Theory of Obfuscation and its practical applications

Amir Herzberg and Haya Shulman

Bar Ilan University
What is Obfuscation? (Intuitively)

• A compiler; output is `obfuscated' program
  – Obfuscated program has same functionality as original, and similar performance

• Intuitive Security Goals:
  – Hide program
  – Hide secrets inside a program
  – Prevent modification of program

• Used to protect program running in untrusted PC:
  – DRM, eVoting, `trusted' TCP, policy enforcement, …
Obfuscation: Theory vs. Practice

• Practice
  – Obfuscation widely used
  – Lots of skepticism: security, impossibility alike

• Theory
  – Precise definitions (several variants)
  – Impossibility results
  – Positive (possibility) results
  – Related tools

• This talk: review of theory and its applications
Theory of Obfuscation: Outline

• Introduction
• Definition: Virtual Black-Box Obfuscation
• Impossibility (negative) result
• Positive results and challenges
• Beyond black-box obfuscation
  – Non-malleability
  – Verifiable non-malleability
• Few related goals and tools
  – Public-key Obfuscation
  – WBRPE (White-Box Remote Program Execution)
• Conclusions and open questions
Definition: Virtual Black-Box Security
[Barak et al., 2001]

• An obfuscator $\mathcal{O}$ is an efficient compiler that on input $P$ outputs $\mathcal{O}(P)$, such that:
  
  – Functionality:
  • For every $P$, $\mathcal{O}(P)$ computes the same function as $P$
  • Program $\mathcal{O}(P)$ is slightly slower (and larger) than $P$

  – Virtual Black-Box:
  • Whatever Adv can compute with obfuscated code $O(P)$
  • A `black-box Adv` BB can compute by only calling $P$

\[
\text{Adv}(O(P)) \approx \text{BB}(P(x)) = \text{P}(x)
\]
Example: Obfuscating a Point Function

[Canetti97]

- Point function $I_x(w) = \{ 1 \text{ if } w = x, \ 0 \text{ otherwise} \}$
- Obfuscate $I_x$ with perfectly one-way function $f$

Let $y = f(x)$

Program Obf$I_x$ $(w)$:

\[
\{ \text{ if } y = f(w) \text{ return 1 else return 0 } \}
\]

- Intuitively: $y = f(x)$ reveals no more info than black-box access to $I_x$

```
Adv(f(x)) \approx Adv(\text{Obf}I_x) \approx \text{BB} \xrightarrow{x} f(x)
```
Example: Obfuscating a Point Function [Canetti97]

- Point function $I_x(w) = \{1$ if $w=x$, 0 otherwise $\}$
- Obfuscate $I_x$ with perfectly one-way function $f$
  Let $y = f(x)$
  Program Obf$I_x$ ($w$):
  \[
  \{ \text{if } y = f(w) \text{ return 1 else return 0} \}
  \]
- Intuitively: $y = f(x)$ reveals no more info than black-box access to $I_x$
- But: this is a very specific obfuscator...
- Is there general obfuscator (for all programs)?
Barak's Unobfuscatable Program

• Is there a general obfuscator (for all programs)?
• [Barak et al, 01] No!
• They present a program P that cannot be obfuscated
  – Hence: no obfuscator for all programs!
• First, they present two programs C, D and then transform it into P
• Let C, D be two programs specified by two secret strings (\(\alpha, \beta\))
  – Upon input x, \(C_{\alpha\beta}\) returns \(\beta\) if \(x=\alpha\) and returns 0 (of same length as \(\beta\)) otherwise
  – Upon input a program, D runs it with input \(\alpha\), and if the result is \(\beta\), returns 1 otherwise 0
No Virtual Black-Box Compiler for Every Program

- Obfuscate \( C \) and \( D \), to obtain \( O(C_{\alpha\beta}) \) on \( O(D_{\alpha\beta}) \)
- Evaluating \( O(C_{\alpha\beta}) \) on \( O(D_{\alpha\beta}) \), always results in 1
- Black-box access to \( C \) and \( D \) is similar to black-box access to \( D \) and some program that always returns 0

\[
\begin{array}{ccc}
D_{\alpha\beta} & 1 & \neq \\
C_{\alpha\beta} & & \\
\end{array}
\]

- To extend the impossibility to single program, define \( P=D(C) \)
No Virtual Black-Box Compiler for Every Program

• So: some programs that cannot be obfuscated according to virtual black box definition

• So what? Practical implication? Options:
  – Ignore…?
  – Consider alternative, e.g., weaker definitions
    • None so far?
  – Consider obfuscation of specific programs
    • Which? Few `positive results`...
      • E.g., point function obfuscation, re-encryption
Theory of Obfuscation: Outline

• Introduction
• Definition: Virtual Black-Box Obfuscation
• Impossibility (negative) result
• Positive results and challenges
• Beyond black-box obfuscation
  – Non-malleability
  – Verifiable non-malleability
• Related goals and tools
  – Public-key Obfuscation
  – Secure function evaluation (SFE) and Garbled Circuits
  – Cryptocomputing and homomorphic encryption
• Conclusions and open questions
Obfuscator for Shared-Key Encryption

• Let \((E_K, D_K)\) be shared key encryption

• [Hofheinz, Malone-Lee, Stam07]: there exists obfuscatable encryption schemes
  – \(O(E_K)\) gives public key encryption!

• How? Let \((E', D')\) be public key encryption

• Define shared key \((E, D)\) scheme with key \((e, d)\).
  (both keys of public key scheme)

• Then \(O(E_{e,d}) = E'_e\) is obfuscation of \((E, D)\)...
  – But again, is this `real' obfuscator??
Challenge: Obfuscatable Program

• All `positive results` use trivial obfuscators
  – Based on properties of program
• Challenge: find programs $P = \{P_K\}$ s.t.:
  – Impossibility does not (seem to) hold for $P$
  – Yet, no `trivial' obfuscator for $P$

  • Preferably, non-trivial obfuscator – e.g. one that may work for some other programs too...
Non-Malleable Obfuscation

• Ensure non-malleability of obfuscated program
  – Alice obfuscates a decryption algorithm which outputs the encrypted message only if certain conditions hold
  – Eve modifies the program to always output the result of decryption

• Goal: prevent modifications of the obfuscated programs
Verifiable Non-Malleable Obfuscation

- Obfuscation is verifiably non-malleable if the only programs attacker can create that pass verification are those it could create given black-box access to obfuscated code
- Allows to detect attacks that were not prevented
  - e.g., digitally sign obfuscated program, then verification procedure will check that the signature attached to the obfuscated program is correct
    - According to unforgebility property it is impossible to modify the program
    - How does the verification procedure obtains the verification key?
      - May not be practical
Public-Key Obfuscation

- Public-Key obfuscator is a pair of algorithms (Compile,D)
  - Use **Compile** to obfuscate secret program P, obtain O(P),K
  - The output is encryption of original program’s output
  - Use **D** with K to recover result of O(P)(x) on some input x
  - Correctness: for any input x, D(C(x))=P(x)
Fin