Obfuscation: Positive Results and Techniques

Benjamin Lynn
Stanford University

Manoj Prabhakaran
Princeton University

Amit Sahai
Princeton University

EUROCRYPT '04
Obfuscation

- Hide the internals of a program/circuit
- Still give complete access to the functionality
- Obfuscate and handover the code
Obfuscation

- Privacy, intellectual property protection, ...
- Numerous cryptographic applications
- Widespread interest
- Many proposed schemes
Definition?

- Introduced in [BGIRSVY'01]

- Cryptographic perspective: semantic security against "efficient" adversaries

- Intuition: Obfuscated code doesn't reveal anything more than what access to the functionality does
Definition

A family of functions $F$ is obfuscatable if:

There is $O$ such that for all $F.exe$ in $F$,

- $O(F.exe) = ob_F.exe$ has the same behaviour as $F.exe$
- $ob_F.exe$ is at most polynomially slower/bigger than $F.exe$
- Virtual Blackbox Property
Virtual Blackbox

For every adversary $A$ there is a “simulator” $S$ such that for all $F.exe$ in $F$, what $A$ can find out about $F$ from $ob_F.exe$, $S$ can find out just from blackbox access to $F$.

\[ | \Pr[A(ob_F.exe)=1] - \Pr[S^F=1] | < \text{negl} \]
There are unobfuscatable functions: in particular there are no universal obfuscators

Unobfuscatable cryptographic schemes

Low-complexity (TC⁰) unobfuscatable functions
Possibility of Obfuscation?

- If “learnable” then trivially obfuscatable

- May be obfuscators for many individual functions of interest

- At least one non-trivial obfuscation?
Compositions?

- Suppose \( \mathcal{F} \) and \( \mathcal{G} \) obfuscatable

- \{ \, f(g(x)) \mid f \text{ in } \mathcal{F}, g \text{ in } \mathcal{G} \, \} \text{ obfuscatable?}

- In particular, \( \mathcal{F}^k \) obfuscatable?

- Not necessarily!
Impossibility of Composition

- Depth 1 threshold circuits: trivially obfuscatable

- But constant depth threshold circuits ($TC^0$) can be unobfuscatable!
Reductions

- If $F$ "reduces to" $G$ and $G$ obfuscatable then $F$ also obfuscatable

- "Blackbox reductions": given any obfuscator for $G$ give one for $F$ in a blackbox manner
Why Reductions?

- Easier constructions and proofs
- If $G$ obfuscated “in hardware”, still can be used to obfuscate $F$
- Theoretical interest: New connections between classes of functions
This Work

- Introduces relevant notions of reduction
- Reductions of some complex families to a simpler family ("point functions")
- Obfuscation of point functions in the "Random Oracle" model
There are two PPT oracle-machines $M$ and $N$ such that for every $F$ in $\mathcal{F}$ there is a $G$ in $\mathcal{G}$ such that $M^G = F$ and $N^F = G$.
Lemma:

If $F < G$ and $G$ obfuscatable
then $F$ obfuscatable
Proof: Intuition

- $\text{ob}_F\.exe = M^{\text{ob}_G\.exe}$

- **Ensure that giving $\text{ob}_G\.exe$ is OK:**
  - *Giving* $\text{ob}_G\.exe$ is "like" giving blackbox access to $G$
  - *Giving* blackbox access to $G$ is not more than giving blackbox access to $F$, because $G = N^F$
Proof: Sketch

- $\text{ob}_F\text{.exe} = M^{\text{ob}_G\text{.exe}}$

- For every adversary $A$ which takes $\text{ob}_F\text{.exe}$ show a “simulator” $S^F$

- Consider $A'$ which takes $\text{ob}_G\text{.exe}$, constructs $\text{ob}_F\text{.exe}$ and calls $A$ on that.

- Consider $S'$: behaves like $A'$, but needs oracle access to $G$

- $S^F$: run $S'$ with access to $N^F$
A simple family $G$ and a complex family $F$

Show $F < G$

Show how to obfuscate $G$ ($G$ non-trivial)

Lemma gives obfuscation of $F$
Simple families

- \( \mathcal{P} \) the family of point functions:
  \[ P_a(x) = 1 \text{ iff } x = a \]

- \( \mathcal{Q} \) point functions with output:
  \[ P_{a,b}(x) = b \text{ iff } x = a \]

- \( \mathcal{Q}^* \) multi-point functions with output:
  \[ P_{A,B}(x) = B_i \text{ iff } x = A_i \]
A more complex family

A Complex Access Control Mechanism:

- An unknown graph defining access to nodes
- Each edge has a password
- Start at start node
- Exponentially many valid access patterns
Obfuscating it

- Ideally would like to provide blackbox access to the access controller/secrets in the nodes.

- But what if the code is public?

- Keep the code obfuscated.
Elements of the Obfuscation/proof

- Probabilistic family $\mathcal{W}$: random keys to nodes
- ACM $< \mathcal{W}$ under an extended definition of “$<$”
- From extended Lemma: if the family obtained by fixing the random tape of $\mathcal{W}$ in every way obfuscatable, then ACM obfuscatable
- Fixing tape of $\mathcal{W}$ gives multi-point functions
Obfuscating point functions

- In the Random Oracle model
- RO a random function
- Both obfuscator and adversary get oracle access to it
- \( \text{ob}_F\.exe \) may be different from \( F \) with negligible probability (over choice of RO)
- \( | \Pr[A^{RO}(\text{ob}_F\.exe)=1] - \Pr[S^F=1] | < \text{negl} \)
Point function $P_a$: Store $RO(a)$

Point function with output $P_{a,b}$: Choose $r$ at random. Store $r$, $RO_1(r,a)$ and $b + RO_2(r,a)$

Multiple points: repeat above for each point with different $r$'s
Some Other Obfuscations

- Public constant size regular expressions with secret strings

- Public regular expression with secret obfuscatable languages, but giving access to the individual secret languages

- Neighbourhood checking on tree metrics
Obfuscations via Reductions

- All reductions to multi-point functions (or underlying obfuscable functions)
- No further use of random oracles
- Useful if the multi-point function primitive can be obfuscated say on hardware
To explore...

- More obfuscations and reductions
  - Algorithmic problems
- Obfuscations without random oracles
- More impossibilities?
- Alternate definitions?