Provable Security for Program Obfuscation

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

Outline

Outline

Outline

Perfect Security

What do we want to get?

Perfect Security

What do we want to get?

We want to be sure that our system is safe to use.

In lecture 4 “Applications of Obfuscation” we’ll discuss what kind of safety we want to get by obfuscation.

Today: what does it mean to be sure about safety?

Usual approach: to build some proof of safety.

Ways to Security

How are we going to prove security?

Ways to Security

How are we going to prove security?

- Theoretic security: obfuscated program doesn't provide enough information to successful attack

  Example: exact reverse engineering. Solution: delete comments
- Computational (cryptographic) security: attack required too much computation

Necessary hardness of attack: average superpolynomial complexity.
Now: no problems with such proved complexity

Ways to Security II

So what can we accept as enough hard problem?

Ways to Security II

So what can we accept as enough hard problem?

- NP-hard problems. Disadvantage: worst case complexity
- NP-hard problems with average complexity results.
  Example: SUBSET SUM
- Problems with wide-believed hardness:
  Examples: FACTORING, DISCRETE LOG

Current Results

What are the best results to the moment?

Current Results

What are the best results to the moment?

- ⇒ Specific attacks on specific programs are computationally hard
- ⇒ For some classes of programs we can hide most of internal information
- ⇒ Some program analysis is proved to be hard

Current Results

What are the best results to the moment?

- ⇨ Specific attacks on specific programs are computationally hard
- ⇨ For some classes of programs we can hide most of internal information
- ⇨ Some program analysis is proved to be hard
- ⇨ And obfuscation in general is impossible!

Ana and BAna

Slide from Lecture 1 — your turn to explain.

Ana and BAna

Slide from Lecture 1 — your turn to explain.

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

Slide from Lecture 1 — your turn to explain.

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: Formal Definition

A nondeterministic algorithm O is a TM obfuscator if three following conditions hold:

- **(functionality)** For every TM M, the string $O(M)$ describes the same function as M.

Black-Box Security: Formal Definition

A nondeterministic algorithm O is a TM obfuscator if three following conditions hold:

- **(functionality)** For every TM M, the string $O(M)$ describes the same function as M.
- **(polynomial slowdown)** The description length and running time of $O(M)$ are at most polynomially larger than that of M.

Black-Box Security: Formal Definition

A nondeterministic algorithm O is a TM obfuscator if three following conditions hold:

- **(functionality)** For every TM M, the string $O(M)$ describes the same function as M.
- **(polynomial slowdown)** The description length and running time of $O(M)$ are at most polynomially larger than that of M.
- **("virtual black box" property)** For any PPT A, there is a PPT S and a negligible function α such that for all TMs M

$$\left|Pr[A(O(M)) = 1] - Pr[S^M(1^{|M|}) = 1]\right| \leq \alpha(|M|).$$

Black-Box Security: Two Programs

A 2-TM obfuscator is defined in the same way as a TM-obfuscator, except the "virtual black box" property is changed as follows

- **("virtual black box" property)** For any PPT A, there is a PPT S and a negligible function α such that for all TMs M and N

$$\left| Pr[A(O(M), O(N)) = 1] - Pr[S^{M,N}(1^{|M|+|N|}) = 1] \right| \leq \alpha(min(|M|, |N|)).$$

Black-Box Security: Two Programs

A 2-TM obfuscator is defined in the same way as a TM-obfuscator, except the "virtual black box" property is changed as follows

- **("virtual black box" property)** For any PPT A, there is a PPT S and a negligible function α such that for all TMs M and N

$$\left|Pr[A(O(M),O(N))=1]-Pr[S^{M,N}(1^{|M|+|N|})=1]\right|\leq\alpha(min(|M|,|N|)).$$

What obfuscator is more powerful?

Two Programs Lemma

Two Programs Lemma

2-TM obfuscators do not exist.

Two Programs Lemma

Two Programs Lemma

2-TM obfuscators do not exist.

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

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

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

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.

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

⇨

$$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] \; = \; 1$$

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

- $$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] \;=\; 1$$
- $$Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] \;=\; 2^{-k}$$

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

- ⇨ $$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] = 1$$
- ⇨ $$Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] = 2^{-k}$$
- ⇨ $$\left| Pr[S^{C_{\alpha,\beta}, D_{\alpha,\beta}} = 1] - Pr[S^{Z_k, D_{\alpha,\beta}} = 1] \right| \leq 2^{-\Omega(k)}$$

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

- ⇨
$$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] \;=\; 1$$

- ⇨
$$Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] \;=\; 2^{-k}$$

- ⇨
$$\left| Pr[S^{C_{\alpha,\beta}, D_{\alpha,\beta}} = 1] - Pr[S^{Z_k, D_{\alpha,\beta}} = 1] \right| \leq 2^{-\Omega(k)}$$

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

- $$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] \;=\; 1$$
- $$Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] \;=\; 2^{-k}$$
- $$\left| Pr[S^{C_{\alpha,\beta}, D_{\alpha,\beta}} = 1] - Pr[S^{Z_k, D_{\alpha,\beta}} = 1] \right| \leq 2^{-\Omega(k)}$$

So we get a contradiction! But...

Lemma Proof: Rough Sketch

Suppose O is 2-TM obfuscator. Let's check its "black box" property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every α, β where $A = N(M)$.

- ⇨
$$Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] \;=\; 1$$
- ⇨
$$Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] \;=\; 2^{-k}$$
- ⇨
$$\left| Pr[S^{C_{\alpha,\beta}, D_{\alpha,\beta}} = 1] - Pr[S^{Z_k, D_{\alpha,\beta}} = 1] \right| \leq 2^{-\Omega(k)}$$

So we get a contradiction! But...

There is a flaw in the proof. Do you see?

Impossibility Theorem

Impossibility Theorem

TM obfuscators do not exist.

Impossibility Theorem

Impossibility Theorem

TM obfuscators do not exist.

$$F_{\alpha,\beta}(b,x) = C_{\alpha,\beta}\#D_{\alpha,\beta}$$

$$G_{\alpha,\beta}(b,x) = Z_k\#D_{\alpha,\beta}$$

Algorithm A is the following: to decompose M into two parts and evaluate the second part on the code (encoding) of the first.

Argument is similar to the Lemma's proof.

Property Hiding

Slide from Lecture 1 — your turn to explain.

Property Hiding

Slide from Lecture 1 — your turn to explain.

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

Slide from Lecture 1 — your turn to explain.

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.

Password Checking Hiding

prog π_1^w;
var x:string, y:bit;
input(x);
if $x = w$ then y:=1 else
y:=0;
output(y);
end of prog;

Password Checking Hiding

prog π_1^w;
var x:string, y:bit;
input(x);
if $x = w$ then y:=1 else
y:=0;
output(y);
end of prog;

prog π_0;
var x:string, y:bit;
input(x);
y:=0; output(y);
end of prog;

Task: Make this families indistinguishable.

Some Theoretical Background

One-Way Permutation is bijection from the set of all binary strings of length k to itself which is easy to compute and difficult to inverse.

$$F : B^k \to B^k$$

Hardcore Predicate for one way permutation F is a predicate (i.e. boolean function) h such that given $F(x)$ its difficult to predict $h(x)$ better than just guess it.

Usual construction of hard-core predicate: choose r by random and take any one way permutation F than given a pair $(F(x), r)$ its difficult to uncover $x \cdot r$.

Program with hidden password checking

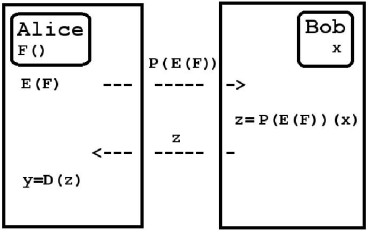
```
prog Π
var x: string, y:bit;
const u, v:string, σ:bit;
input(x);
if ONE_WAY(x)=v then
      if x · u = σ then y:=1 else y:=0;
else y:=0;
output(y);
end of prog;
```

Model of Encrypted Computation

Slide from Lecture 1 — your turn to explain.

Model of Encrypted Computation

Slide from Lecture 1 — your turn to explain.



Basic task: keep F unknown to Bob.

Homomorphic Encryption

General idea: to design an encoding such that it is possible to evaluate various operations over encrypted messages (and getting encrypted results) without decrypting them.

In particular encoding is called

- ⇨ **Additively homomorphic** if it is possible to compute $E(x+y)$ from $E(x)$ and $E(y)$
- ⇨ **Multiplicatively homomorphic** if it is possible to compute $E(xy)$ from $E(x)$ and $E(y)$
- ⇨ **Mixed multiplicatively homomorphic** if it is possible to compute $E(xy)$ from $E(x)$ and y.

Homomorphic Encryption II

Fact: there exists additively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

Corollary: there exists additively & mixed multiplicatively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

Homomorphic Encryption II

Fact: there exists additively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

Corollary: there exists additively & mixed multiplicatively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

Proof: Mixed multiplication could be done by polynomial number of additions.

Encrypting Polynomials

Let P be polynomial over $\mathbb{Z}/N\mathbb{Z}$ ring.

$$P = \sum a_{i_1 \dots i_s} X_1^{i_1} \dots X_s^{i_s}$$

Then we can encrypt P by just sending encrypted coefficients (using MM-A homomorphic encryption). Bob is able to compute $E(P(X))$ and return it back to Alice.

What we reveal to Bob? Only set of nonzero coefficients of P.

Mobile cryptography results

What are further results for encrypted computation?

Mobile cryptography results

What are further results for encrypted computation?

⇨ Other presentations of function.
- [Loreiro, Molva] – function as a matrix.
- [Sander, Tschudin] – another basic hard problem: decomposition of rational functions.

More Black-Box Security

What are other functions obfuscated with black-box security?

More Black-Box Security

What are other functions obfuscated with black-box security?

⇨ **[LPS 2004]** – interactive access control system.

More Black-Box Security

What are other functions obfuscated with black-box security?

- [LPS 2004] – interactive access control system.
- Next results I expect from you!

Quality of obfuscating transformations

What is hard to get from programs after obfuscating transformations?

Quality of obfuscating transformations

What is hard to get from programs after obfuscating transformations?

- Alias analysis is NP-hard!
- Average hardness is proved only for several fixed analysis algorithms

Summary

⇨ We can prove property extracting to be hard in some cases.

Summary

- We can prove property extracting to be hard in some cases.
- We can use cryptographic constructions to hide some internal constants.

Summary

- ⇨ We can prove property extracting to be hard in some cases.
- ⇨ We can use cryptographic constructions to hide some internal constants.
- ⇨ Obfuscation in general is impossible.

Summary

- We can prove property extracting to be hard in some cases.
- We can use cryptographic constructions to hide some internal constants.
- Obfuscation in general is impossible.

Question Time!

Not covered by the talk

- ⇨ Black box security with relations to zero-knowledge