The Pseudorandom Oracle Model and Ideal Obfuscation

Aayush Jain  Rachel Lin  Ji Luo  Daniel Wichs

Carnegie Mellon University  UNIVERSITY of WASHINGTON  Northeastern University  NTT Research
Program Obfuscation (for Circuits)

Correctness.
Obf(C)(x) = C(x)

Security.
Obf(C) is “unintelligible”
Indistinguishability Obfuscation ($iO$)

$|C_0| = |C_1|$ and $\forall x: C_0(x) = C_1(x) \implies \text{Obf}(C_0) \approx \text{Obf}(C_1)$
Indistinguishability Obfuscation ($iO$)

$|C_0| = |C_1|$ and $\forall x: C_0(x) = C_1(x) \Rightarrow \text{Obf}(C_0) \approx \text{Obf}(C_1)$

✓ very powerful primitive

deniable encryption, short signatures, injective TDF, NIZK, OT [SW]
FHE [ABFGGTW] FE for P [GGHRSW] 2-round MPC [GGHR]
succinct garbled RAM [CH] (+ many more!)
Indistinguishability Obfuscation ($iO$)

\[ |C_0| = |C_1| \text{ and } \forall x: C_0(x) = C_1(x) \implies \text{Obf}(C_0) \approx \text{Obf}(C_1) \]

- **very powerful** primitive
  - deniable encryption, short signatures, injective TDF, NIZK, OT [SW]
  - FHE [ABFGGTW] FE for P [GGHRSW] 2-round MPC [GGHR]
  - succinct garbled RAM [CH] (+ many more!)

- **from well-studied assumptions** [JLS]
Indistinguishability Obfuscation ($iO$)

$|C_0| = |C_1|$ and $\forall x: C_0(x) = C_1(x) \implies \text{Obf}(C_0) \approx \text{Obf}(C_1)$

- very powerful primitive
- deniable encryption, short signatures, injective TDF, NIZK, OT
- FHE [ABFGGTW] FE for P [GGHRSW] 2-round MPC [GGHR]
- succinct garbled RAM [CH] (+ many more!)
- from well-studied assumptions [JLS]

⚠️ weak, unintuitive security guarantees
⚠️ unclear security for natural usages
⚠️ convoluted techniques for applications

🤔 What else to desire?
Natural Applications of Obfuscation

Cryptographic Program

secret-key encryption + obfuscation \(\rightarrow\) public-key encryption [DH]

\[
\begin{align*}
sk &= \text{skeK} \\
pk &= \text{Obf(skeEnc(skeK, .))}
\end{align*}
\]
Natural Applications of Obfuscation

Cryptographic Program

**secret-key encryption + obfuscation** $\Rightarrow$ **public-key encryption** [DH]

- $sk = \text{skeK}$
- $pk = \text{Obf}(\text{skeEnc}(\text{skeK}, \cdot))$

maybe $iO +$ techniques (e.g., puncturing)
Natural Applications of Obfuscation

Cryptographic Program

secret-key encryption + obfuscation $\Rightarrow$ public-key encryption \[\text{[DH]}\]

$sk = \text{skeK}$

$pk = \text{Obf(skeEnc(skeK, \cdot))}$

Non-Cryptographic Programs

software vulnerability patches

programs with copyright protection

machine learning models

...
Natural Applications of Obfuscation

Cryptographic Program

secret-key encryption + obfuscation $\Rightarrow$ public-key encryption $[DH]$

\[
\begin{align*}
\text{sk} &= \text{skeK} \\
\text{pk} &= \text{Obf(}\text{skeEnc(skeK, \cdot)}\text{)}
\end{align*}
\]

Non-Cryptographic Programs

software vulnerability patches
programs with copyright protection
machine learning models

...
Natural Applications of Obfuscation

Cryptographic Program

secret-key encryption + obfuscation $\Rightarrow$ public-key encryption [DH]

$sk = skeK$

$pk = Obf(skeEnc(skeK, \cdot))$

Non-Cryptographic Programs

software vulnerability patches

programs with copyright protection

machine learning models ...

Model parameters could leak sensitive training data.

$x \rightarrow_{\text{ML model}} \rightarrow \text{prediction}$

Want. $Obf(\text{ML model})$ hides training data.

(If they cannot be extracted from prediction...)
Simulation-Secure Obfuscation

\[ \tilde{C} \approx C(x) \]
Simulation-Secure Obfuscation

Ideal Obfuscation.
\[ \exists S \quad \forall C: \quad \text{Obf}(C) \approx S^C(1^{|C|}, 1^{|x|}). \]

Virtual Black-Box Obfuscation.
\[ \forall \text{ one-bit } \mathcal{A} \quad \exists S_{\mathcal{A}} \quad \forall C: \quad \mathcal{A}(\text{Obf}(C)) \approx S_{\mathcal{A}}^C(1^{|C|}, 1^{|x|}). \]
Simulation-Secure Obfuscation

Ideal Obfuscation.
\[ \exists S \quad \forall C: \quad \text{Obf}(C) \approx S(C(1^{\mid C \mid}, 1^{\mid x \mid})) \]

\[ \times \] impossible for unlearnable circuits

Virtual Black-Box Obfuscation.
\[ \forall \text{ one-bit } A \quad \exists S_A \quad \forall C: \quad A(\text{Obf}(C)) \approx S_A(C(1^{\mid C \mid}, 1^{\mid x \mid})) \]

\[ \times \] VBB not possible in general [BGIRSVY]
(contrived “self-eating” programs)
Simulation-Secure Obfuscation

Ideal Obfuscation.
\[ \exists S \forall C: \text{Obf}(C) \approx S^C(1^{|C|}, 1^{|x|}). \]

✅ strong, intuitive security guarantees

Virtual Black-Box Obfuscation.
\[ \forall \text{one-bit } A \exists S_A \forall C: A(\text{Obf}(C)) \approx S_A^C(1^{|C|}, 1^{|x|}). \]

⚠️ VBB not possible in general [BGIRSVY] (contrived “self-eating” programs)
Simulation-Secure Obfuscation

Ideal Obfuscation.
\[ \exists S \forall C: \text{Obf}(C) \approx S^C(1^{|C|}, 1^{|x|}). \]

Virtual Black-Box Obfuscation.
\[ \forall \text{one-bit } A \exists S_A \forall C: \text{Obf}(C) \approx S_A^C(1^{|C|}, 1^{|x|}). \]

✅ strong, intuitive security guarantees
✅ simple, intuitive designs in applications
✅ only path to certain (plausible) applications

DE-PIR [BIPW,LMW] FHE for RAM [HHWW,LMW]
VGB obfuscation [BC] FE with optimal Dec time [ACFQ]
OT from binary erasure channel [AIKNPPR]
public-coin diO, input-unbounded obfuscation for TM [IPS]
input-hiding obfuscation for evasive functions [BBCKPS]
extractable WE [GKPVZ] wiretap-channel coding [IKLS,IJLSZ]
refuting dream XOR lemma [BIKSW]

⚠️ VBB not possible in general [BGIRSVY]
(contrived “self-eating” programs)

strong protection for “non-cryptographic” programs
Current State of Obfuscation

- VBB/ideal
- impossible

programs
Current State of Obfuscation

programs

VBB/ideal impossible
VBB possible
Current State of Obfuscation

- mystery
- VBB/ideal
- impossible
- VBB possible
Current State of Obfuscation

programs

- VBB/ideal
- impossible
- VBB
- possible

mystery

applications

- cryptographic applications
Current State of Obfuscation

programs

mystery

VBB
possible

VBB/ideal
impossible

cryptographic
applications

iO suffices
Current State of Obfuscation

- Programs:
  - Mystery
  - VBB/ideal
  - Impossible
  - Possible

- Applications:
  - Needs VBB/ideal
  - Cryptographic applications
  - $i\mathcal{O}$ suffices
Motivation

Natural Heuristics. For natural programs, ideal obfuscation is possible! Natural applications of ideal obfuscation are plausible!
Motivation

Natural Heuristics. For **natural** programs, **ideal** obfuscation is **possible**! **Natural** applications of **ideal** obfuscation are **plausible**!

![Motivation Diagram](image-url)
Hash Functions

$\{0,1\}^* \xrightarrow{h(\cdot)} \{0,1\}^\lambda$

basic, useful primitive
Hash Functions

\[ \{0,1\}^* \xrightarrow{h(\cdot)} \{0,1\}^\lambda \]

**Basic, useful primitive**

- one-wayness
- second-preimage resistance
- collision resistance

\[ \text{Sign}(sk, m) \overset{\text{def}}{=} \text{Sign}_\lambda(sk, h(m)) \]

\[ \checkmark \text{UF-CMA } \iff \text{UF-CMA + CR} \]
Hash Functions

\[
\begin{align*}
\{0,1\}^* & \xrightarrow{h(\cdot)} \{0,1\}^\lambda \\
\end{align*}
\]

**basic, useful primitive**

- one-wayness
- second-preimage resistance
- collision resistance

\[
\text{Sign}(sk, m) \triangleq \text{Sign}_\lambda (sk, h(m))
\]

\[
\text{UF-CMA} \iff \text{UF-CMA} + \text{CR}
\]

\[
\text{Sign}(sk, m) \triangleq \text{TDP}^{-1} (sk, h(m))
\]

❓ What assumption for \( h \)?

intuition of security **beyond complexity assumptions**
Hash Functions: Idealization

Standard Model

\[ \{0,1\}^* \xrightarrow{h(\cdot)} \{0,1\}^\lambda \]

- one-wayness
- second-preimage resistance
- collision resistance

Random Oracle Model [BR]

\[ \{0,1\}^* \xrightarrow{RF} \{0,1\}^\lambda \]

\[ x \xrightarrow{h(x)} RF(x) \]
Hash Functions: Idealization

**Standard Model**

\[ \{0,1\}^* \xrightarrow{h(\cdot)} \{0,1\}^\lambda \]

- one-wayness
- second-preimage resistance
- collision resistance

**Random Oracle Model** \([BR]\)

\[ \{0,1\}^* \xrightarrow{RF} \{0,1\}^\lambda \]

- practically used and more efficient schemes
  - Schnorr signature \([Schnort]\)
  - RSA-OAEP \([BR]\)
  - TLS \([KPW, DFGS, DJ]\) ...

\[ x \rightleftharpoons h(x) = RF(x) \]
Hash Functions: Idealization

**Standard Model**

\[ \{0,1\}^* \xrightarrow{h(\cdot)} \{0,1\}^\lambda \]

- one-wayness
- second-preimage resistance
- collision resistance

**Random Oracle Model [BR]**

\[ \{0,1\}^* \xrightarrow{RF} \{0,1\}^\lambda \]

\[ x \xleftarrow{\text{practically used and more efficient schemes}} h(x) = RF(x) \]

- Schnorr signature [Schnorr]
- RSA-OAEP [BR]
- TLS [KPW, DFGS, DJ] …

⚠️ (contrived) uninstantiability results [CGH]

✅ proof in ROM better than no proof at all
  - problematic if cannot write proof in ROM
Hash Functions: Idealization

**Standard Model**

- \( \{0,1\}^* \rightarrow h(\cdot) \rightarrow \{0,1\}^\lambda \)

- one-wayness
- second-preimage resistance
- collision resistance
- correlation intractability

**Random Oracle Model** [BR]

- \( \{0,1\}^* \rightarrow RF \rightarrow \{0,1\}^\lambda \)

- \( h(x) = RF(x) \)

- **practically used** and **more efficient** schemes
  - Schnorr signature [Schnorr]
  - RSA-OAEP [BR]
  - TLS [KPW, DFGS, DJ]...

- **(contrived) uninstantiability results** [CGH]

- **proof in ROM** better than **no proof at all**
  - problematic if cannot write proof in ROM

- **precursor** to standard-model version
Question

CRHF → idealize → random oracle

iO → idealize → ideal obfuscation
CRHF $\xrightarrow{\text{idealize}}$ random oracle

RO($\cdot$) = $\text{Obf}(F(k, \cdot))$ for PRF $F$

$i\Omega$ $\xrightarrow{\text{idealize}}$ ideal obfuscation
Question

CRHF $\xrightarrow{\text{idealize}}$ random oracle

$RO(\cdot) = \text{Obf}(F(k, \cdot))$ for PRF $F$

$i\mathcal{O} \xrightarrow{\text{idealize}}$ ideal obfuscation
Question

CRHF $\xrightarrow{\text{idealize}}$ random oracle

$\text{RO}(\cdot) = \text{Obf}(F(k, \cdot))$ for PRF $F$

Black-box use of hash functions does not help building ideal obfuscation. [CKP]

$i\Omega \xrightarrow{\text{idealize}}$ ideal obfuscation
Question

CRHF $\xrightarrow{\text{idealize}}$ random oracle

Can non-black-box use of hash functions help? $\mathbb{Q}$

$\text{Black-box use of hash functions does not help building ideal obfuscation. [CKP]}$

$\text{RO}(\cdot) = \text{Obf}(F(k, \cdot))$ for PRF $F$

iO $\xrightarrow{\text{idealize}}$ ideal obfuscation

$\text{Can non-black-box use of hash functions help?}$

$\text{Black-box use of hash functions does not help building ideal obfuscation. [CKP]}$
Result

CRHF \[\rightarrow\] idealize \[\rightarrow\] morally ROM but more flexible pseudorandom oracle

\[\rightarrow\] idealize \[\rightarrow\] ideal obfuscation
Assuming polynomially secure FE for circuits, there exists ideal obfuscation in the pseudorandom oracle model.
Result

Theorem

Assuming polynomially secure FE for circuits, there exists ideal obfuscation in the pseudorandom oracle model.

- Justifies all downstream applications.
- Hope of $i\mathcal{O}$ from polynomial security.
- Multiple interpretations. (heuristics / bootstrapping / hardware tokens)

$CRHF \xrightarrow{\text{idealize}} \text{pseudorandom oracle}$

$morally \text{ ROM but more } \text{flexible}$

$i\mathcal{O} \xrightarrow{\text{idealize}} \text{ideal obfuscation}$
Take-Home Message

CRHF

Reduction of Heuristics

accept heuristics of ideal hash functions
(+ accept PrOM as a good model of it)
⇒ accept heuristics of ideal obfuscation

iO

ideализировать

morally ROM but more flexible

pseudorandom oracle

+ FE

ideализировать

ideal obfuscation
Take-Home Message

CRHF → idealize → morally ROM but more flexible pseudorandom oracle

Reduction of Heuristics

- Accept heuristics of ideal hash functions
  (+ accept PrOM as a good model of it)
  ⇒ Accept heuristics of ideal obfuscation

"Heuristically assuming ideal obfuscation is not “crazier” than heuristically assuming ideal hash functions."

iO → idealize → ideal obfuscation
Pseudorandom Oracle Model (PrOM)

Two aspects of the model.

- \( h \) looks like a random function
- \( h \) has (short) code
Pseudorandom Oracle Model (PrOMUX)

Two aspects of the model.

• $h$ looks like a random function
• $h$ has (short) code
Pseudorandom Oracle Model (PrOMUX)

Two aspects of the model.
- \( h \) looks like a random function
- \( h \) has (short) code

PrOMUX for PRF \( F \)

handle map

generate handle for PRF key \( k \)

handle \( h \)
Pseudorandom Oracle Model (PrΩM)

Two aspects of the model.

• $h$ looks like a random function
• $h$ has (short) code

PrΩM for PRF $F$

(handle map)

get handle for PRF key $k$
Pseudorandom Oracle Model (PrOMUX)

Two aspects of the model.
• $h$ looks like a random function
• $h$ has (short) code

PrOMUX for PRF $F$

get handle for PRF key $k$

handle map

handle $h$

evaluate $h$ at $x$

$h(x) = F(k, x)$
Pseudorandom Oracle Model (PrOM)

Two aspects of the model.
- $h$ looks like a random function
- $h$ has (short) code

PrOM for PRF $F$

$h(x) = F(k, x)$
Pseudorandom Oracle Model (PrOMIC)

Two aspects of the model.

- $h$ looks like a random function
- $h$ has (short) code

**PrOMIC for PRF $F$**

$\text{PrOMIC}$ for PRF $F$

$\text{ROM}$ evaluate $h$ at $x$

$h(x) = F(k, x)$

$\text{PrOMIC}$ get handle for PRF key $k$

$\text{handle map}$

$h$ handle $h$
Pseudorandom Oracle Model (PrOM)

Two aspects of the model.

• $h$ looks like a random function
• $h$ has (short) code

PrOM for PRF $F$

But $h(x)$ is not random if $k$ is present?!
Basic Recipe of Using PrOM
Basic Recipe of Using PrOM

\[ h \text{ for } k \leftarrow $, \quad \text{FHE/GC/FE}(k) \]
Basic Recipe of Using PrOM

\[ h(x) = F(k, x) \]
Basic Recipe of Using PrOM

\[ h \text{ for } k \leftarrow $, \quad \text{FHE/GC/FE}(k) \]

\[ h(x) = F(k, x) \]

⚠️ *h cannot be monitored or programmed*
Basic Recipe of Using PrOM

\[ h \text{ for } k \leftarrow $, \quad \text{FHE/GC/FE}(k) \]

\[ h(x) = F(k, x) \]

\[ h \text{ cannot be monitored or programmed} \]

\[ \approx \]

\[ h \text{ for } k \leftarrow $, \quad \text{Sim}(\{F(k, x_i)\}_i) \]

\[ h(x) = F(k, x) \]

(FHE/GC/FE security)
Basic Recipe of Using PrOM

$h$ for $k \leftarrow \$, FHE/GC/FE($k$)

$h(x) = F(k, x)$

$h$ cannot be monitored or programmed

$h$ for $k \leftarrow \$, Sim($\{F(k, x_i)\}_i$)

$h(x) = F(k, x)$

(FHE/GC/FE security)

$h$, Sim($\{RF(x_i)\}_i$)

$h(x) = RF(x)$

(PRF security of $F$)
Basic Recipe of Using PrOM

1. **FHE/GC/FE Security**
   - $h(k) \leftarrow \$, FHE/GC/FE($k$)
   - $h(x) = F(k, x)$
   - $h$ cannot be monitored or programmed

2. **FHE/GC/FE Security**
   - $h(k) \leftarrow \$, Sim($\{F(k, x_i)\}_{i}$)
   - $h(x) = F(k, x)$
   - (FHE/GC/FE security)

3. **PRF Security of $F$**
   - $h$, Sim($\{RF(x_i)\}_{i}$)
   - $h(x) = RF(x)$
   - $h$ can be monitored and programmed
Basic Recipe of Using PrOM

1. For $k \leftarrow \$, $h(x) = F(k, x)$
   - Limited Use of Code
   - must $\approx$ hybrid with only black-box use
   - $h$ cannot be monitored or programmed

2. For $k \leftarrow \$, Sim($\{F(k, x_i)\}_i$)
   - $h(x) = F(k, x)$
   - (FHE/GC/FE security)

3. $h$, Sim($\{RF(x_i)\}_i$
   - $h(x) = RF(x)$
   - (PRF security of $F$)
   - $h$ can be monitored and programmed
Instantiating $\text{PrOM}$

**Rule of Thumb.** good for ROM $\Rightarrow$ good for $\text{PrOM}$
Instantiating PrOМ

**Rule of Thumb.** good for ROM $\implies$ good for PrOМ

**Example.** Let $h = k$ be random and $F(k, x) = \text{SHA3}(k \parallel x)$

**Rationale.** Good hash functions are “self-obfuscated PRF”.
Instantiating \( \Pr\mathcal{O}M \)

**Rule of Thumb.** good for ROM \( \implies \) good for \( \Pr\mathcal{O}M \)

**Example.** Let \( h = k \) be random and \( F(k, x) = \text{SHA3}(k \parallel x) \)

**Rationale.** Good hash functions are “self-obfuscated PRF”.

\( \Pr\mathcal{O}M \) does **not demand more than ROM**.

— the only meaningful thing to do with a prefix is to evaluate the function at its extensions.
Ideal Obfuscation

$\mathcal{O}$: oracle of idealized model (e.g., $\Pr^{\mathcal{O}^F}$)

$\mathcal{C}$ (no oracle)

$\tilde{\mathcal{C}}$ (oracle)

$\forall C, x: \Pr[\tilde{\mathcal{C}} \leftarrow \text{Obf}^{\mathcal{O}}(C) : \tilde{\mathcal{C}}^{\mathcal{O}}(x) = C(x)] = 1.$
Ideal Obfuscation: Security

Real

Simulation
Ideal Obfuscation: Security
Ideal Obfuscation: Security

Real

Obf^{O}

Obf^{O}(C)

Simulation

O
Ideal Obfuscation: Security

\[ \mathcal{O} \]

\[ \text{Real} \]

\[ \text{Simulation} \]

\[ \text{Obf}^\mathcal{O} \]

\[ C \]

\[ \text{Obf}^\mathcal{O} (C) \]
Ideal Obfuscation: Security

Real

\[ \mathcal{O} \]

\[ \text{Obf}^\mathcal{O} \]

\[ \mathcal{O} \]

\[ \text{Obf}^\mathcal{O} (C) \]

Simulation

\[ \mathcal{S}_{\text{pre}} \]

\[ \mathcal{S}_{\text{Obf}} \]

\[ \mathcal{S}_{\text{post}} \]

simulator
Ideal Obfuscation: Security

Real

$\mathcal{O}$

Obf$^\mathcal{O}$

Obf$^\mathcal{O} (C)$

Simulation

$\mathcal{S}_{pre}$

$\mathcal{S}_{Obf}$

$\mathcal{S}_{post}$

simulator
Ideal Obfuscation: Security

Real

\[ O \]

\[ \text{Obf}^{O} \]

\[ O \]

\[ C \leftrightarrow \text{Obf}^{O}(C) \]

Simulation

\[ \mathcal{S}_{\text{pre}} \]

\[ \mathcal{S}_{\text{Obf}} \]

\[ \mathcal{S}_{\text{post}} \]

simulator

\[ C \]

\[ C_{1}^{1|C|, 1|x|} \]
Ideal Obfuscation: Security

Real

$O$

$\text{Obf}^{O}$

$O$

\[ \text{Obf}^{O} (C) \]

Simulation

$\mathcal{S}_{\text{pre}}$

$\mathcal{S}_{\text{Obf}}$

$\mathcal{S}_{\text{post}}$

simulator

$C$

$\{1^{|C|}, 1^{|x|} \}$
 Ideal Obfuscation: Security

Real

\[ \mathcal{O} \]

\[ \text{Obf}^\mathcal{O} \]

\[ \mathcal{O} \]

\[ \text{Obf}^\mathcal{O}(C) \]

Simulation

\[ \mathcal{S}_{\text{pre}} \]

\[ \mathcal{S}_{\text{Obf}} \]

\[ \mathcal{S}_{\text{post}} \]

simulator

\[ C \]

\[ 1^{|C|}, 1^{|x|} \]
Ideal Obfuscation: Security

Real

Simulation

$\mathcal{O}$
$\text{Obf}^\mathcal{O}$
$\mathcal{O}$

$\mathcal{O}$

$\mathcal{O}$

$simulated \tilde{\mathcal{C}}$

$\mathcal{S}_{\text{pre}}$
$\mathcal{S}_{\text{Obf}}$
$\mathcal{S}_{\text{post}}$

simulator

$\mathcal{C}$

$\text{Obf}^\mathcal{O}(C)$

$\mathcal{C}$

$1^{|c|}, 1^{|x|}$

$\mathcal{C}$
Ideal Obfuscation: Security

\[
\begin{align*}
\mathcal{O} & \quad \xymatrix{\ar@{~>}[r] & \mathcal{O}} \\
\text{Obf}^\mathcal{O} & \quad \xymatrix{\ar@{~>}[r] & \text{Obf}^\mathcal{O}(C)} \\
\mathcal{O} & \quad \xymatrix{\ar@{~>}[r] & \mathcal{O}} \\
\end{align*}
\]

\[
\begin{align*}
\mathcal{S}_\text{pre} & \quad \xymatrix{\ar@{~>}[r] & \mathcal{S}_\text{Obf}} \\
\mathcal{S}_\text{post} & \quad \xymatrix{\ar@{~>}[r] & \mathcal{S}_\text{Obf}} \\
\text{simulator} & \quad \xymatrix{\ar@{~>}[r] & \mathcal{S}_\text{Obf}} \\
\end{align*}
\]

\[
\begin{align*}
1^{|c|}, 1^{|x|} & \quad \xymatrix{\ar@{~>}[r] & C} \\
\end{align*}
\]
Ideal Obfuscation: Security

Real

\( \mathcal{O} \)

\( \text{Obf}^{\mathcal{O}} \)

\( \mathcal{O} \)

\( \mathcal{O} \)

\( \text{Obf}^{\mathcal{O}}(C) \)

Simulation

\( \mathcal{S}_{\text{pre}} \)

\( \mathcal{S}_{\text{Obf}} \)

\( \mathcal{S}_{\text{post}} \)

simulator

\( C \)

\( 1^{|C|}, 1^{|x|} \)
Ideal Obfuscation: Security

\[(\mathcal{O}, \text{Obf}^\mathcal{O}(C), \mathcal{O}) \approx (\mathcal{S}_{\text{pre}}, \mathcal{S}_{\text{Obf}}^C, \mathcal{S}_{\text{post}}^C)\]
(Standard-Model) Functional Encryption
(Standard-Model) Functional Encryption

Setup

msk

mpk
(Standard-Model) Functional Encryption

\[ f \rightarrow \text{KeyGen} \]

\[ \text{Setup} \rightarrow \text{mpk} \]

\[ \text{msk} \rightarrow \text{KeyGen} \]

\[ \text{sk}_f \rightarrow \text{KeyGen} \]
(Standard-Model) Functional Encryption
(Standard-Model) Functional Encryption

Setup → KeyGen

msk → KeyGen

mpk → Enc

x → Enc

Enc → Dec

sk_f → Dec

ct(x) → Dec

Dec → f(x)

f → KeyGen
(Standard-Model) Functional Encryption

Setup → KeyGen (mpk) → Dec (ct(x)) → Enc (x) → KeyGen (msk) → Setup

\[ f(x) \]

**efficiency**

\[ T_{Enc} = O(|f|^{1-\alpha} + |x|^{2-\alpha}) \]
(Standard-Model) Functional Encryption

- **Setup**: $\text{Setup}(\text{msk})$ to $\text{mpk}$
- **KeyGen**: $\text{KeyGen}(\text{msk})$ to $\text{sk}_f$
- **Enc**: $\text{Enc}(\text{mpk}, \text{sk}_f, x)$
- **Dec**: $\text{Dec}(\text{mpk}, \text{sk}_f, \text{ct}(x))$

**efficiency**: $T_{\text{Enc}} = O(|f|^{1-\alpha} + |x|^{2-\alpha})$

**Security**: if $f(x_0) = f(x_1)$ then $(\text{mpk, sk}_f, \text{ct}(x_0)) \approx (\text{mpk, sk}_f, \text{ct}(x_1))$

**adaptive**: $x_0, x_1$ chosen after seeing $\text{mpk, sk}_f$
(Standard-Model) Functional Encryption

KeyGen \(\rightarrow\) Setup

\(\text{mpk} \rightarrow \text{Enc} \rightarrow \text{Dec} \rightarrow f(x)\)

\(\text{msk} \rightarrow \text{KeyGen} \rightarrow \text{sk}_f\)

Efficiency:

\[ T_{\text{Enc}} = O(|f|^{1-\alpha} + |x|^{2-\alpha}) \]

Adaptive: \(x_0, x_1\) chosen after seeing \(\text{mpk}, \text{sk}_f\)

Security:

if \(f(x_0) = f(x_1)\) then \((\text{mpk}, \text{sk}_f, \text{ct}(x_0)) \approx (\text{mpk}, \text{sk}_f, \text{ct}(x_1))\)

Fact:

Such FE follows from “obfuscation-minimum” FE, which can be based on well-studied assumptions.

[ABSV, GVW, GS, LM, AJS, BV, AS, KNTY, JLL, JLS]
perfect binary tree (leaf $\Leftrightarrow$ input)

root $\varepsilon$

leaf $x$
perfect binary tree (leaf ⇔ input)

\[ \text{root } \varepsilon \]

\[ \text{node } \chi \)

(root-to-node path)

\[ \text{leaf } x \]

\[ \text{ct}_\chi = \text{FE ciphertext of } (C, \chi, \ldots) \text{ for } \chi \in \{0,1\}^{\leq D} \]
\( iO \) from FE: Traversal and Evaluation

\[
ct^{\chi}(C, \chi, \ldots) \quad \text{traverse/evaluate} = \text{decrypt using FE key } sk_f
\]

\[
ct^{\varepsilon}
\]
$iO$ from FE: Traversal and Evaluation

$ct_\chi(C, \chi, ...) \quad \text{traverse/evaluate} = \text{decrypt using FE key } sk_f$
iO from FE: Traversal and Evaluation

\[ ct_\chi (C, \chi, ...) \]

\[ \text{traverse/evaluate} = \text{decrypt using FE key} \ sk_f \]
$iO$ from FE: Traversal and Evaluation

$ct_{\chi}(C, \chi, \ldots)$

traverse/evaluate = decrypt using FE key $sk_f$

Diagram:
- $ct_\varepsilon$
- $ct_0$
- $ct_1$
- $ct_{\chi||0}$
- $ct_{\chi||1}$
iO from FE: Traversal and Evaluation

$ct_{\chi} (C, \chi, ...) \quad \text{traverse/evaluate} = \text{decrypt using FE key sk}_f$
\[ ct_\chi(C, \chi, \ldots) \quad \text{traverse/evaluate} = \text{decrypt using FE key sk}_f \]
iO from FE: Traversal and Evaluation

\[ ct_\chi (C, \chi, ...) \]

\[ \text{traverse/evaluate} = \text{decrypt using FE key sk}_f \]

\[ f(C, \chi, ...) = (ct_{\chi\|0}, ct_{\chi\|1}) \text{ when } |\chi| < D \]
From FE: Traversal and Evaluation

\[ \text{ct}_\chi (C, \chi, \ldots) \]

\[ \text{traverse/evaluate} = \text{decrypt using FE key sk}_f \]

\[ f(C, \chi, \ldots) = (\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1) \text{ when } |\chi| < D \]

\[ f(C, \chi, \ldots) = C(\chi) \text{ when } |\chi| = D \]
$iO$ from FE: Traversal and Evaluation

```
ct_\chi (C, \chi, ...)
```

traverse/evaluate = decrypt using FE key $sk_f$

```
ct_\varepsilon
c_t_0
c_t_1
ct_\chi
```

PRG seed / PRF key & information for proof

```
f(C, \chi, ...) = (ct_{\chi\|0}, ct_{\chi\|1}) \text{ when } |\chi| < D
```

needs FE. Enc randomness

```
f(C, \chi, ...) = C(\chi) \text{ when } |\chi| = D
```
\[ i\Omega \text{ from FE} \]

\[ ct_{\chi}(C, \chi) \]

\[ sk_f: (C, \chi) \mapsto (ct_{\chi\|0}, ct_{\chi\|1}) / C(\chi) \]

\[ \text{Obf}(C) = (sk_f, ct_{\varepsilon}) \]
$iO$ from FE

$$ct_{\chi}(C, \chi)$$

$$sk_f: (C, \chi) \mapsto (ct_{\chi\|0}, ct_{\chi\|1})/C(\chi)$$

$$\text{Obf}(C) = (sk_f, ct_{\varepsilon})$$

\[\text{could evaluate at many inputs.}\]
iO from FE

\[ \text{ct}_\chi(C, \chi) \]

\[ \text{sk}_f: (C, \chi) \mapsto (\text{ct}_{\chi|0}, \text{ct}_{\chi|1})/C(\chi) \]

\[ \text{Obf}(C) = (\text{sk}_f, \text{ct}_\varepsilon) \]

could evaluate at many inputs. Do **not know where** it explores.

**Simulation.** Cannot hardwire **all** evaluations into **short** \(\text{Obf}(C)\).

**Proof.** Must go over **all** \(x\), incurring **exponential loss**.
\( iO \) from FE

\[
sk_f: (C, \chi) \mapsto (ct_{\chi\|0}, ct_{\chi\|1})/C(\chi)
\]

\[
Obf(C) = (sk_f, ct_{\varepsilon})
\]

could evaluate at many inputs. Do **not know where** it explores.

**Simulation.** Cannot hardwire all evaluations into short \( Obf(C) \).

**Proof.** Must go over all \( x \), incurring exponential loss.

**Idealized models could help?**
Simplified Ideas in ROM

\[ \text{ct}_{\chi}(C, \chi) \quad \text{sk}_f: (C, \chi) \mapsto h(\chi) \oplus ((\text{ct}_{\chi||0}, \text{ct}_{\chi||1})/C(\chi)) \]

ct_\varepsilon
Simplified Ideas in ROM

\[ \text{ct}_\chi (C, \chi) \quad \text{sk}_f : (C, \chi) \mapsto h(\chi) \oplus (\text{ct}_\chi^{\parallel 0}, \text{ct}_\chi^{\parallel 1}) / C(\chi) \]
Simplified Ideas in ROM

\[ \text{ct}_\chi(C, \chi) \quad \text{sk}_f: (C, \chi) \mapsto h(\chi) \oplus (\text{ct}_{\chi \parallel 0}, \text{ct}_{\chi \parallel 1}) / C(\chi) \]

must query \( h(\varepsilon) \) and XOR to get \( \text{ct}_\varepsilon \)
Simplified Ideas in ROM

$$ct_{\chi}(C, \chi) \quad \text{sk}_f: (C, \chi) \mapsto h(\chi) \oplus (ct_{\chi\|0}, ct_{\chi\|1})/C(\chi)$$

must query $h(\varepsilon)$ and XOR to get $ct_0$ and $ct_1$.
Simplified Ideas in ROM

$$\text{ct}_\chi(C, \chi)$$

$$\text{sk}_f : (C, \chi) \mapsto h(\chi) \oplus ((\text{ct}_\chi\|0, \text{ct}_\chi\|1)/C(\chi))$$

must query $h(\varepsilon)$ and XOR to get

$$\text{ct}_0 \quad \text{ct}_1$$

$$\text{ct}_\chi \quad \text{ct}_\chi\|0 \quad \text{ct}_\chi\|1$$
Simplified Ideas in ROM

\[ \text{ct}_\chi(C, \chi) \quad \quad \text{sk}_f(C, \chi) \mapsto h(\chi) \oplus \left( (\text{ct}_{\chi\parallel0}, \text{ct}_{\chi\parallel1}) / C(\chi) \right) \]

must query \( h(\varepsilon) \)
and XOR to get

\( \text{ct}_\varepsilon \)

\( \text{ct}_0 \)
\( \text{ct}_1 \)

must query \( h(\chi) \)
and XOR to get

\( \text{ct}_\chi \)

\( \text{ct}_{\chi\parallel0} \)
\( \text{ct}_{\chi\parallel1} \)
Simplified Ideas in ROM

\[
\text{ct}_\chi(C, \chi) \quad \text{sk}_f : (C, \chi) \mapsto h(\chi) \oplus ((\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi))
\]

must query \(h(\varepsilon)\) and XOR to get

\[
\begin{align*}
\text{ct}_\varepsilon & \quad \text{ct}_0 \\
\text{ct}_\chi & \quad \text{ct}_1
\end{align*}
\]

must query \(h(\chi)\) and XOR to get

\[
\begin{align*}
\text{ct}_{\chi \parallel 0} & \quad \text{ct}_{\chi \parallel 1}
\end{align*}
\]
Simplified Ideas in ROM

\[ ct_\chi(C, \chi) \quad \text{sk}_f: (C, \chi) \mapsto h(\chi) \oplus ((ct_{\chi\|0}, ct_{\chi\|1})/C(\chi)) \]

must query \( h(\varepsilon) \) and XOR to get \( ct_0 \) and \( ct_1 \)

must query \( h(\chi) \) and XOR to get \( ct_{\chi\|0} \) and \( ct_{\chi\|1} \)
Simplified Ideas in ROM

\[ \text{ct}_\chi(C, \chi) \]

\[ \text{sk}_f: (C, \chi) \mapsto h(\chi) \oplus ((\text{ct}_\chi \| 0, \text{ct}_{\chi \| 1}) / C(\chi)) \]

- must query \( h(\varepsilon) \) and XOR to get \( \text{ct}_\varepsilon \)
- must query \( h(\chi) \) and XOR to get \( \text{ct}_\chi \)
- must query \( h(x) \) and XOR to get \( C(x) \)
Simplified Ideas in ROM

\[ ct_\chi(C, \chi) \quad \text{sk}_f : (C, \chi) \mapsto h(\chi) \oplus (ct_{\chi\|0}, ct_{\chi\|1})/C(\chi) \]

- must query \( h(\varepsilon) \) and XOR to get \( ct_\varepsilon \)
- must query \( h(\chi) \) and XOR to get \( ct_\chi \)
- must query \( h(x) \) and XOR to get \( C(x) \)

\((ct_{\chi\|0}, ct_{\chi\|1})/C(\chi)\) is hidden if \( h(\chi) \) is not queried.
Simplified Ideas in ROM

$$\text{ct}_\chi(C, \chi)$$

$$\text{sk}_f : (C, \chi) \mapsto h(\chi) \oplus ((\text{ct}_\chi \| 0, \text{ct}_\chi \| 1) / C(\chi))$$

must query $h(\varepsilon)$ and XOR to get $\text{ct}_\varepsilon$

must query $h(\chi)$ and XOR to get $\text{ct}_\chi$

$(\text{ct}_\chi \| 0, \text{ct}_\chi \| 1) / C(\chi)$ is hidden if $h(\chi)$ is not queried.

Observe RO queries to know exploration path.

Program RO responses to hardwire $C(x)$. 
Simplified Ideas in ROM

\[ \text{ct}_\chi(C, \chi) \quad \text{sk}_f : (C, \chi) \rightarrow h(\chi) \oplus ((\text{ct}_{\chi \| 0}, \text{ct}_{\chi \| 1})/C(\chi)) \]

\( \times \) cannot query RO in circuit sent to FE. KeyGen

\( (\text{ct}_{\chi \| 0}, \text{ct}_{\chi \| 1})/C(\chi) \) is hidden if \( h(\chi) \) is not queried.

Observe RO queries to know exploration path.

Program RO responses to hardwire \( C(x) \).
Simplified Fix in PrOM

\[ \text{ct}_\chi(C, \chi, k) \quad \text{sk}_f: (C, \chi, k) \mapsto F(k, \chi) \oplus ((\text{ct}_\chi \parallel_0, \text{ct}_\chi \parallel_1)/C(\chi)) \]

\[ \tilde{C} = (\text{sk}_f, \text{ct}_\epsilon, h) \]
Simplified Fix in PrOM

\[ \text{ct}_\chi(C, \chi, k) \]

\[ \text{sk}_f: (C, \chi, k) \mapsto F(k, \chi) \oplus (\text{ct}_{\chi \| 0}, \text{ct}_{\chi \| 1})/C(\chi) \]

\[ \tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h) \]
Simplified Fix in **PrOM**

\[
\text{ct}_\chi(C, \chi, k) \quad \text{sk}_f: (C, \chi, k) \mapsto F(k, \chi) \oplus \left((\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi)\right)
\]

\[
\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h) \quad \text{Dec}(\text{sk}_f, \text{ct}_\chi) \oplus h(\chi) = (\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi)
\]

**obfuscator can use code of** \(F\) **with** \(k\)

**evaluator can call evaluation oracle with** \(h\)**
Simplified Simulator

\[
\text{ct}_\chi \left\{ C, \chi, k \right. \\
\sigma_\chi \right.
\]

\[
\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h)
\]

\[
\text{sk}_f \left\{ \begin{array}{l}
(C, \chi, k) \mapsto F(k, \chi) \oplus ((\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi)) \\
\sigma_\chi \mapsto G(\sigma_\chi)
\end{array} \right.
\]

\[
\text{Dec}(\text{sk}_f, \text{ct}_\chi) \oplus h(\chi) = (\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi)
\]

**Dec invariant** preserved during simulation
Simplified Simulator

\[
\text{ct}_\chi \begin{cases} C, \chi, k \\ \sigma_\chi \end{cases} \quad \text{sk}_f \begin{cases} (C, \chi, k) \mapsto F(k, \chi) \oplus (\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi) \\ \sigma_\chi \mapsto G(\sigma_\chi) \end{cases}
\]

\[\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h)\]

\[
\text{Dec}(\text{sk}_f, \text{ct}_\chi) \oplus h(\chi) = (\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi)
\]

**Dec invariant** preserved during simulation

**Simulation.**

- \(\text{ct}_\chi\) is FE ciphertext of \(\sigma_\chi\)

- \(h(\chi)\) is programmed as \(G(\sigma_\chi) \oplus ((\text{ct}_\chi \parallel 0, \text{ct}_\chi \parallel 1)/C(\chi))\)
Simplified Simulator

\[
\text{ct}_\chi \begin{cases} 
C, \chi, k \\
\sigma_{\chi}
\end{cases}
\quad \text{sk}_f \begin{cases} 
(C, \chi, k) \mapsto F(k, \chi) \oplus ((\text{ct}_{\chi\|0}, \text{ct}_{\chi\|1})/C(\chi)) \\
\sigma_{\chi} \mapsto G(\sigma_{\chi})
\end{cases}
\]

\[\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h) \quad \text{Dec}(\text{sk}_f, \text{ct}_\chi) \oplus h(\chi) = (\text{ct}_{\chi\|0}, \text{ct}_{\chi\|1})/C(\chi)\]

**Dec invariant** preserved during simulation

Simulation.

- \( \text{ct}_\chi \) is FE ciphertext of \( \sigma_{\chi} \)
  \[\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h) \text{ does not use } C\]

- \( h(\chi) \) is programmed as \( G(\sigma_{\chi}) \oplus ((\text{ct}_{\chi\|0}, \text{ct}_{\chi\|1})/C(\chi)) \)
  \[C(x) \text{ is queried upon query to } h(x)\]
Simplified Simulator

$$\text{ct}_\chi \begin{cases} C, \chi, k \\ \sigma_\chi \end{cases}$$

$$\text{sk}_f \begin{cases} (C, \chi, k) \mapsto F(k, \chi) \oplus ((\text{ct}_\chi \| 0, \text{ct}_\chi \| 1)/C(\chi)) \\ \sigma_\chi \mapsto G(\sigma_\chi) \end{cases}$$

$$\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h)$$

$$\text{Dec}(\text{sk}_f, \text{ct}_\chi) \oplus h(\chi) = (\text{ct}_\chi \| 0, \text{ct}_\chi \| 1)/C(\chi)$$

**Dec invariant** preserved during simulation

**Simulation.**

- **ct}_\chi is FE ciphertext of} \sigma_\chi\]

  $$\tilde{C} = (\text{sk}_f, \text{ct}_\varepsilon, h) \text{ does not use } C$$

- **h(\chi) is programmed as} G(\sigma_\chi) \oplus ((\text{ct}_\chi \| 0, \text{ct}_\chi \| 1)/C(\chi))$$

  $$C(x) \text{ is queried upon query to } h(x)$$

⚠️ More tricks in actual scheme to make proof work
Pseudorandom Oracle Model
(novel model for \textit{ideal} hash functions \textit{with code})

+ Functional Encryption

\begin{itemize}
  \item \textbf{Future. Other uses of PrOM?}
  \item \textbf{Ideal Obfuscation}
\end{itemize}

Thanks!

ia.cr/2022/1204
luoji@cs.washington.edu / luoji.bio
Alternative Interpretations

**Hardware.** tokens for PRF (implements PrO\(M\))
\[\Rightarrow\] ideal obfuscation for all circuits

**Bootstrapping.** ideal obfuscation for PRF (candidate: hash functions)
\[\Rightarrow\] ideal obfuscation for all circuits
Alternative Interpretations

**Hardware.** tokens for PRF (implements PrOM) → ideal obfuscation for all circuits

**Bootstrapping.** ideal obfuscation for PRF (candidate: hash functions) → ideal obfuscation for all circuits

**Previous.**
- tokens for more complex functionality [GISVW, DMMN, BCGHKR, NFRCLSG]
- bootstrapping from more complex underlying class of circuits [GGHRSW, A, CLTV]
  - no candidate / requires generic multilinear maps
Recent Models of Hash Functions \([Z, CCS, CCGCS]\)

**PrOM.** Enables applications by modelling use of code.
Recent Models of Hash Functions \([Z, CCS, CCGCS]\)

**Augmented ROM.** Transforms secure in AROM avoid known proofs of uninstantiability. 

*Gets stronger proofs by capturing contrivance.*

**PrOM.** Enables *applications* by *modelling use of code.*
Recent Models of Hash Functions $[Z, \text{CCS, CCGCS}]$

**Augmented ROM.**
Transforms secure in AROM
avoid known proofs of uninstantiability.

\textbf{Gets stronger proofs by capturing contrivance.}

**Low-Degree ROM.**
SNARK in $\mathcal{O}$-model for NP$^\mathcal{O}$.
Candidate instantiation is obfuscating algebraic PRF.

\textbf{Enables applications by adding arithmetic structure.}

\textbf{PrOM.}
Enables applications by modelling use of code.
Recent Models of Hash Functions \([Z, \text{ CCS, CCGCS}]\)

**Augmented ROM.** Transforms secure in AROM avoid known proofs of uninstantiability. **Gets stronger proofs by capturing contrivance.**

**Low-Degree ROM.** SNARK in \(\mathcal{O}\)-model for \(NP^0\). Candidate instantiation is obfuscating algebraic PRF. **Enables applications by adding arithmetic structure.**

**Arithmetized ROM.** PCD in \(\mathcal{O}\)-model for computation in \(\mathcal{O}\)-model. Candidate instantiation is hash functions. **Enables applications by modelling very specific use of code and adding arithmetic structure.** ([^ SAT reduction])

**Pr\(\mathcal{O}\)M.** Enables applications by modelling use of code.
Relation with Best-Possible Obfuscation

**Best-Possible.** \( \mathcal{A}(i\mathcal{O}(C_0)) \approx \mathcal{S}(C_1) \).

**Folklore.** 
\( i\mathcal{O}(\text{padded}(C)) \approx i\mathcal{O}(\text{Obf}(C)) \).
LHS hides whatever RHS hides.
\( i\mathcal{O} \) not less secure than any Obf.
Relation with Best-Possible Obfuscation \([\text{BR}]\)

**Best-Possible.** \(\mathcal{A}(iO(C_0)) \approx S(C_1)\).

**Folklore.** \(iO(\text{padded}(C)) \approx iO(\text{Obf}(C))\).
LHS hides whatever RHS hides.
\(iO\) **not less secure** than any Obf.

---

**Our result justifies this precondition!**

\[\exists \text{ Obf hiding } \mathcal{A}(C) \implies iO \text{ hides } \mathcal{A}(C)\]
Relation with Best-Possible Obfuscation

Best-Possible. \( A(i\mathcal{O}(C_0)) \approx \mathcal{S}(C_1) \).  

Folklore. \( i\mathcal{O}(\text{padded}(C)) \approx i\mathcal{O}(\text{Obf}(C)) \).  
LHS hides whatever RHS hides.  
\( i\mathcal{O} \) not less secure than any Obf.

Our result justifies this precondition!  
\( \exists \text{Obf hiding } A(C) \implies i\mathcal{O} \text{ hides } A(C) \)

Q. Use \( i\mathcal{O} \) in place of ideal obfuscation in applications? (A. Not clear.)

\( \Pi^{\text{Obf}} \text{ secure } \implies \Pi^{i\mathcal{O}(\text{Obf}(\cdot))} \text{ secure } \implies \Pi^{i\mathcal{O}(\text{padded}(\cdot))} \text{ secure } \)