Random Oracles and Obfuscation

Christina Brzuska[1] **Pooya Farshim**[2] Arno Mittelbach[3]

[1]Microsoft Research Cambridge
[2]**Queen's University Belfast**
[3]Technische Universität Darmstadt

University of Minho
4 November 16

Random Oracles

- Random oracles (ROs) model ideal hash functions [BR93].
  In the RO model:

  All parties have oracle access to a
  uniformly chosen **random function.**

- ROs enable the security proofs of a wide range of practical and strongly secure cryptosystems: encryption & signature schemes, key exchange, disk encryption, ...

Random Oracles

- Random oracles (ROs) model ideal hash functions [BR93].
  In the RO model:

  All parties have oracle access to a
  uniformly chosen **random function.**

- ROs enable the security proofs of a wide range of practical and strongly secure cryptosystems: encryption & signature schemes, key exchange, disk encryption, ...
- Standard-model counterparts are often less secure and/or less efficient.

RO Uninstantiability

- Reliance on ROs, although practical, is somewhat debatable:

  There are **uninstantiable** ROM schemes [CGH98].

RO Uninstantiability

- Reliance on ROs, although practical, is somewhat debatable:

  There are **uninstantiable** ROM schemes [CGH98].

  This means $\exists$ scheme Enc^{O} s.t.

  1. $\mathsf{Enc}^{\mathcal{RO}}$ is secure.
  2. $\mathsf{Enc}^{\mathsf{Hash}}$ is insecure for **any** concrete Hash.

RO Uninstantiability

- Reliance on ROs, although practical, is somewhat debatable:

  There are **uninstantiable** ROM schemes [CGH98].

This means $\exists$ scheme Enc^{O} s.t.

1. $\mathsf{Enc}^{\mathcal{RO}}$ is secure.
2. $\mathsf{Enc}^{\mathsf{Hash}}$ is insecure for **any** concrete Hash.

Scheme $\mathsf{Enc}^{O}(K, M)$:

1. Interpret M as the **description** of a hash function Hash.
2. If $O(x) = \mathsf{Hash}(x)$ for $x = 1 \ldots n$ append K to ciphertexts.
3. Else return a normal/good encryption of M.

RO Uninstantiability

- Reliance on ROs, although practical, is somewhat debatable:

  There are **uninstantiable** ROM schemes [CGH98].

  This means $\exists$ scheme Enc^{O} s.t.

  1. $\mathrm{Enc}^{\mathcal{RO}}$ is secure.
  2. $\mathrm{Enc}^{\mathrm{Hash}}$ is insecure for **any** concrete Hash.

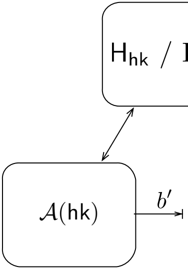
  Scheme $\mathrm{Enc}^{O}(K, M)$:

  1. Interpret M as the **description** of a hash function Hash.
  2. If $O(x) = \mathrm{Hash}(x)$ for $x = 1 \ldots n$ append K to ciphertexts.
  3. Else return a normal/good encryption of M.

- Lack of a **definition** formalizing "RO-like" behavior.

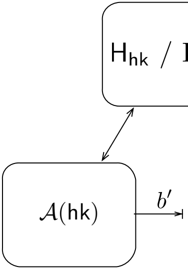
(Very) Naïve Attempt at Modeling ROs

Call a hash function "IND-RO" if:



(Very) Naïve Attempt at Modeling ROs

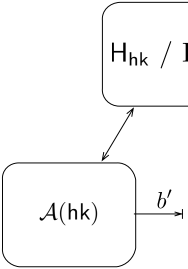
Call a hash function "IND-RO" if:



$$\mathbf{Adv}^{\text{ind-ro}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

(Very) Naïve Attempt at Modeling ROs

Call a hash function "IND-RO" if:

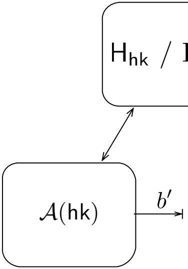


$$\mathbf{Adv}^{\text{ind-ro}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Clearly uninstantiable:

(Very) Naïve Attempt at Modeling ROs

Call a hash function "IND-RO" if:



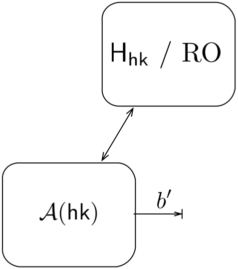
$$\mathbf{Adv}^{\mathrm{ind\text{-}ro}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Clearly uninstantiable:

$\mathcal{A}(\mathsf{hk})$: compute $\mathsf{H}_{\mathsf{hk}}(0)$ and compare to the oracle's reply.

(Very) Naïve Attempt at Modeling ROs

Call a hash function "IND-RO" if:



$$\mathbf{Adv}^{\text{ind-ro}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Clearly uninstantiable:

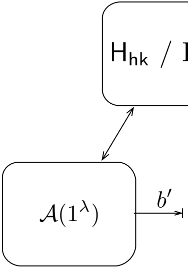
$\mathcal{A}(\mathsf{hk})$: compute $\mathsf{H}_{\mathsf{hk}}(0)$ and compare to the oracle's reply.

But observe:

The adversary knows a full input, namely $(\mathsf{hk}, 0)$.

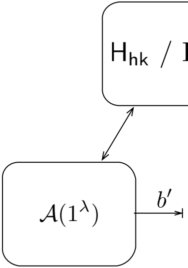
Can We Fix the Naïve Model?

Let's hide hk. We get PRF security:



Can We Fix the Naïve Model?

Let's hide hk. We get PRF security:

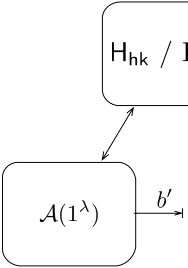


$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Not so useful in the context of hashing: hk is publicly available.

Can We Fix the Naïve Model?

Let's hide hk. We get PRF security:



$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Not so useful in the context of hashing: hk is publicly available.

First idea:

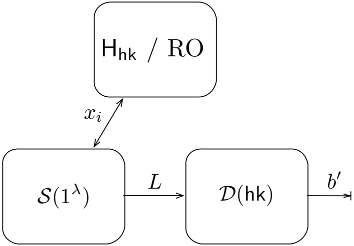
Split $\mathcal{A}$: one part gets hk and the other gets oracle access.

Modeling ROs via Split Adversaries

Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:

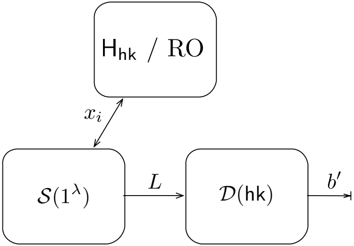
Modeling ROs via Split Adversaries

Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



Modeling ROs via Split Adversaries

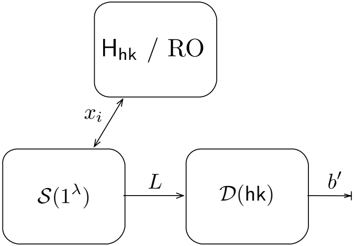
Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Modeling ROs via Split Adversaries

Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:

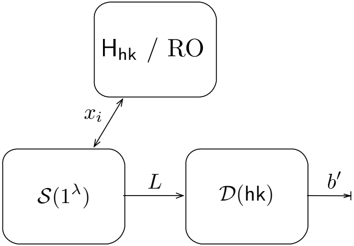


$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Still uninstantiable:

Modeling ROs via Split Adversaries

Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



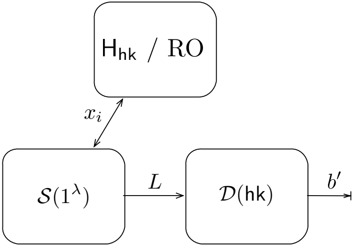
$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Still uninstantiable:

$\mathcal{S}$ leaks oracle's response on 0 via L, and $\mathcal{D}$(hk) checks where it's coming from.

Modeling ROs via Split Adversaries

Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

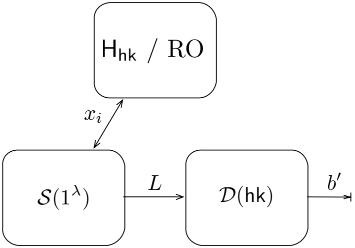
Still uninstantiable:

$\mathcal{S}$ leaks oracle's response on 0 via L, and $\mathcal{D}(\mathsf{hk})$ checks where it's coming from.

Second idea:

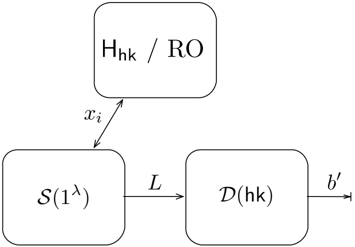
Restrict L: it must not leak any of $\mathcal{S}$'s queries.

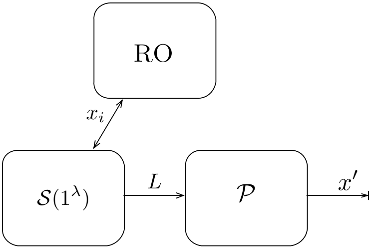
Universal Computational Extractors (UCEs) [BHK13a]



$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b = b'\right] - 1$$

Universal Computational Extractors (UCEs) [BHK13a]

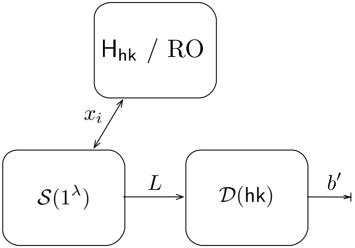


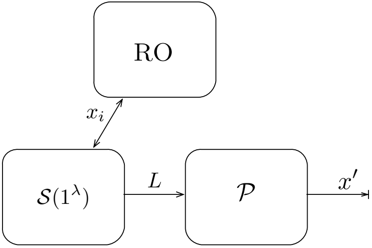


$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2\cdot\Pr\left[b = b'\right] - 1$$

$$\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \Pr\left[x' \in \{x_1, \ldots, x_n\}\right]$$

Universal Computational Extractors (UCEs) [BHK13a]



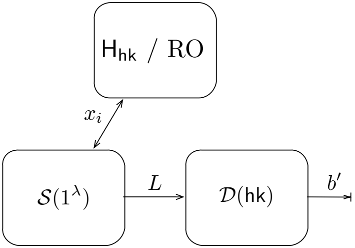


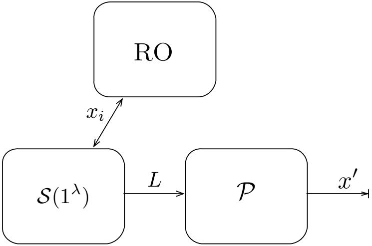
$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2\cdot\Pr\left[b = b'\right] - 1$

$\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \Pr\left[x' \in \{x_1, \ldots, x_n\}\right]$

$\mathcal{S}$ is **unpredictable** iff: $\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\cdot) \in \textsc{Negl}$ for any **efficient** $\mathcal{P}$.

Universal Computational Extractors (UCEs) [BHK13a]





$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2\cdot\Pr\left[b = b'\right] - 1$ | $\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \Pr\left[x' \in \{x_1, \ldots, x_n\}\right]$

$\mathcal{S}$ is **unpredictable** iff: $\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\cdot) \in \textsc{Negl}$ for any **efficient** $\mathcal{P}$.

H is **UCE1 secure** iff: $\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\cdot) \in \textsc{Negl}$ for any **unpredictable** $\mathcal{S}$.

Applications of UCE [BHK13a]

UCE-secure hash functions can instantiate the RO in

- Deterministic PKEs
- RKA and KDM security
- Point-function obfuscation
- Message-locked encryption
- Proofs of storage
- Poly-many hard-core bits for any OWF
- OAEP, garbling schemes, . . .

Applications of UCE [BHK13a]

UCE-secure hash functions can instantiate the RO in

- Deterministic PKEs
- RKA and KDM security
- Point-function obfuscation
- Message-locked encryption
- Proofs of storage
- Poly-many hard-core bits for any OWF
- OAEP, garbling schemes, . . .

UCEs model many RO-like properties.

Is UCE Security Instantiable?

Suppose we could **strongly obfuscate** H:

- $\mathcal{S}$: Choose a random x. Query x to get y. Leak as L the values

$$y, \quad \mathrm{Obf}(\mathrm{H}(\cdot, x)) \ .$$

- $\mathcal{D}$: Run the obfuscated code on hk. Check if the output matches y.

Is UCE Security Instantiable?

Suppose we could **strongly obfuscate** H:

- $\mathcal{S}$: Choose a random x. Query x to get y. Leak as L the values

$$y, \quad \mathrm{Obf}(\mathrm{H}(\cdot, x)) \; .$$

- $\mathcal{D}$: Run the obfuscated code on hk. Check if the output matches y.

Recall we need unpredictability for a nontrivial attack:

$\mathrm{Obf}(\mathrm{H}(\cdot, x))$ **must hide** x for unpredictability.

Is UCE Security Instantiable?

Suppose we could **strongly obfuscate** H:

- $\mathcal{S}$: Choose a random x. Query x to get y. Leak as L the values

$$y, \quad \mathsf{Obf}(\mathsf{H}(\cdot, x)) \ .$$

- $\mathcal{D}$: Run the obfuscated code on hk. Check if the output matches y.

Recall we need unpredictability for a nontrivial attack:

Obf(H($\cdot$, x)) **must hide** x for unpredictability.

It's unclear if such obfuscators exist.

Is UCE Security Instantiable?

Suppose we could **strongly obfuscate** H:

- $\mathcal{S}$: Choose a random x. Query x to get y. Leak as L the values

$$y, \quad \mathrm{Obf}(\mathrm{H}(\cdot, x)) \ .$$

- $\mathcal{D}$: Run the obfuscated code on hk. Check if the output matches y.

Recall we need unpredictability for a nontrivial attack:

$\mathrm{Obf}(\mathrm{H}(\cdot, x))$ **must hide** x for unpredictability.

It's unclear if such obfuscators exist.

Perhaps a different circuit and/or obfuscator might help?

Indistinguishability Obfuscation (iO)

Let C_0 and C_1 be two **functionally equivalent** circuits:

$$\forall x : C_0(x) = C_1(x) \ .$$

iO **security**: cannot efficiently distinguish obfuscations of such circuits:

$$\mathrm{iO}(C_0) \ \stackrel{c}{\approx} \ \mathrm{iO}(C_1) \ .$$

[GGH$^+$13]: indistinguishability obfuscation for all poly-sized circuits from intractability assumptions related to multi-linear maps.

Indistinguishability Obfuscation (iO)

Let C_0 and C_1 be two **functionally equivalent** circuits:

$$\forall x : C_0(x) = C_1(x) \; .$$

iO **security**: cannot efficiently distinguish obfuscations of such circuits:

$$\mathrm{iO}(C_0) \overset{c}{\approx} \mathrm{iO}(C_1) \; .$$

[GGH$^+$13]: indistinguishability obfuscation for all poly-sized circuits from intractability assumptions related to multi-linear maps.

Can we use iO to attack UCEs?

The iO Attack

$\mathcal{S}$: Leak an indistinguishably obfuscation of the Boolean circuit

$$\boxed{\mathsf{H}(\cdot, x) \stackrel{?}{=} y}$$

where $y :=$ oracle's response on x.

The iO Attack

$\mathcal{S}$: Leak an indistinguishably obfuscation of the Boolean circuit

$$\boxed{\mathsf{H}(\cdot, x) \stackrel{?}{=} y}$$

where $y :=$ oracle's response on x.

$\mathcal{D}$: Run the obfuscation on hk and return the result.

The iO Attack

$\mathcal{S}$: Leak an indistinguishably obfuscation of the Boolean circuit

$$\boxed{\mathsf{H}(\cdot, x) \stackrel{?}{=} y}$$

where $y :=$ oracle's response on x.

$\mathcal{D}$: Run the obfuscation on hk and return the result.

- Oracle $= \mathsf{H}$: $y = \mathsf{H}(\mathsf{hk}, x) \implies$ check always passes.

The iO Attack

$\mathcal{S}$: Leak an indistinguishably obfuscation of the Boolean circuit

$$\boxed{\mathsf{H}(\cdot, x) \stackrel{?}{=} y}$$

where $y :=$ oracle's response on x.

$\mathcal{D}$: Run the obfuscation on hk and return the result.

- Oracle $= \mathsf{H}$: $y = \mathsf{H}(\mathsf{hk}, x) \implies$ check always passes.
- Oracle $= \mathcal{RO}$: y is random $\implies$ check passes with prob $2^{-|y|}$.

Advantage of $(\mathcal{S}, \mathcal{D})$ is:

$$1 - 2^{-|y|} .$$

The iO Attack

$\mathcal{S}$: Leak an indistinguishably obfuscation of the Boolean circuit

$$\boxed{\mathsf{H}(\cdot, x) \stackrel{?}{=} y}$$

where $y :=$ oracle's response on x.

$\mathcal{D}$: Run the obfuscation on hk and return the result.

- Oracle $= \mathsf{H}$: $y = \mathsf{H}(\mathsf{hk}, x) \implies$ check always passes.
- Oracle $= \mathcal{RO}$: y is random $\implies$ check passes with prob $2^{-|y|}$.

Advantage of $(\mathcal{S}, \mathcal{D})$ is:

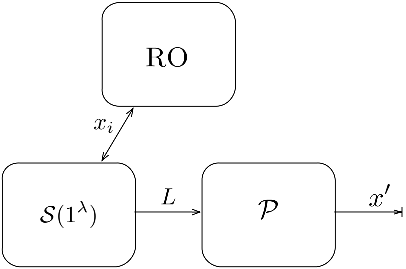
$$1 - 2^{-|y|} .$$

Is $\mathcal{S}$ unpredictable?

Proving Unpredictability

Need to ensure obfuscation hides *x* when *y* is **truly random**:

unpredictability was defined wrt the **random oracle.**



Proving Unpredictability

Need to ensure obfuscation hides x when y is **truly random**:

unpredictability was defined wrt the **random oracle.**

For any x, by the union bound we have

$$\Pr_{y \leftarrow_{\$} \mathsf{Rng}}\Big[\ \exists \mathsf{hk}\ \text{ s.t. }\ \mathsf{H}(\mathsf{hk}, x) = y\Big] \leq \frac{|\mathsf{KSp}|}{|\mathsf{Rng}|}\ .$$

Proving Unpredictability

Need to ensure obfuscation hides x when y is **truly random**:

unpredictability was defined wrt the **random oracle.**

For any x, by the union bound we have

$$\Pr_{y \leftarrow\$ \mathsf{Rng}}\left[\ \exists \mathsf{hk}\ \text{ s.t. }\ \mathsf{H}(\mathsf{hk}, x) = y \right] \leq \frac{|\mathsf{KSp}|}{|\mathsf{Rng}|}\ .$$

If $|\mathsf{KSp}| \ll |\mathsf{Rng}|$, we get that

$$\mathsf{H}(\cdot, x) \stackrel{?}{=} y \qquad \equiv \qquad \text{Constant zero circuit}$$

with overwhelming probability.

Proving Unpredictability

Need to ensure obfuscation hides x when y is **truly random**:

unpredictability was defined wrt the **random oracle.**

For any x, by the union bound we have

$$\Pr_{y \leftarrow\!\!\$\, \mathrm{Rng}}\Big[\ \exists \mathrm{hk}\ \text{ s.t. }\ \mathrm{H}(\mathrm{hk}, x) = y\Big] \leq \frac{|\mathrm{KSp}|}{|\mathrm{Rng}|}\ .$$

If $|\mathrm{KSp}| \ll |\mathrm{Rng}|$, we get that

$$\mathrm{H}(\cdot, x) \stackrel{?}{=} y \qquad \equiv \qquad \text{Constant zero circuit}$$

with overwhelming probability. Now by the security of iO

$$\mathrm{iO}(\mathrm{H}(\cdot, x) \stackrel{?}{=} y) \quad \text{leaks no more than} \quad \mathrm{iO}(\mathrm{Zero})\ ,$$

and the latter is independent of x.