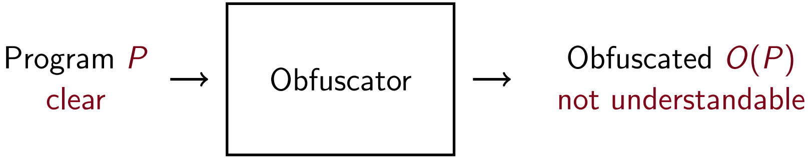
Introduction to Obfuscation. Black-box Security

Yury Lifshits

Steklov Institute of Mathematics, St.Petersburg, Russia
yura@logic.pdmi.ras.ru

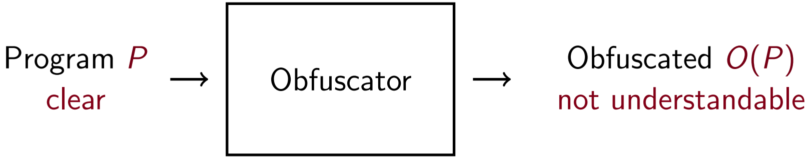
Tartu University
13/03/2006

Idea of Obfuscation



Program P clear → Obfuscator → Obfuscated $O(P)$ not understandable

Idea of Obfuscation



Program P
clear
$\longrightarrow$ Obfuscator $\longrightarrow$
Obfuscated $O(P)$
not understandable

Three properties:

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

```
                                           $_=’ev
                                       al("seek\040D
           ATA,0,                  0;");foreach(1..3)
       {<DATA>;}my               @camel1hump;my$camel;
  my$Camel  ;while(             <DATA>){$_=sprintf("%-6
9s",$_);my@dromedary           1=split(//);if(defined($
_=<DATA>)){@camel1hum        p=split(//);}while(@dromeda
 ry1){my$camel1hump=0      ;my$CAMEL=3;if(defined($_=shif
        t(@dromedary1    ))&&/\S/){$camel1hump+=1<<$CAMEL;}
       $CAMEL--;if(d   efined($_=shift(@dromedary1))&&/\S/){
      $camel1hump+=1  <<$CAMEL;}$CAMEL--;if(defined($_=shift(
     @camel1hump))&&/\S/){$camel1hump+=1<<$CAMEL;}$CAMEL--;if(
     defined($_=shift(@camel1hump))&&/\S/){$camel1hump+=1<<$CAME
     L;;}$camel.=(split(//,"\040..m‘{/J\047\134}L^7FX"))[$camel1h
      ump];}$camel.="\n";}@camel1hump=split(/\n/,$camel);foreach(@
      camel1hump){chomp;$Camel=$_;y/LJF7\173\175‘\047/\061\062\063\
      064\065\066\067\070/;y/12345678/JL7F\175\173\047‘/;$_=reverse;
       print"$_\040$Camel\n";}foreach(@camel1hump){chomp;$Camel=$_;y
        /LJF7\173\175‘\047/12345678/;y/12345678/JL7F\175\173\0 47‘/;
         $_=reverse;print"\040$_$Camel\n";}’;;s/\s*//g;;eval;   eval
           ("seek\040DATA,0,0;");undef$/;$_=<DATA>;s/\s*//g;(   );;s
             ;^.*_;;;map{eval"print\"$_\"";}/.{4}/g; __DATA__   \124
               \1   50\145\040\165\163\145\040\157\1 46\040\1  41\0
                    40\143\141  \155\145\1 54\040\1   51\155\  141
                    \147\145\0  40\151\156 \040\141    \163\16 3\
                     157\143\   151\141\16  4\151\1     57\156
                     \040\167  \151\164\1   50\040\      120\1
                     45\162\   154\040\15    1\163\      040\14
                     1\040\1   64\162\1      41\144       \145\
                     155\14    1\162\       153\04        0\157
                      \146\     040\11     7\047\         122\1
                      45\15      1\154\1  54\171          \040
                      \046\         012\101\16            3\16
                      3\15           7\143\15             1\14
                      1\16            4\145\163           \054
                     \040            \111\156\14         3\056
                    \040\         125\163\145\14         4\040\
                    167\1        51\164\1  50\0         40\160\
                  145\162                              \155\151
                \163\163                                \151\1
              57\156\056
```

Outline

1. Applications of Obfuscation
   - Classification of Threats
   - Applications in Software Protection
   - Applications in Mobile Agents
   - Applications in Cryptography
   - More Applications

Outline

1. Applications of Obfuscation
    - Classification of Threats
    - Applications in Software Protection
    - Applications in Mobile Agents
    - Applications in Cryptography
    - More Applications

2. Blackbox Secure Obfuscation
    - Defining Security of Obfuscation
    - Impossibility Result

Outline

1. Applications of Obfuscation
   - Classification of Threats
   - Applications in Software Protection
   - Applications in Mobile Agents
   - Applications in Cryptography
   - More Applications

2. Blackbox Secure Obfuscation
   - Defining Security of Obfuscation
   - Impossibility Result

Different Types of Attacks

How can adversary act with program?

Different Types of Attacks

How can adversary act with program?

- Study program (extracting knowledge)
- Decompose program (reusing code/algorithms of it)
- Change program behavior (making illegal modifications)

Different Types of Attacks

How can adversary act with program?

- Study program (extracting knowledge)
- Decompose program (reusing code/algorithms of it)
- Change program behavior (making illegal modifications)

More attacks?

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data
  - Availability of customization

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data
  - Availability of customization
  - Quality of performed tasks

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data
  - Availability of customization
  - Quality of performed tasks
  - Number of runs

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data
  - Availability of customization
  - Quality of performed tasks
  - Number of runs
- Watermarks protection

Applications in Software Protection

Situation: company distribute (sell) software products.

Question: Threats and applications you see?

- Integrity protection
  - Against decomposition and reusing code fragments
  - Against adding new functionalities
  - Against changing the order of computation
- Protection of internal constraints on:
  - Usage time
  - Input data
  - Availability of customization
  - Quality of performed tasks
  - Number of runs
- Watermarks protection
  - Deleting watermarks in obfuscated program is much harder

Protection of IF Operator

Consider a program containing 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.

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
    - Sending hard computational task to untrusted claster

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
    - Sending hard computational task to untrusted claster
    - Auxiliary computing devices for smart cards

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
  - Sending hard computational task to untrusted claster
  - Auxiliary computing devices for smart cards
- Illegal agent modification

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
    - Sending hard computational task to untrusted claster
    - Auxiliary computing devices for smart cards
- Illegal agent modification
    - Network monitoring system

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
  - Sending hard computational task to untrusted claster
  - Auxiliary computing devices for smart cards
- Illegal agent modification
  - Network monitoring system
- Keys protection

Mobile Agents Technology

Situation: author distribute programs for his own needs.

Question: Threats and applications you see?

- Privacy of data in mobile agents
  - Sending hard computational task to untrusted claster
  - Auxiliary computing devices for smart cards
- Illegal agent modification
  - Network monitoring system
- Keys protection
  - Buying agents

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 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.

Applications in Cryptography

What applications in cryptography can we imagine?

Applications in Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem → Public key cryptosystem

Applications in Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem → Public key cryptosystem
  - It was mentioned even in famous Diffie-Hellman paper

Applications in Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem → Public key cryptosystem
  - It was mentioned even in famous Diffie-Hellman paper
- Constructing homomorphic encryption schemes

Applications in Cryptography

What applications in cryptography can we imagine?

- Private key cryptosystem → Public key cryptosystem
  - It was mentioned even in famous Diffie-Hellman paper
- Constructing homomorphic encryption schemes
- Realizing random oracles in cryptosystems

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

Constructing Homomorphic Encryption

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

Question: Any ideas how to do this?

Constructing Homomorphic Encryption

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

Question: Any ideas how to do this?

Construction: as such homomorphic encryption we can take just any public key cryptosystem:

> Input: $E(x), E(y)$
> Program algorithm: using private key decrypt x and y, compute $x + y$ (respectively xy), 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!

More Applications

- Diversity producing (every user receive his own version)
  Makes virus attacks harder
- Guaranteed slowdown of encrypting procedure in cryptosystems
  Makes brute-force attacks harder
- Digital Rights Management software
  Protection against extracting secret keys from players for copyrighted media files

More Applications

- Diversity producing (every user receive his own version)
  Makes virus attacks harder
- Guaranteed slowdown of encrypting procedure in cryptosystems
  Makes brute-force attacks harder
- Digital Rights Management software
  Protection against extracting secret keys from players for copyrighted media files

Question: Your ideas of applications?

Outline

1. Applications of Obfuscation
   - Classification of Threats
   - Applications in Software Protection
   - Applications in Mobile Agents
   - Applications in Cryptography
   - More Applications
2. Blackbox Secure Obfuscation
   - Defining Security of Obfuscation
   - Impossibility Result

Security Definitions in Cryptography (1)

1. Define adversary's inputs
2. Define adversary's goal
3. Security $=$ achieving goal is computationally hard

Security Definitions in Cryptography (1)

1. Define adversary's inputs
2. Define adversary's goal
3. Security = achieving goal is computationally hard

Proof instrument:

Reduction: "If somebody can break this new system than he also able to solve some well-known hard problem"

Security Definitions in Cryptography (1)

1. Define adversary's inputs
2. Define adversary's goal
3. Security = achieving goal is computationally hard

Proof instrument:

Reduction: "If somebody can break this new system than he also able to solve some well-known hard problem"

Example: security of pseudorandom generators

Security Definitions in Cryptography (2)

1. Define ideal model
2. Security = adversary cannot compute more than in ideal model

Security Definitions in Cryptography (2)

1. Define ideal model
2. Security = adversary cannot compute more than in ideal model

Proof instrument:

Simulation: “For any property that could be extracted from the new system almost the same property can be extracted from the ideal model”

Security Definitions in Cryptography (2)

1. Define ideal model
2. Security = adversary cannot compute more than in ideal model

Proof instrument:

Simulation: "For any property that could be extracted from the new system almost the same property can be extracted from the ideal model"

Example: security of zero-knowledge proofs

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

Black-box Security

Randomized algorith O is an Obfuscator if three following conditions hold:

1. (functionality) $\forall$ TM M: $O(M) \approx M$
2. (effectiveness) $\exists p$: $M(x)$ terminates in t steps $\Rightarrow O(M)(x)$ terminates in $p(t)$ steps.

Black-box Security

Randomized algorith O is an Obfuscator if three following conditions hold:

1. (functionality) $\forall$ TM M: $O(M) \approx M$
2. (effectiveness) $\exists p$: $M(x)$ terminates in t steps $\Rightarrow O(M)(x)$ terminates in $p(t)$ steps.
3. (black-box security) For every PPT A there exists PPT S such that for all TMs M:

$$\left| Pr\{A(O(M)) = 1\} - Pr\{S^M(1^{|M|}) = 1\} \right| = \nu(|M|).$$

Unobfuscatable Function Family

Family $\mathcal{H} = \cup H_k$
H_k is a set (distribution) of functions $B^{n_k} \rightarrow B^{m_k}$

- $h \in H_k$ computable in poly(k) time
- $\exists \pi : \mathcal{H} \rightarrow \{0, 1\}$ such that
  - $|Pr\{S^h(1^k) = \pi(h)\} - 1/2| = \nu(k)$
  - $\exists A$ such that for every TM M computing h, $A(M) = \pi(h)$

Unobfuscatable 2-Functions Family

Family $\mathcal{G} = \cup G_k$

G_k is a set (distribution) of pairs of functions $B^{n_k} \longrightarrow B^{m_k}$

- $(g_1, g_2) \in G_k$ computable in poly(k) time
- $\exists \pi : \mathcal{G} \longrightarrow \{0, 1\}$ such that
  - $|Pr\{S^{g_1,g_2}(1^k) = \pi(g_1, g_2)\} - 1/2| = \nu(k)$
  - $\exists A$ such that for every TMs M_1, M_2 computing g_1, g_2, $A(M_1, M_2) = \pi(g_1, g_2)$

Unobfuscatable 2-Functions Family

Family $\mathcal{G} = \cup G_k$
G_k is a set (distribution) of pairs of functions $B^{n_k} \longrightarrow B^{m_k}$

- $(g_1, g_2) \in G_k$ computable in poly(k) time
- $\exists \pi : \mathcal{G} \longrightarrow \{0, 1\}$ such that
  - $|Pr\{S^{g_1,g_2}(1^k) = \pi(g_1, g_2)\} - 1/2| = \nu(k)$
  - $\exists A$ such that for every TMs M_1, M_2 computing g_1, g_2, $A(M_1, M_2) = \pi(g_1, g_2)$

Existence of unobfuscatable function families and 2-finction families. What follows from what?

Counterexample

Cannibalistic construction:

$$C_{\alpha,\beta}(x) \;=\; \begin{cases} \beta, & x = \alpha \\ 0, & \text{otherwise} \end{cases}$$

$$D_{\alpha,\beta}(C) \;=\; \begin{cases} 1, & C(\alpha) = \beta \\ 0, & \text{otherwise} \end{cases}$$

$$Z_k(x) \;= 0^k$$

Intuition: it is difficult to distinguish pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ from pair $Z_k, D_{\alpha,\beta}$ given only black box access to these programs.

Technical Details

We leave out technical details:

- Truncated version of D
- Combining pair of functions into a single one.

Extensions of Impossibility Result

More impossibilities of obfuscation:

- Unobfuscatable functional properties (not only predicates)
- Computationally easy but still unobfuscatable programs (in TC_0 class)
- Attack (deobfuscation algorithm) is known in advance
- Obfuscator might preserve functionality only approximately
- Impossibility of obfuscation for sampling algorithms

Home Problem 1

Whether the family $f_\alpha(x) = x \cdot \alpha$ is obfuscatable with black-box security?

Summary

Main points:

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

Summary

Main points:

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Black-box security: obfuscated program tells no more than input-output behaviour.

Summary

Main points:

- Rough idea of applications: cryptosystem design, mobile agents technology, software protection.
- Black-box security: obfuscated program tells no more than input-output behaviour.
- There exists unobfuscatable function families

Reading List

- B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S., Vadhan, K. Yang
  On the (Im)possibility of Obfuscating Programs, 2001.
  http://eprint.iacr.org/2001/069.

- K. Yang
  Talk on "(Im)possibility of Obfuscating", 2001.
  http://www.cs.cmu.edu/~yangke/papers/obf-talk.pdf.

- N. Kuzyurin, N. Varnovsky and V. Zakharov
  Mathematical problems of Obfuscation, 2004.
  http://www.ispras.ru/news/downloads/obfuscation/obfuscation.pdf.

- Yu. Lifshits
  Lecture Notes on Program Obfuscation, 2005.
  http://logic.pdmi.ras.ru/~yura/obfuscation.html.

Thanks for attention. Questions?