Usage of Hard Problems for Program Obfuscation

Basic Complexity Results

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Outline

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Rice's Theorem

Program analysis framework:

Each TM compute some partially defined function: input is a string which is written on the tape at the start and output is a string which is written after halting of TM.

Given any nontrivial function property P we can search for algorithm for determining P for a function computed by any given TM.

Does this algorithm exists?

Rice's Theorem

For any nontrivial property P problem whether a function computed by given TM satisfies P is undecidable.

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Kerckhoff's Principle

Auguste Kerckhoffs (19th century):

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

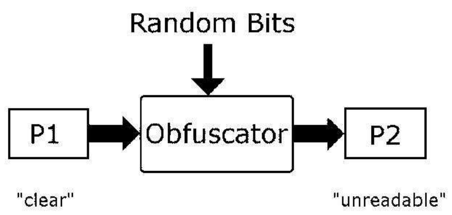
It was reformulated (perhaps independently) by Claude Shannon as "the enemy knows the system". It is widely embraced by cryptographers, in opposition to security through obscurity.

In accordance with Kerckhoffs' law, the majority of civilian cryptography makes use of publicly-known algorithms. By contrast, ciphers used to protect classified government or military information are often kept secret.

Eric Raymond: Security Through Transparency

Open-source software is inherently more secure than closed-source.

Random Bits of Obfuscator



- Random choice of obfuscating transformation
- Random choice of parameters of a single transformation

Deobfuscation is in NP

So, what is NP class about?

- We say $L \in NP$ iff there exists polynomial algorithm A such that
$$x \in L \Leftrightarrow \exists w : A(x, w) = 1$$
- We say $S \in \widetilde{NP}$ iff there exists polynomial algorithm A such that
$$(x, y) \in S \Leftrightarrow \exists w : A(x, y, w) = 1$$

Deobfuscation is in $\widetilde{NP}$

Deobfuscation input: $O(P)$, solution: P.

Proof: Take random bits of obfuscator as w!

What Means Always Hard?

Complexity theory:
- Worst case complexity

Cryptography:
- Almost every case complexity

Security proofs in classical cryptography:
If somebody can break given cryptosystem then he is also able to solve some computational problem with high every-case complexity.

Always Hard Problems

Some examples of problems with believed high every-time complexity:

- FACTORING: given $N = pq$ find p and q.
- DISCRETE LOG: given a, N and $(a^x \ mod\ N)$ find x.
- SUBSET SUM: given $w_1, \ldots, w_n$ and t determine whether exist $x_1, \ldots, x_n \in \{0, 1\}$ such that $\sum x_i w_i = t$
- Decomposition of multivariate polynomials
- Some special linear codes decoding: given message x find nearest codeword.

Oblivious Transfer

So, what is Oblivious Transfer?

- Two players Alice and Bob
- Bob holds some information items $x_1, \dots x_n$
- Alice want to get x_i from Bob and at the same time keep i as a secret from Bob
- Bob wanted to reveal not more than one item to Alice

And there are protocols achieving this goal!

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Secret Multiparty Computation

So, what is Secret Multiparty Computation?

- Several players $A_1, \dots, A_k$
- Several input items $x_1, \dots, x_n$
- Predefined function $F(x_1, \dots, x_n)$
- Every player knows only subset of input set
- Goal: to compute F in the way that nobody get more knowledge about $x_1, \dots, x_n$ than just his subset and value of F

Examples: Millionaire problem, Electronic voting

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Homomorphic Encryption

Slide from Lecture 3 — your turn to explain.

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 $E(y)$
- **Algebraically homomorphic** if it is both additive and multiplicative.

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

One-Way Functions

So, what is One-Way Functions and One-Way Permutations and Trap-Door Functions?

Informally:

- One-Way Function:
  - polynomially computable function
  - but not polynomially reversible
- One-Way Permutation:
  - polynomially computable bijection
  - but not polynomially reversible
- Trap-Door Function: parametric function with such a description that:
  - it is polynomially computable
  - not polynomially reversible given only description
  - but given explicit value of parameter is polynomially reversible!

Pseudo-Random Functions

So, what are Pseudo-Random Generators and Pseudo-Random Functions?

Informally:

- Pseudo-Random Generator is a family of functions such that:
  - they compute mappings from $\mathbb{B}^n$ to $\mathbb{B}^m$, $m > n$
  - given a black-box access to representative of family it is computationally hard to distinguish it from truly random generator
- Pseudo-Random Function is a function G such that:
  - it computes a mapping from $\mathbb{B}^n$ to $\{F : \mathbb{B}^m \rightarrow \mathbb{B}^k\}$, $k > m$
  - given a black-box access random result of G it is computationally hard to distinguish whether it was generated by G or was randomly chosen from all functions ($\{F : \mathbb{B}^m \rightarrow \mathbb{B}^k\}$)

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Necessity of Model

What do we need to define in order to prove security of obfuscated program?

- Program representation
- Secret of program
- Adversary knowledge about program
- Adversary success
- Security of obfuscated program

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Security Definition

How can we define security of obfuscated program

- Explicitly
  - Define adversary task and require that it should be computationally difficult
  - Disadvantage: there are a lot of threats and some of them are difficult to formulate in terms of computational problems
- Implicitly
  - Define ideal security model and require that our case is nearly as good as ideal one
  - Disadvantage: Impossibility result by **[Barak et al.]**

Basic Complexity Results
Yury Lifshits
Around Kerckhoff's Principle
Hardness of Program Analysis
Kerckhoff's Principle
Nondeterminism in Obfuscation
Potentially Useful Constructions
Always Hard Problems
Famous Cryptographic Notions
From Cryptography to Obfuscation
Necessity of Model
Not Formalized Concepts
Informal Guidelines
Summary

Obfuscation: Cryptography vs. Obscurity

Is cryptographic security necessary?

- For most applications obfuscation without guaranteed security isn't acceptable solution
- Still some applications (competitors threat, watermarks protection) can benefit from "good" obfuscation
- Possible way out: challenge proofs of security

Obfuscation Limits

If obfuscation in general is impossible can we find some necessary and/or sufficient conditions of existence of secure obfuscation?

- First limit of obfuscation: it is useless against black-box attacks
- Are there other limits? **[Barak et al.]**: Yes! Can we describe them?
- Any classes with possible secure obfuscation?

Basic Complexity Results
Yury Lifshits

Program Secrets

How can you define program secrets?

- Key's or parameters involved in program
- State of the program
- Data structure
- Used algorithms?

Basic Complexity Results
Yury Lifshits

Informal Guidelines

What ideas can we suggest for development of new obfuscation methods?

- Obfuscation: general vs. local
- Kernel approach
- Inductive constructions
- Encryption of all intermediate results
- Hidden self-checking

Basic Complexity Results
Yury Lifshits

Summary

- Theoretical background: Rice's theorem, Kerckhoff's law.
- Cryptographic Constructions: One-Way Functions, PRG, MSC, OT and Homomorphic Encryption.
- Guidelines for future obfuscation: randomness, locality, usage of cryptographic constructions.

Question Time!

For Further Reading

B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang.
On the (Im)possibility of Obfuscating Programs

Basic Complexity Results
Yury Lifshits
Appendix
References
Not covered by the talk

Disassembling hardness
Rareness of event
Random oracle model
Zero-knowledge connections [Hada]
Secret sharing
Coin flipping protocols