniques development depends on used program representation

So what sort of programs are we going to protect?

Currently studied representations

Obfuscating techniques development depends on used program representation

So what sort of programs are we going to protect?

- ⇨ Turing Machines / Circuits (function computing)

Currently studied representations

Obfuscating techniques development depends on used program representation

So what sort of programs are we going to protect?

- Turing Machines / Circuits (function computing)
- C++/Java code

Currently studied representations

Obfuscating techniques development depends on used program representation

So what sort of programs are we going to protect?

- Turing Machines / Circuits (function computing)
- C++/Java code
- Assembler code

Currently studied representations

Obfuscating techniques development depends on used program representation

So what sort of programs are we going to protect?

- Turing Machines / Circuits (function computing)
- C++/Java code
- Assembler code
- Rational function / Matrix representation

Search for other representations

Is it enough?

Search for other representations

Is it enough?

Not! New models should contain:

Search for other representations

Is it enough?

Not! New models should contain:

- Current state of the program.

Search for other representations

Is it enough?

Not! New models should contain:

- Current state of the program.
- Self-modifiable code

Search for other representations

Is it enough?

Not! New models should contain:

- Current state of the program.
- Self-modifiable code
- Notion of computation trace.

Search for other representations

Is it enough?

Not! New models should contain:

- Current state of the program.
- Self-modifiable code
- Notion of computation trace.
- Other formalizations for functionality preserving.

Adversary

What should we specify about adversary?

Adversary

What should we specify about adversary?

- Adversary knowledge about protected program

Adversary

What should we specify about adversary?

- Adversary knowledge about protected program
  - Member of family

Adversary

What should we specify about adversary?

- Adversary knowledge about protected program
  - Member of family
  - Known function – unknown parameters (data) and state.

Adversary

What should we specify about adversary?

- Adversary knowledge about protected program
  - Member of family
  - Known function – unknown parameters (data) and state.
- Adversary task (attack)

Adversary

What should we specify about adversary?

- Adversary knowledge about protected program
  - Member of family
  - Known function – unknown parameters (data) and state.
- Adversary task (attack)
  - Classification follows in Lecture 4.

Potential for Obfuscation

Is it possible to protect **every** program?

Potential for Obfuscation

Is it possible to protect every program?

- How to measure potential of obfuscation?
  - Learnability: black-box learnable functions are impossible to obfuscate.
- What couldn't be protected?
  - Input-Outbut behaviour
  - Traces

Network model

What are interesting network extentions of the model?

Network model

What are interesting network extentions of the model?

- Many programs cooperate
- Programs are migrating
- Programs can be recharged
- Different sources for inputs (outside connections)

Summary

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.

Summary

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Basic approaches: obfuscating transformations, black-box security, encrypted computation.

Summary

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Basic approaches: obfuscating transformations, black-box security, encrypted computation.
- Further aspects of the model: program representation, state protection, adversary description, functionality preserving.

Summary

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Basic approaches: obfuscating transformations, black-box security, encrypted computation.
- Further aspects of the model: program representation, state protection, adversary description, functionality preserving.

Summary

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Basic approaches: obfuscating transformations, black-box security, encrypted computation.
- Further aspects of the model: program representation, state protection, adversary description, functionality preserving.

Question Time!

Not covered by the talk

Gray & white security
Approximate obfuscators
Operations on obfuscated code
Adversary success
Nondeterministic nature
Modifying algorithm vs. modifying code
Complexity of deobfuscation: NP, NP-hard, undecidable, one-way...
Obfuscation on specification level
Wroblewsky model

For Further Reading

Yury Lifshits
Program Obfuscation. A Survey [in Russian]
http://logic.pdmi.ras.ru/~yura/of/survey1.pdf

Luis F.G. Sarmenta
Protecting Programs from Hostile Environments
http://bayanihancomputing.net/papers/ae/ae.ps