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 $\mathcal{F}$ is obfuscatable if:

There is $\mathcal{O}$ such that for all F.exe in $\mathcal{F}$,

- $\mathcal{O}$(F.exe) = ob_F.exe has 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 $\mathcal{F}$, what A can find out about F from ob_F.exe, S can find out just from blackbox access to F.

$$| \Pr[A(\text{ob\_F.exe})=1] - \Pr[S^F=1] | < \text{negl}$$

Impossibility of Obfuscation

[BGIRSVY'01]

- There are unobfuscatable functions: in particular there are no universal obfuscators

- Unobfuscatable cryptographic schemes

- Low-complexity (TC^0) 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)) | f in $\mathcal{F}$, g in $\mathcal{G}$ } 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 $\mathcal{F}$ "reduces to" $\mathcal{G}$ and $\mathcal{G}$ obfuscatable then $\mathcal{F}$ also obfuscatable
- "Blackbox reductions": given any obfuscator for $\mathcal{G}$ give one for $\mathcal{F}$ in a blackbox manner

Why Reductions?

- Easier constructions and proofs
- If $\mathcal{G}$ obfuscated "in hardware", still can be used to obfuscate $\mathcal{F}$
- 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

$\mathcal{F} < \mathcal{G}$

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 $\mathrm{M}^{\mathrm{G}} = \mathrm{F}$ and $\mathrm{N}^{\mathrm{F}}=\mathrm{G}$

Using the Reduction

- Lemma:

If $\mathcal{F} < \mathcal{G}$ and $\mathcal{G}$ obfuscatable

then $\mathcal{F}$ obfuscatable

Proof: Intuition

- ob_F.exe = $M^{\text{ob\_G.exe}}$
- Ensure that giving ob_G.exe is OK:
  - Giving 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

- ob_F.exe = $M^{\text{ob\_G.exe}}$
- For every adversary A which takes ob_F.exe show a "simulator" S^F
  - Consider A' which takes ob_G.exe, constructs ob_F.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

Using Reductions

- A simple family $\mathcal{G}$ and a complex family $\mathcal{F}$
- Show $\mathcal{F} < \mathcal{G}$
- Show how to obfuscate $\mathcal{G}$ ($\mathcal{G}$ non-trivial)
- Lemma gives obfuscation of $\mathcal{F}$

Simple families

- $\mathcal{P}$ the family of point functions:

$$P_a(x) = 1 \text{ iff } x=a$$

- $\mathcal{Q}$ point functions with output:

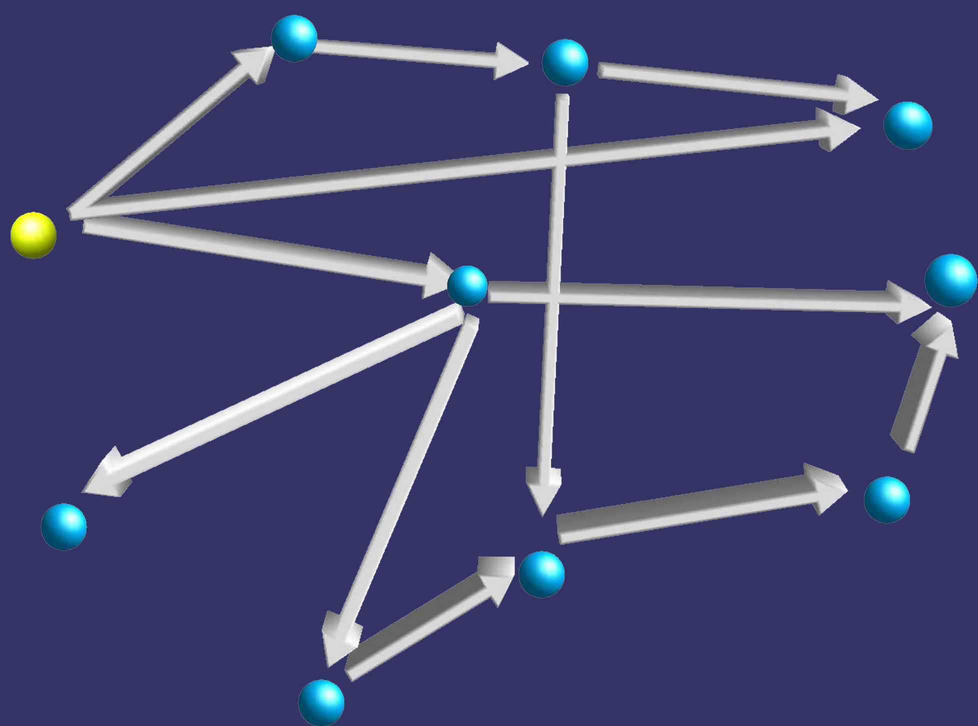
$$P_{a,b}(x) = b \text{ iff } x=a$$

- $\mathcal{Q}^*$ 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
- ob_F.exe may be different from F with negligible probability (over choice of RO)
- $|\ \Pr[A^{RO}(\text{ob\_F.exe})=1] - \Pr[S^{F}=1]\ | < \text{negl}$

Obfuscating point functions

- 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 obfus-catable 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 obfuscatable 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?

Thank You!