February 21, 2002

Ke Yang

Saili Vadhun
Amit Sahai

Steven Rudich
Russell Impagliazzo

Oded Goldreich
Boaz Barak

Joint work with

Programs

On the (Im)possibility of Obfuscating
(American Heritage Dictionary)

understand

To make so confused or opaque as to be difficult to perceive or

\texttt{\textbackslash \texttt{Obfuscate}} \texttt{\textbackslash Ob\texttt{\textbackslash tuss}} case :
Obfuscating a Program

Obfuscator: a "compiler" that makes programs "unreadable" that is built to understand (to both humans and computers) making the program impossible for-
gcc converts any program into a "human-unreadable" form.

```
prog.o  gcc  prog.c
```

What do Obfuscators Look Like?
"If you don't understand it, you can't steal it."

• Intellectual Property Protection

"If you don't understand it, you can't mess around with it."

• Security through obscurity

• FUN

---

Why do People do Obscuration?
Obfuscated

Protecting parts of its OS from reverse engineering.

- Microsoft
- "Tamper Resistant Software"
- cloakware.com

Who Makes Obfuscators?
Why are they too good to be true?

Why do we want them so badly?

What are obfuscators?

We investigate the theoretical notion of program obfuscation.
What is an Obfuscator — Intuitions
We need to make a distinction between Black Boxes and source codes... But what does it mean to say it is unreadable?

A program is always executable.
What can you do with a black box?

Running time
• Input-output behavior

Very limited information...

```c
#include <stdio.h>

int myStressTest(int x, int y) {
    /*
    * number
    * and outputs a stressful
    * operations to input x and
    * this function does weird
    */

    mysterious.h
```
What can you do with the source code?
Mutation Analysis: Change Fragments of the Program.

Efficiency Analysis: Statistics, hot-spots

Dynamic Analysis: Stacks, Program Flow

Static Analysis: Basic blocks, variable usage

A Source Code Analyzer Can do More...
as an oracle.

* **Bana** is a black-box analyzer that only queries the program.

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

We are interested in 2 types of polynomial-time analyzers:
Is it true that Ana is always strictly more powerful than Bana?

Furthermore, Ana can obtain information that Bana cannot get, like the program flow.

• Ana can simulate Bana.

Ana seems more powerful than Bana.

Ana vs. Bana
**Case Study: How to Make Instruction Trace Useful**
Consider the following lazy compiler...
So \texttt{Bana} can generate the trace without even knowing the
\texttt{P1}'s of the same running time.

The instruction trace of \texttt{P2} is always the same for different

\texttt{P2}

\begin{center}
\begin{tabular}{|c|c|}
\hline
Virtual Machine & \texttt{P1} \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|}
\hline
\texttt{P1} \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|}
\hline
Lazy compiler \\
\hline
\end{tabular}
\end{center}

\textbf{Useless Instruction Trace}
... same amount of time.

- Add dummy code, so each piece of the program uses the same for all programs of the same running time.
- Create dummy variables, so the number of variables is the same.

A compiler can hide a lot of features from the source code.

More Generally...
unreadable programs.

Intuition: Ana isn’t necessarily more powerful than Bana for

Information from Ana.

It seems that you can always change your program to „hide“

Is Ana Always More Powerful?
and an obfuscator converts any program into a virtual black-box.

Anything Ana can do, Banana can do as well...

An unreadable program is like a virtual black-box.
\( \text{Ana}(P_2) \approx \text{Bana}_{P_2}(\text{time}(P_2)) \)

- For every \( \text{Ana} \), there is a \( \text{Bana} \), such that \( P_1 \) and \( P_2 \) compute the same function.

- An Obfuscator is an efficient, randomized compiler.

---

**What's an obfuscator?**

---

**Semi-formal Definition**
What can we do with them?

We have seen a definition of obtusscators

Why do we want obtusscators? That
A strong guarantee that nobody can understand your program.

Obfuscated Code Protection
The private key is the key.
The public key is the obfuscated encryptor with a fixed key.

Converting Any Private-Key System to a Public-Key System
caption is shown to be computationally infeasible.

"... the task of decloaking a Data Encryption Standard/appil-

DES encyptor.

They used their Temper-Resistant Technology on obfuscating a

Cloakware.com
Removal of the Random Oracle

Homomorphic Encryption

Private-key to Public-key Conversion

no one knows how to do now.

Obfuscators will imply a lot of cryptographic applications that

Badly?

Why People Want Obfuscators so
However, they are too good to be true...

We have seen what one can do obtusators

Why are obtusators too good to be true?
\[
\forall \varphi : \lambda \rightarrow \mathbb{N} \quad \exists \varphi' : D \end{align*}
\]

called version, \(P_2\) such that for any Turing Machine \(P_1\) and its obtus-

For any \(\text{Ana}\), there exists a virtual Black-Box Condition:

- \(\text{Poly} \begin{align*}
\text{Polynomial Slowdown: } \exists \text{such that a polynomial } \end{align*}
\]
cryptographically applicable applications.
- An even weaker definition probably won’t have provable
  - All other definitions imply this one.
  - Empirically, this one is the “minimal” definition:

There are many possible definitions we considered

Am I cheating by presenting a definition that’s too strong?

„Straw-man Definition“
cats vs. don't exist.

The existence of secret-leaking functions will imply that obtus-

But any program that computes $f$, leak' the secret i

high probability

No banana using $f$ as an oracle can obtain the secret with

Each $f$ contains a "secret" $s$

There exists efficient functions $f$s such that

Secret-Leaking Functions
but any source code will leak the secret to Ana.

The secret-leaking function cannot leak the secret to Banana.

How to leak your secret?
and Bana doesn't?

How do you make sure that Ana knows the correct input?

- The function outputs the secret if you give the correct input.
- Doesn't work for every source code.
- Encode the secret as comments in source code.
- Simple approaches don't work.

Mission Impossible?
But not obtainable via black-box access.

Obtainable from any source code.

We need the correct input to be...
What's the correct input?

How about making the source code itself the correct input?
which behaves exactly like CANNIBAL.

• But Ana can since she has the source code for CANNIBAL.

behaves like CANNIBAL.

• Without the source code, Ana cannot produce a prog that

| ELSE OUTPUT "000"; |
| THEN OUTPUT secret; |
| IF (prog behaves like me) |
| FUNC CANNIBAL (prog) |

"Feed me somebody that behaves like me, and I'll leak my secret!"

Cannibalistic Function (Intuition)
\( s = (p x) (\text{ID} x) \)

Leaker \( p \) has the correct behavior. Leaker will output the secret \( s \) only when the input program

ID has the correct "behavior" of CANNIBAL.

\[
\begin{cases}
0 & \text{otherwise} \\
(p) x & \text{if } s \\
0 & \text{otherwise}
\end{cases}
\]

\[
\begin{cases}
0 & \text{otherwise} \\
(x) & \text{if } p \\
0 & \text{otherwise}
\end{cases}
\]

The function CANNIBAL consists of 2 parts: ID and Leaker.

---

Formal Definition
\[
\begin{align*}
I & = q \\ 0 & = q
\end{align*}
\]

We combine 2 functions into one single function.

**Putting 2 Functions Together**
How Ana can obtain the secret

STEP 1: Generate the source code for ID.

```c
int my_variable;
const int b = 0;
ID(char *y)
    { ... 
        int my_variable;
        cannibal(char *y, int b);
    }
```

STEP 2: Run cannibal on ID to get the secret.

```c
secret = cannibal(ID, 1);
```

How Ana can obtain the secret
them is exponentially small.

If \( a', b', s \) are all chosen randomly, the probability to find

\[ Bana \text{ only makes polynomially many queries.} \]

\[
\begin{align*}
\left\{ \begin{array}{ll}
\text{otherwise} & 0 \\
\text{otherwise} & 0
\end{array} \right. \\
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{ll}
\text{otherwise} & 0 \\
\text{otherwise} & 0
\end{array} \right. \\
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{ll}
\text{otherwise} & 0 \\
\text{otherwise} & 0
\end{array} \right. \\
\end{align*}
\]

\[ (d)^{s,b'} \]

\[ = (x)^{d,b'} \]

\[ \text{ID} \]
No obfuscators exist for CANNIBAL.

\{ \text{CANNIBAL}_x^{g,s} \}

\begin{align*}
q = 1 \quad &\left( \text{leak}\text{er} \right)_{g^{s,\infty}} \\
q = 0 \quad &\left( \text{ID} \right)_{g^{s,\infty}}
\end{align*}

Putting Everything Together...
them to the circuit piece-by-piece. Since a circuit cannot eat itself, you have to chop it into pieces and feed as well, though the proof is trickier. The result holds for the Circuit model for the Turing Machine model. We just proved the impossibility results.
For example: private-key encryption functions.

How about special-purpose obfuscators for some natural functions. It is impossible to design a general-purpose obfuscator for

What did we just prove?
BALANCED-ENCRYPTOR is insecure.

- Any source-code implementation of CANNIBAL-ENCRYPTOR is a secure private-key system if used as a black-box.

<table>
<thead>
<tr>
<th>CANNIBAL-ENCRYPTOR (X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF (X behaves like me)</td>
</tr>
<tr>
<td>THEN OUTPUT secret-key!</td>
</tr>
<tr>
<td>ELSE OUTPUT encrypt(X)</td>
</tr>
</tbody>
</table>

"Feed me somebody that behaves like me, and I'll leak my secret key!"
Message Authentication Codes (MAC)

Pseudorandom Functions

Digital Signature Schemes

Encryption Schemes

There don't exist obfuscators for:

Obfuscation

More Impossibility Results on
Conclusions

• The impossibility results hold for obfuscating natural crypto-
  graphical functions

• Applications for Obfuscators

• Definitions of Obfuscators (virtual black-box property)
Any questions?