The State Of The Art In Program Obfuscation

Mike Rosulek
rosulek@cs.uiuc.edu

September 14, 2006
UIUC CS Theory Seminar

Outline

Introduction
- Motivation
- Defining Obfuscation

Negative Results
- *On the (Im)possibility of Obfuscating Programs* (B+01)

Positive Results
- *Positive Results & Techniques in Obfuscation* (LPS04)
- *On Obfuscating Point Functions* (W05)

Additional Topics
- Obfuscatability-Preserving Reductions
- Obfuscations In Context

Motivating Example

Alice is a new CS professor.

Motivating Example

Alice is a new CS professor.

- Discovers $O(n \log n)$ algorithm for integer factorization.

Motivating Example

Alice is a new CS professor.

- Discovers $O(n \log n)$ algorithm for integer factorization.
- Sends `factor.exe` to faculty mailing list, demands immediate tenure.

Motivating Example

Alice is a new CS professor.

- Discovers $O(n \log n)$ algorithm for integer factorization.
- Sends `factor.exe` to faculty mailing list, demands immediate tenure.
- Can dishonest faculty now steal her algorithm?

Motivating Example

Alice is a new CS professor.

- Discovers $O(n \log n)$ algorithm for integer factorization.
- Sends `factor.exe` to faculty mailing list, demands immediate tenure.
- Can dishonest faculty now steal her algorithm?

Ideally: Only reveal input-output functionality (oracle access).
Reality: Must send an actual implementation of the algorithm.

Simulation-based Definitions in Cryptography

When is a security definition the "right" one?

Simulation-based Definitions in Cryptography

When is a security definition the “right” one?

Cryptographic world	$\Longleftrightarrow$	**Ideal world**

Simulation-based Definitions in Cryptography

When is a security definition the "right" one?

Anything an adversary
can do here ...

Cryptographic world $\iff$ **Ideal world**

Simulation-based Definitions in Cryptography

When is a security definition the "right" one?

Anything an adversary can do here ...		*... could have been done here, too!*
Cryptographic world	$\Longleftrightarrow$	**Ideal world**

Simulation-based Definitions in Cryptography

When is a security definition the "right" one?

Anything an adversary can do here …		*… could have been done here, too!*
Cryptographic world	$\Longleftrightarrow$	**Ideal world**
Encryption scheme	$\Longleftrightarrow$	Secure channel

Simulation-based Definitions in Cryptography

When is a security definition the “right” one?

Anything an adversary can do here …		*… could have been done here, too!*
Cryptographic world	$\Longleftrightarrow$	**Ideal world**
Encryption scheme	$\Longleftrightarrow$	Secure channel
Zero-knowledge proof	$\Longleftrightarrow$	Trustworthy NP oracle

Simulation-based Definitions in Cryptography

When is a security definition the “right” one?

Anything an adversary can do here ... *... could have been done here, too!*

Cryptographic world	$\Longleftrightarrow$	**Ideal world**
Encryption scheme	$\Longleftrightarrow$	Secure channel
Zero-knowledge proof	$\Longleftrightarrow$	Trustworthy NP oracle
An obfuscation of f	$\Longleftrightarrow$	Oracle access to f

Notation

ϵ-close: For $f, g, \epsilon : \mathbb{N} \to [0,1]$, define

$$f(n) \approx_\epsilon g(n) \iff \left|f(n) - g(n)\right| \leq \epsilon(n)$$

Notation

ϵ-**close:** For $f, g, \epsilon : \mathbb{N} \to [0, 1]$, define

$$f(n) \approx_\epsilon g(n) \iff \big|f(n) - g(n)\big| \leq \epsilon(n)$$

Negligibly close: Write $f(n) \approx g(n)$ when $\epsilon(n) = n^{-\omega(1)}$.

Defining Obfuscation

Definition: $\mathcal{O}$ is an *obfuscator* for class $\mathcal{F}$ if:

Defining Obfuscation

Definition: $\mathcal{O}$ is an *obfuscator* for class $\mathcal{F}$ if:

- **Functionality**: $\forall F \in \mathcal{F}$, $\mathcal{O}(F)$ and F compute the same function, with probability ≈ 1 over randomness of $\mathcal{O}$.

Defining Obfuscation

Definition: $\mathcal{O}$ is an *obfuscator* for class $\mathcal{F}$ if:

- **Functionality**: $\forall F \in \mathcal{F}$, $\mathcal{O}(F)$ and F compute the same function, with probability ≈ 1 over randomness of $\mathcal{O}$.
- **Efficiency**: $\mathcal{O}(F)$ can be computed in poly($|F|$) time.

Defining Obfuscation

Definition: $\mathcal{O}$ is an *obfuscator* for class $\mathcal{F}$ if:

- **Functionality**: $\forall F \in \mathcal{F}$, $\mathcal{O}(F)$ and F compute the same function, with probability ≈ 1 over randomness of $\mathcal{O}$.
- **Efficiency**: $\mathcal{O}(F)$ can be computed in poly($|F|$) time.
- **Virtual Black-Box**: $\forall A \; \exists S \; \forall F \in \mathcal{F}$,

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

Defining Obfuscation

Definition: $\mathcal{O}$ is an *obfuscator* for class $\mathcal{F}$ if:

- **Functionality**: $\forall F \in \mathcal{F}$, $\mathcal{O}(F)$ and F compute the same function, with probability ≈ 1 over randomness of $\mathcal{O}$.
- **Efficiency**: $\mathcal{O}(F)$ can be computed in poly($|F|$) time.
- **Virtual Black-Box**: $\forall A\ \exists S\ \forall F \in \mathcal{F}$,

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

- **Polynomial Slowdown**: Running times of $\mathcal{O}(F)$ and F within poly factor (if encoded as TMs).

Learnable Functions

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

What if $\mathcal{F}$ is learnable via oracle queries?

Learnable Functions

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

What if $\mathcal{F}$ is learnable via oracle queries?

Simulator S:

- Query oracle to learn circuit for F.
- Run obfuscator to get $\mathcal{O}(F)$.
- Run $A(\mathcal{O}(F))$; perfect simulation.

Learnable Functions

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

What if $\mathcal{F}$ is learnable via oracle queries?

Simulator S:

- Query oracle to learn circuit for F.
- Run obfuscator to get $\mathcal{O}(F)$.
- Run $A(\mathcal{O}(F))$; perfect simulation.

Call $\mathcal{F}$ *trivially obfuscatable*.

On the (Im)possibility of Obfuscating Programs [B+01]

Main Result: Universal obfuscation impossible!

On the (Im)possibility of Obfuscating Programs [B+01]

Main Result: Universal obfuscation impossible!

Construct class $\mathcal{F}$ and predicate $P : \mathcal{F} \to \{0,1\}$ such that:

On the (Im)possibility of Obfuscating Programs [B+01]

Main Result: Universal obfuscation impossible!

Construct class $\mathcal{F}$ and predicate $P : \mathcal{F} \to \{0,1\}$ such that:

- Given *any* TM implementing $F \in \mathcal{F}$, easy to compute $P(F)$.
- Given oracle access to random $F \in \mathcal{F}$, hard to compute $P(F)$.

Unobfuscatable Class

For $\alpha, \beta \in \{0,1\}^n$ and $b \in \{0,1\}$, define:

$$A_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0^n & \text{else} \end{cases}$$

Unobfuscatable Class

For $\alpha, \beta \in \{0,1\}^n$ and $b \in \{0,1\}$, define:

$$A_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0^n & \text{else} \end{cases}$$

$$B_{\alpha,\beta,b}(M) = \begin{cases} b & \text{if } M(\alpha) = \beta \text{ (interpret } M \text{ as TM)} \\ 0 & \text{else} \end{cases}$$

Unobfuscatable Class

For $\alpha, \beta \in \{0,1\}^n$ and $b \in \{0,1\}$, define:

$$A_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0^n & \text{else} \end{cases}$$

$$B_{\alpha,\beta,b}(M) = \begin{cases} b & \text{if } M(\alpha) = \beta \text{ (interpret } M \text{ as TM)} \\ 0 & \text{else} \end{cases}$$

$$F_{\alpha,\beta,b}(i,x) = \begin{cases} A_{\alpha,\beta}(x) & \text{if } i = 1 \\ B_{\alpha,\beta,b}(x) & \text{if } i = 2 \end{cases}$$

Unobfuscatable Class

For $\alpha, \beta \in \{0,1\}^n$ and $b \in \{0,1\}$, define:

$$A_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0^n & \text{else} \end{cases}$$

$$B_{\alpha,\beta,b}(M) = \begin{cases} b & \text{if } M(\alpha) = \beta \text{ (interpret } M \text{ as TM)} \\ 0 & \text{else} \end{cases}$$

$$F_{\alpha,\beta,b}(i,x) = \begin{cases} A_{\alpha,\beta}(x) & \text{if } i = 1 \\ B_{\alpha,\beta,b}(x) & \text{if } i = 2 \end{cases}$$

Use $\mathcal{F} = \{F_{\alpha,\beta,b}\}_{\alpha,\beta,b}$, and predicate $P(F_{\alpha,\beta,b}) = b$.

Unobfuscatable Class (continued)

To compute $P(F_{\alpha,\beta,b}) = b$ given M that computes $F_{\alpha,\beta,b}$:

1. From M, construct $M_1(\cdot) \equiv M(1,\cdot) \equiv A_{\alpha,\beta}(\cdot)$.

Unobfuscatable Class (continued)

To compute $P(F_{\alpha,\beta,b}) = b$ given M that computes $F_{\alpha,\beta,b}$:

1. From M, construct $M_1(\cdot) \equiv M(1,\cdot) \equiv A_{\alpha,\beta}(\cdot)$.
2. From M, construct $M_2(\cdot) \equiv M(2,\cdot) \equiv B_{\alpha,\beta,b}(\cdot)$.

Unobfuscatable Class (continued)

To compute $P(F_{\alpha,\beta,b}) = b$ given M that computes $F_{\alpha,\beta,b}$:

1. From M, construct $M_1(\cdot) \equiv M(1,\cdot) \equiv A_{\alpha,\beta}(\cdot)$.
2. From M, construct $M_2(\cdot) \equiv M(2,\cdot) \equiv B_{\alpha,\beta,b}(\cdot)$.
3. Run $M_2(M_1) = b$.

Unobfuscatable Class (continued)

To compute $P(F_{\alpha,\beta,b}) = b$ given M that computes $F_{\alpha,\beta,b}$:

1. From M, construct $M_1(\cdot) \equiv M(1,\cdot) \equiv A_{\alpha,\beta}(\cdot)$.
2. From M, construct $M_2(\cdot) \equiv M(2,\cdot) \equiv B_{\alpha,\beta,b}(\cdot)$.
3. Run $M_2(M_1) = b$.

Can show that for all polytime S,

$$\Pr_{\alpha,\beta,b}[S^{F_{\alpha,\beta,b}}() = b] \approx 1/2$$

Unobfuscatable Class (continued)

To compute $P(F_{\alpha,\beta,b}) = b$ given M that computes $F_{\alpha,\beta,b}$:

1. From M, construct $M_1(\cdot) \equiv M(1,\cdot) \equiv A_{\alpha,\beta}(\cdot)$.
2. From M, construct $M_2(\cdot) \equiv M(2,\cdot) \equiv B_{\alpha,\beta,b}(\cdot)$.
3. Run $M_2(M_1) = b$.

Can show that for all polytime S,

$$\Pr_{\alpha,\beta,b}[S^{F_{\alpha,\beta,b}}() = b] \approx 1/2$$

Conclusion: $\mathcal{F}$ unobfuscatable.

Nontrivial Positive Results

Known only for *point functions*:

$$\delta_x(w) = \begin{cases} 1 & \text{if } w = x \\ 0 & \text{otherwise} \end{cases}$$

Nontrivial Positive Results

Known only for *point functions*:

$$\delta_x(w) = \begin{cases} 1 & \text{if } w = x \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta = \{\delta_x\}_{x \in \{0,1\}^*}$$

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^n \to \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^n \to \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

- On input w, if $R(w) = R(x)$, output 1, else output 0.

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^n \to \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

- On input w, if $R(w) = R(x)$, output 1, else output 0.

Blue values "hardwired" into this program.

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^n \rightarrow \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

- On input w, if $R(w) = R(x)$, output 1, else output 0.

Blue values "hardwired" into this program.

Obfuscator $\mathcal{O} : \delta_x \mapsto R(x)$

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^{2n} \to \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

- On input w, if $R(ws) = R(xs)$, output 1, else output 0.

Blue values "hardwired" into this program.

Obfuscator $\mathcal{O} : \delta_x \mapsto (s, R(xs))$ for random $s \in \{0,1\}^n$.

Positive Results & Techniques in Obfuscation [LPS04]

Obfuscator for Δ (in random oracle model):

Construction: Given random oracle $R : \{0,1\}^{2n} \to \{0,1\}^{3n}$, $x \in \{0,1\}^n$, obfuscate δ_x by:

- On input w, if $R(ws) = R(xs)$, output 1, else output 0.

Blue values "hardwired" into this program.

Obfuscator $\mathcal{O} : \delta_x \mapsto (s, R(xs))$ for random $s \in \{0,1\}^n$.

Functionality, polynomial slowdown, efficiency: straight-forward

Virtual Black-Box

Virtual black-box property

$$\forall A\ \exists S\ \forall F \in \mathcal{F},\ \Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^{F}() = 1]$$

Virtual Black-Box

Virtual black-box property for random oracle model:

$$\forall A \; \exists S \; \forall F \in \mathcal{F}, \; \Pr_{R}[A^{R}(\mathcal{O}^{R}(F)) = 1] \approx \Pr[S^{F}() = 1]$$

Virtual Black-Box

Virtual black-box property for random oracle model:

$$\forall A\ \exists S\ \forall F \in \mathcal{F},\ \Pr_{R}[A^{R}(\mathcal{O}^{R}(F)) = 1] \approx \Pr[S^{F}() = 1]$$

S could run A with any (contrived) oracle.

Virtual Black-Box

Virtual black-box property for random oracle model:

$$\forall A\ \exists S\ \forall F \in \mathcal{F},\ \Pr_{R}[A^{R}(\mathcal{O}^{R}(F)) = 1] \approx \Pr[S^{F}() = 1]$$

S could run A with any (contrived) oracle.

Called *programmable* random oracle model. Not very realistic!

On Obfuscating Point Functions [W05]

Can replace random oracle with hash function, if we use:

- Slightly nonstandard crypto assumption.
- Slightly weaker virtual black-box property.
- Nonuniform simulator.

On Obfuscating Point Functions [W05]

Can replace random oracle with hash function, if we use:

- Slightly nonstandard crypto assumption. (necessary)
- Slightly weaker virtual black-box property. (necessary)
- Nonuniform simulator. (necessary)

Cryptographic Assumption

Definition:

π is a *one-way permutation* if: $\forall A$ of size $s = \text{poly}(n)$,

$$\Pr_{x \in \{0,1\}^n}[A(\pi(x)) = x] \approx 0 \quad (\leq n^{-\omega(1)})$$

Cryptographic Assumption

Definition:
π is a *strong one-way permutation* if: $\exists c\ \forall A$ of size $s = \text{poly}(n)$,

$$\Pr_{x \in \{0,1\}^n}[A(\pi(x)) = x] \leq \frac{s^c}{2^n}$$

Strongest hardness conceivable (within poly factors).

Weakening the Virtual Black-Box Property

Virtual Black-Box: $\forall A\ \exists S\ \forall F \in \mathcal{F}$,

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \ \Pr[S^F() = 1]$$

Weakening the Virtual Black-Box Property

Weak Virtual Black-Box: $\forall A, \epsilon(\cdot) = 1/\text{poly}(\cdot)\ \exists S\ \forall F \in \mathcal{F}$,

$$\Pr[A(\mathcal{O}(F)) = 1] \approx_\epsilon \Pr[S^F() = 1]$$

Simulator depends on ϵ!

Hash Construction

Replace random oracle with hash of $x \in \{0,1\}^n$:

$$h(x; r_1, \ldots, r_{3n}) = \Big(\langle \pi(x), r_1 \rangle, \langle \pi^2(x), r_2 \rangle, \cdots, \langle \pi^{3n}(x), r_{3n} \rangle\Big)$$

Hash Construction

Replace random oracle with hash of $x \in \{0,1\}^n$:

$$h(x; r_1, \ldots, r_{3n}) = \Big(\langle \pi(x), r_1 \rangle, \langle \pi^2(x), r_2 \rangle, \cdots, \langle \pi^{3n}(x), r_{3n} \rangle \Big)$$

Obfuscate δ_x by:

- On input w, if $h(w; \vec{r}) = h(x; \vec{r})$, output 1, else output 0.

Obfuscator $\mathcal{O} : \delta_x \mapsto (\vec{r}, h(x; \vec{r}))$ for random $\vec{r}$

Hash Construction

Replace random oracle with hash of $x \in \{0,1\}^n$:

$$h(x; r_1, \ldots, r_{3n}) = \Big(\langle \pi(x), r_1 \rangle, \langle \pi^2(x), r_2 \rangle, \cdots, \langle \pi^{3n}(x), r_{3n} \rangle\Big)$$

Obfuscate δ_x by:

- On input w, if $h(w; \vec{r}) = h(x; \vec{r})$, output 1, else output 0.

Obfuscator $\mathcal{O} : \delta_x \mapsto (\vec{r}, h(x; \vec{r}))$ for random $\vec{r}$

Functionality, Efficiency, Polynomial-slowdown: straight-forward.

Weak Virtual Black-Box Property

Given adversary A and $\epsilon = 1/\text{poly}$, construct simulator S to get ϵ-close.

Weak Virtual Black-Box Property

Given adversary A and $\epsilon = 1/\text{poly}$, construct simulator S to get ϵ-close.

Idea: Simulate $\mathcal{O}(\delta_x) = (\vec{r}, h(x; \vec{r}))$ by random bits ($\mathcal{U}$).

Weak Virtual Black-Box Property

Given adversary A and $\epsilon = 1/\text{poly}$, construct simulator S to get ϵ-close.

Idea: Simulate $\mathcal{O}(\delta_x) = (\vec{r}, h(x; \vec{r}))$ by random bits ($\mathcal{U}$).

Main Lemma: $\forall A, \epsilon$, define:

$$L_{A,\epsilon} = \left\{ x \in \{0,1\}^n \;\middle|\; \Pr[A(\mathcal{O}(\delta_x)) = 1] \not\approx_\epsilon \Pr[A(\mathcal{U}) = 1] \right\}$$

Weak Virtual Black-Box Property

Given adversary A and $\epsilon = 1/\text{poly}$, construct simulator S to get ϵ-close.

Idea: Simulate $\mathcal{O}(\delta_x) = (\vec{r}, h(x; \vec{r}))$ by random bits $(\mathcal{U})$.

Main Lemma: $\forall A, \epsilon$, define:

$$L_{A,\epsilon} = \left\{ x \in \{0,1\}^n \;\middle|\; \Pr[A(\mathcal{O}(\delta_x)) = 1] \not\approx_\epsilon \Pr[A(\mathcal{U}) = 1] \right\}$$

Then $|L_{A,\epsilon}| \leq \text{poly}(n, 1/\epsilon)$.

Proof: Reduce to strong one-wayness of π.

Weak Virtual Black-Box Property (simulator)

Given A and ϵ, simulate $A(\mathcal{O}(\delta_x))$ using oracle access to δ_x:

1. For each $y \in L_{A,\epsilon}$: (hardwired; only poly many)
2. Query oracle: If $\delta_x(y) = 1$, then run $A(\mathcal{O}(\delta_y))$.
3. Otherwise, run $A(\mathcal{U})$.

Weak Virtual Black-Box Property (simulator)

Given A and ϵ, simulate $A(\mathcal{O}(\delta_x))$ using oracle access to δ_x:

1. For each $y \in L_{A,\epsilon}$: (hardwired; only poly many)
2. Query oracle: If $\delta_x(y) = 1$, then run $A(\mathcal{O}(\delta_y))$.
3. Otherwise, run $A(\mathcal{U})$.

Analysis:
Discover x? Can give perfect simulation $A(\mathcal{O}(\delta_x))$.

Weak Virtual Black-Box Property (simulator)

Given A and ϵ, simulate $A(\mathcal{O}(\delta_x))$ using oracle access to δ_x:

1. For each $y \in L_{A,\epsilon}$: (hardwired; only poly many)
2. Query oracle: If $\delta_x(y) = 1$, then run $A(\mathcal{O}(\delta_y))$.
3. Otherwise, run $A(\mathcal{U})$.

Analysis:

Discover x? Can give perfect simulation $A(\mathcal{O}(\delta_x))$.

Otherwise $x \notin L_{A,\epsilon}$, so $A(\mathcal{U})$ gives ϵ-close simulation.

Extensions of Positive Results:

Both positive results extend to point functions with output:

$$\delta_{x,y}(w) = \begin{cases} y & \text{if } w = x \\ \perp & \text{otherwise} \end{cases}$$

Obfuscatability-Preserving Reductions

[LPS04] define a reduction $\preceq$ such that if $\mathcal{F} \preceq \mathcal{G}$ and $\mathcal{G} \preceq \mathcal{F}$, then

$$\mathcal{F} \text{ obfuscatable} \iff \mathcal{G} \text{ obfuscatable}$$

Obfuscatability-Preserving Reductions

[LPS04] define a reduction $\preceq$ such that if $\mathcal{F} \preceq \mathcal{G}$ and $\mathcal{G} \preceq \mathcal{F}$, then

$$\mathcal{F} \text{ obfuscatable } \iff \mathcal{G} \text{ obfuscatable}$$

Open Problem: Sharper obfu-preserving reductions.

On the Impossibility of Obfuscation w/ Aux. Input [GK05]

Definition concerns obfuscated programs *in isolation*. Does security hold when adversary has other information?

$\forall A \; \exists S \; \forall F \in \mathcal{F}$,

$$\Pr[A(\mathcal{O}(F)) = 1] \approx \Pr[S^F() = 1]$$

On the Impossibility of Obfuscation w/ Aux. Input [GK05]

Definition concerns obfuscated programs *in isolation*. Does security hold when adversary has other information?

$\forall A\ \exists S\ \forall F \in \mathcal{F}, z \in \{0,1\}^{\text{poly}}$,

$$\Pr[A(\mathcal{O}(F), z) = 1] \approx \Pr[S^F(z) = 1]$$

[GK05] gives (conditional) impossibility results for several classes of functions.

On the Impossibility of Obfuscation w/ Aux. Input [GK05]

Definition concerns obfuscated programs *in isolation*. Does security hold when adversary has other information?

$\forall A\ \exists S\ \forall F \in \mathcal{F}, z \in \{0,1\}^{\mathsf{poly}}$,

$$\Pr[A(\mathcal{O}(F), z) = 1] \approx \Pr[S^F(z) = 1]$$

[GK05] gives (conditional) impossibility results for several classes of functions.

Open Problem: *Any* nontrivial obfuscations under this definition?

Composing Obfuscated Programs

Does security hold in presence of other obfuscated programs?

Composing Obfuscated Programs

Does security hold in presence of other obfuscated programs?

- Obfuscations of [LPS04] have *self-composable security*:

$$\Pr[A(\mathcal{O}(\delta_{x_1}), \ldots, \mathcal{O}(\delta_{x_m})) = 1] \approx \Pr[S^{\delta_{x_1}, \ldots, \delta_{x_m}}() = 1]$$

Composing Obfuscated Programs

Does security hold in presence of other obfuscated programs?

- Obfuscations of [LPS04] have *self-composable security*:

$$\Pr[A(\mathcal{O}(\delta_{x_1}), \ldots, \mathcal{O}(\delta_{x_m})) = 1] \approx \Pr[S^{\delta_{x_1}, \ldots, \delta_{x_m}}() = 1]$$

- Obfuscations of [W05] not known to self-compose.

Composing Obfuscated Programs

Does security hold in presence of other obfuscated programs?

- Obfuscations of [LPS04] have *self-composable security*:

$$\Pr[A(\mathcal{O}(\delta_{x_1}), \ldots, \mathcal{O}(\delta_{x_m})) = 1] \approx \Pr[S^{\delta_{x_1}, \ldots, \delta_{x_m}}() = 1]$$

- Obfuscations of [W05] not known to self-compose.
- All known *impossibility* results involve composing several obfuscations.

Composing Obfuscated Programs

Does security hold in presence of other obfuscated programs?

- Obfuscations of [LPS04] have *self-composable security*:

$$\Pr[A(\mathcal{O}(\delta_{x_1}),\ldots,\mathcal{O}(\delta_{x_m}))=1] \approx \Pr[S^{\delta_{x_1},\ldots,\delta_{x_m}}()=1]$$

- Obfuscations of [W05] not known to self-compose.
- All known *impossibility* results involve composing several obfuscations.

Open Problem: Find nontrivial obfuscations with self-composable security.

Any questions?

Thank you!