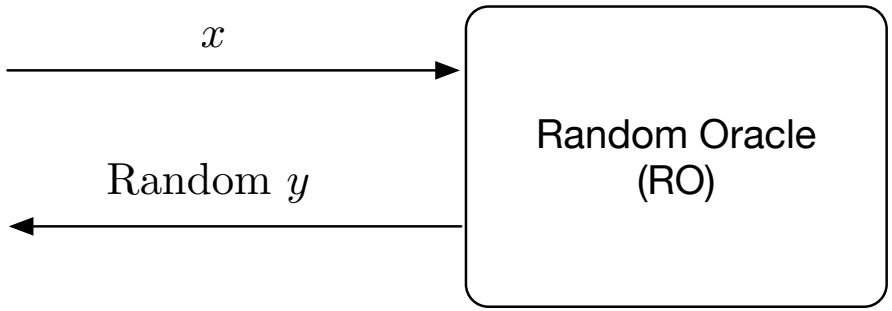
Random Oracles and Obfuscation

Pooya Farshim

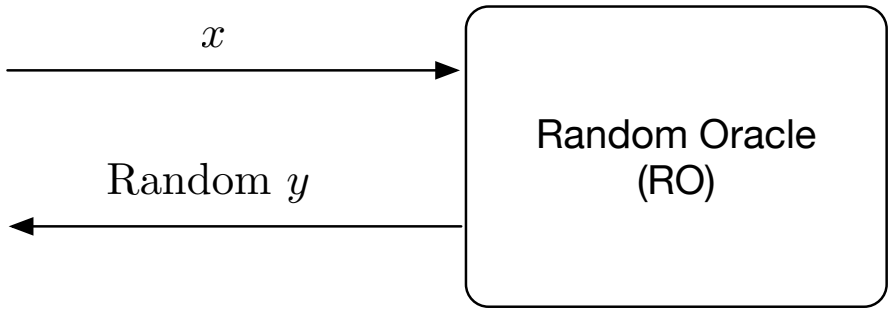
École Normale Supérieure

Summer School on Cryptology Crypto-CO

Random Oracles



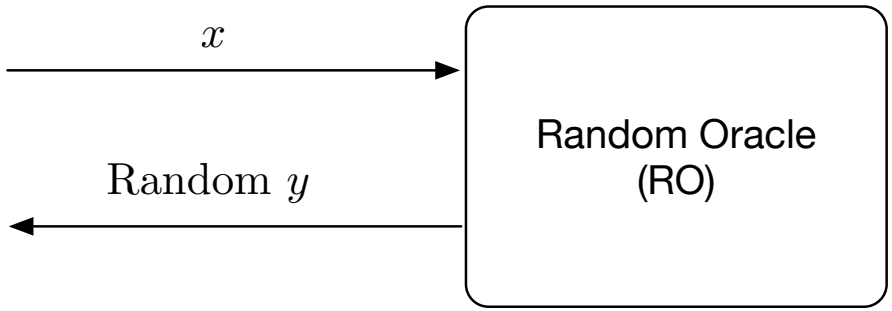
Random Oracles



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

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

Random Oracles

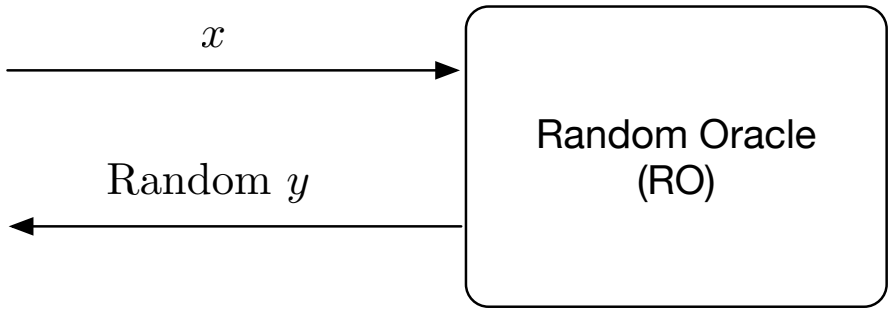


- Random oracles (ROs) model ideal hash functions [BR93].
  In the RO model:

  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 $\mathsf{Enc}^{\mathcal{O}}$ 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:

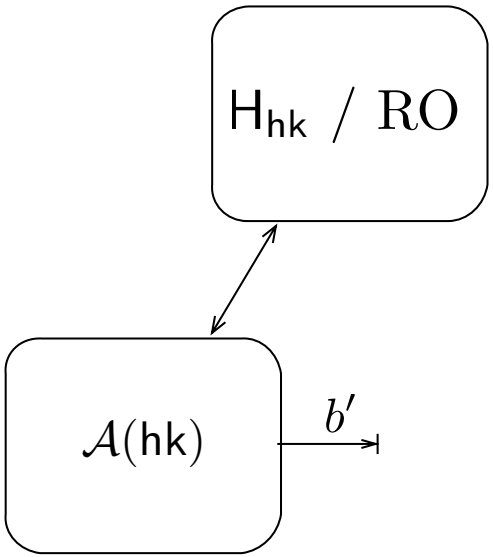
  There are **uninstantiable** ROM schemes [CGH98].

  This means $\exists$ scheme Enc^{O} s.t.
  1. $\mathsf{Enc}^{\mathcal{RO}}$ 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.
- Lack of a **definition** formalizing "RO-like" behavior.

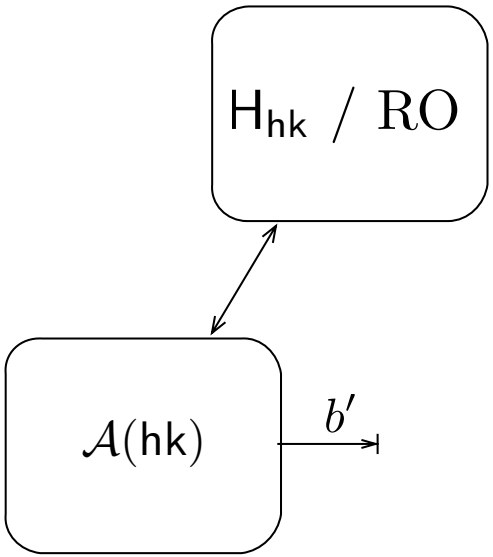
(Very) Naïve Attempt at Modeling ROs

Call a hash function "IND-RO" if:



(Very) Naïve Attempt at Modeling ROs

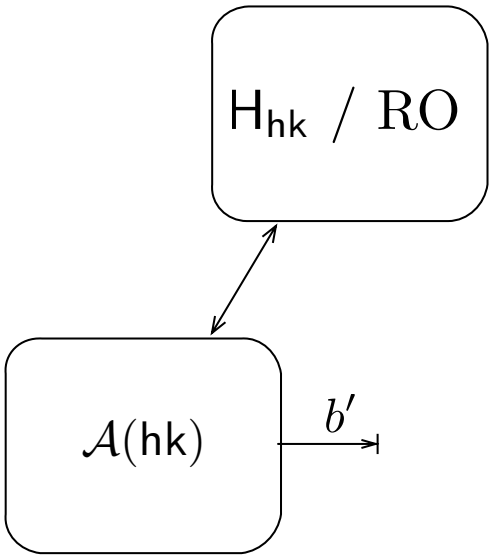
Call a hash function "IND-RO" if:



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

(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:

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

But observe:

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

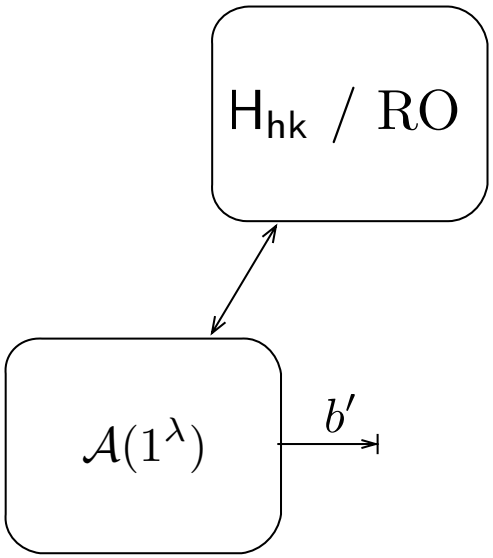
Can We Fix the Naïve Model?

Can We Fix the Naïve Model?

Let's hide hk:

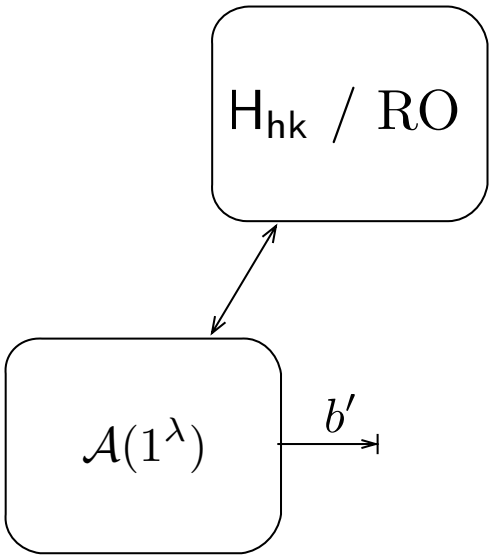
Can We Fix the Naïve Model?

Let's hide hk: PRF security:



Can We Fix the Naïve Model?

Let's hide hk: PRF security:

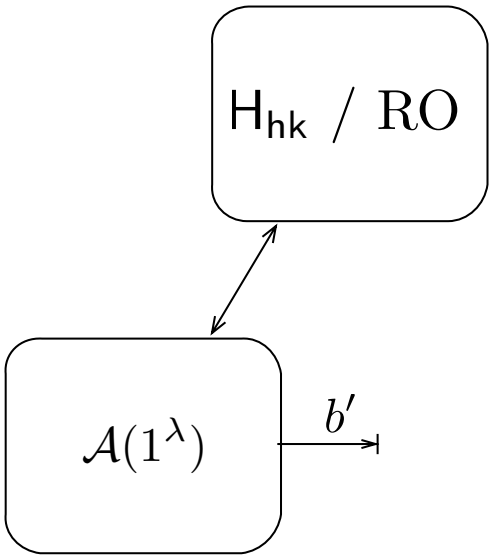


$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

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

Can We Fix the Naïve Model?

Let's hide hk: 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:

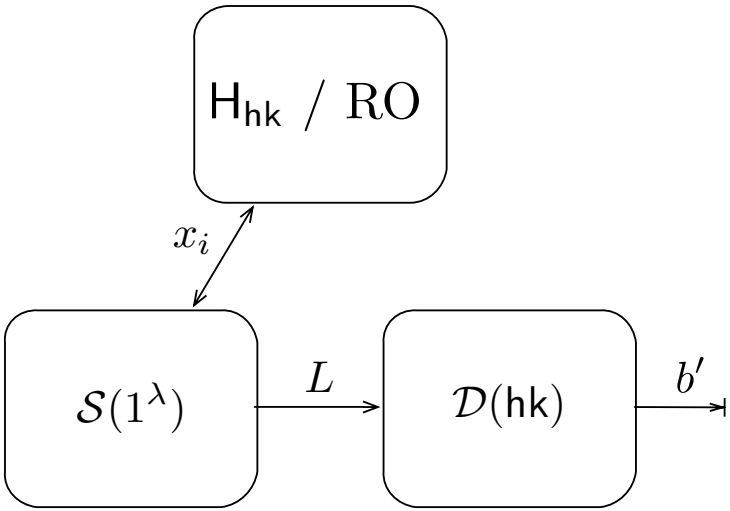
Split $\mathcal{A}$: one part gets hk and the other gets oracle access.

Modeling ROs via Split Adversaries

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

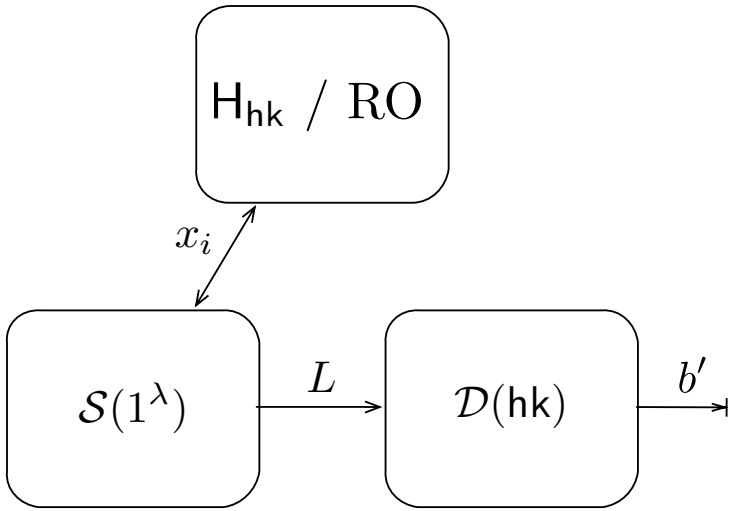
Modeling ROs via Split Adversaries

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



Modeling ROs via Split Adversaries

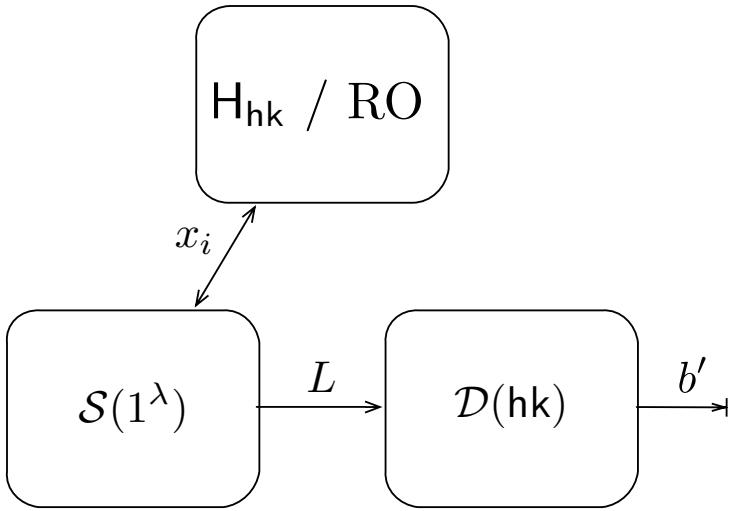
Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



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

Modeling ROs via Split Adversaries

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



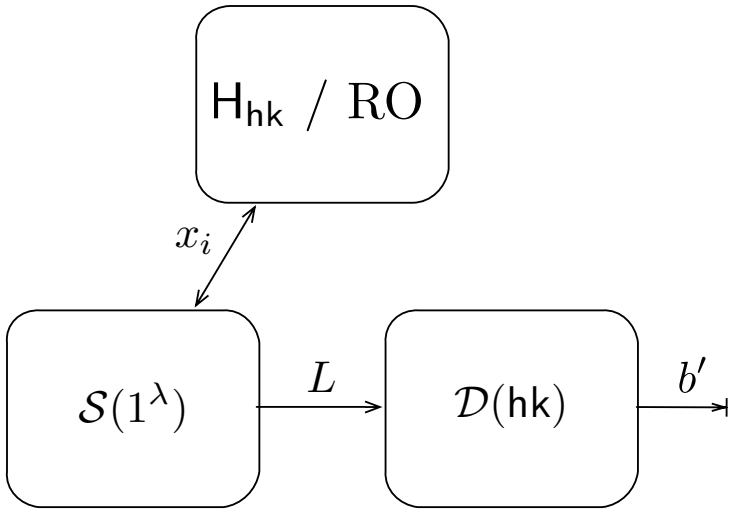
$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b' = b\right] - 1$$

Still uninstantiable:

$\mathcal{S}$ leaks oracle's response on 0 via L, and
$\mathcal{D}(\mathsf{hk})$ 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** $\mathcal{D}$:



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

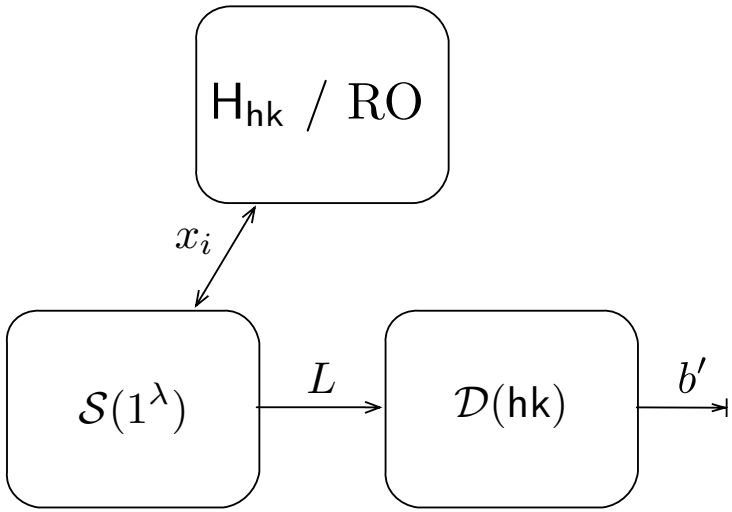
Still uninstantiable:

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

Second idea:

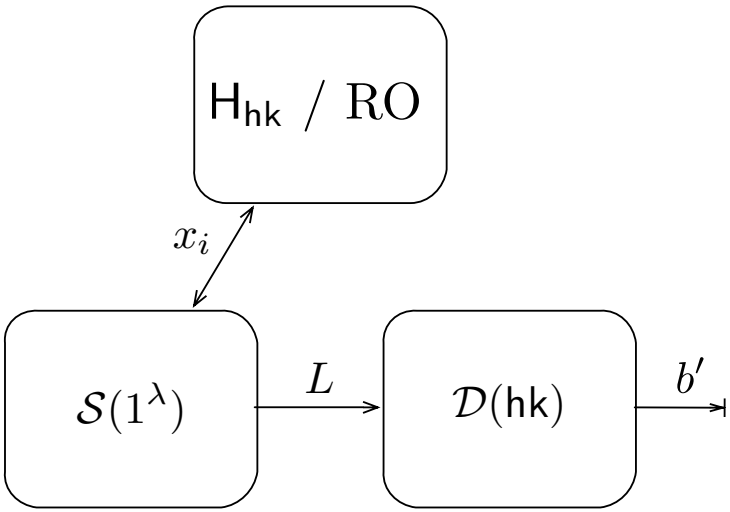
Restrict L: it must not leak any of $\mathcal{S}$'s queries.

Universal Computational Extractors (UCEs) [BHK13a]

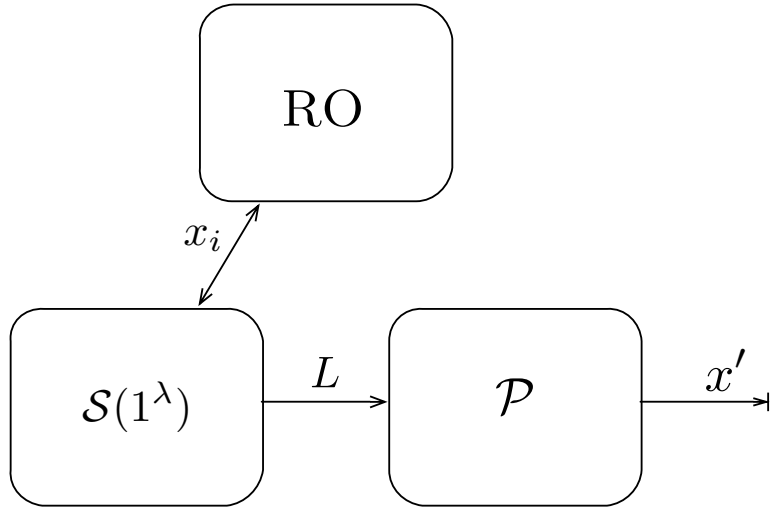


$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[b = b'\right] - 1$$

Universal Computational Extractors (UCEs) [BHK13a]

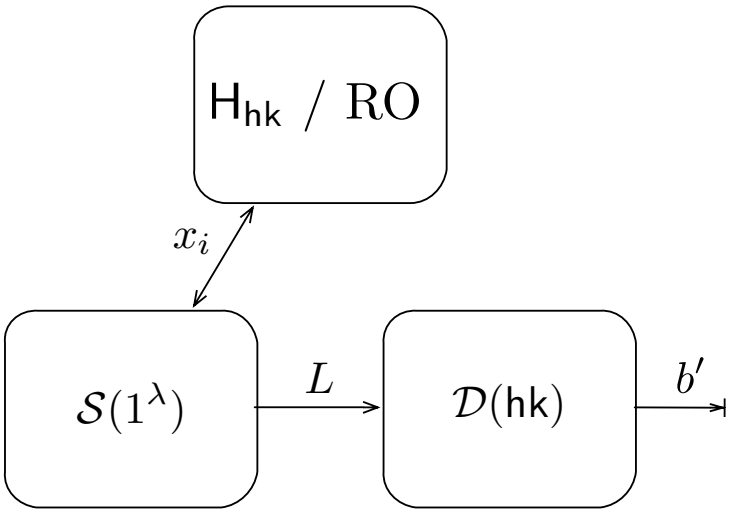


$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \mathsf{Pr}\left[b = b'\right] - 1$$

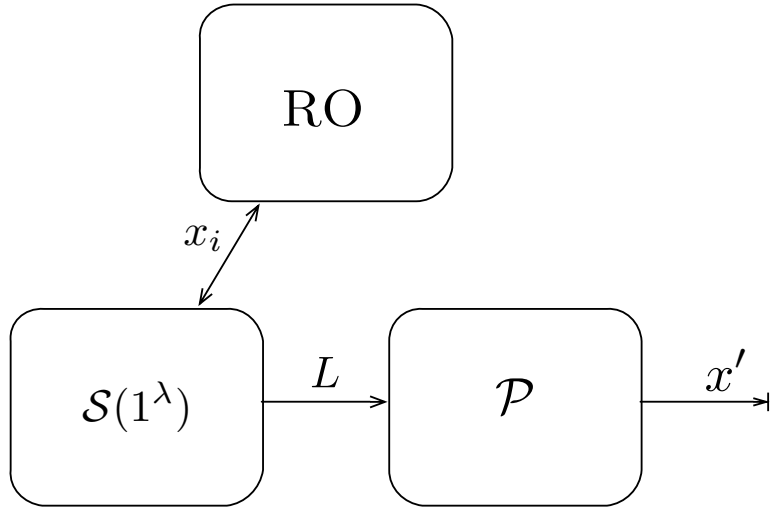


$$\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \mathsf{Pr}\left[x' \in \{x_1, \ldots, x_n\}\right]$$

Universal Computational Extractors (UCEs) [BHK13a]



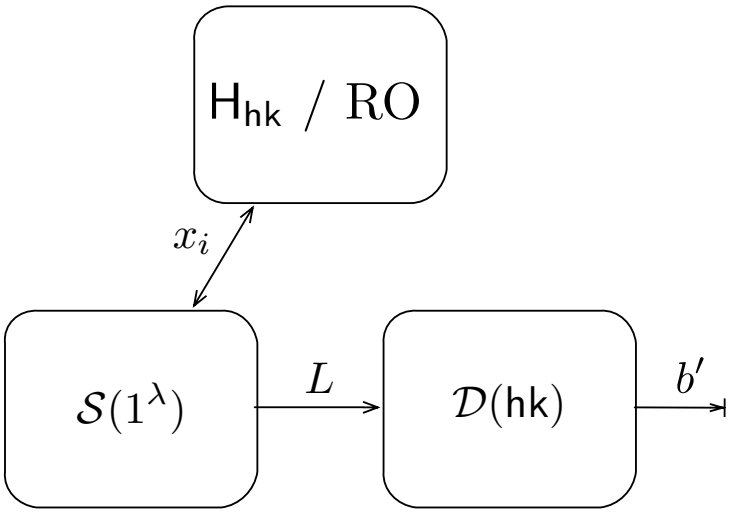
$$\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \mathsf{Pr}\left[b = b'\right] - 1$$



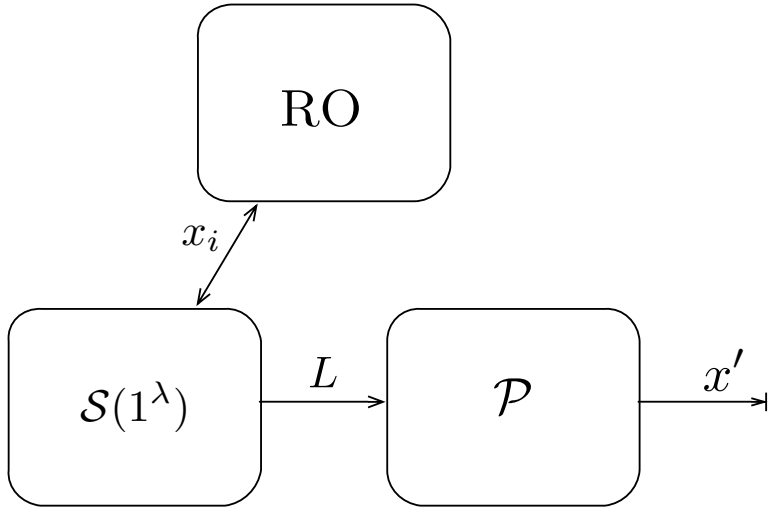
$$\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \mathsf{Pr}\left[x' \in \{x_1, \ldots, x_n\}\right]$$

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

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

Bottom line:

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

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

- $\mathcal{D}$: 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

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

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

Recall we need unpredictability for a nontrivial attack:

$$\mathsf{Obf}(\mathsf{H}(\cdot, x)) \textbf{ must hide } x \text{ for unpredictability.}$$

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:

$$\mathsf{Obf}(\mathsf{H}(\cdot, x)) \textbf{ must hide } x \text{ for unpredictability.}$$

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

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

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

Recall we need unpredictability for a nontrivial attack:

$$\mathsf{Obf}(\mathsf{H}(\cdot, x)) \textbf{ must hide } x \text{ for unpredictability.}$$

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:

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

[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:

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

[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 x is random and y is 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 x is random and y is 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 x is random and y is 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 x is random and y is 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:

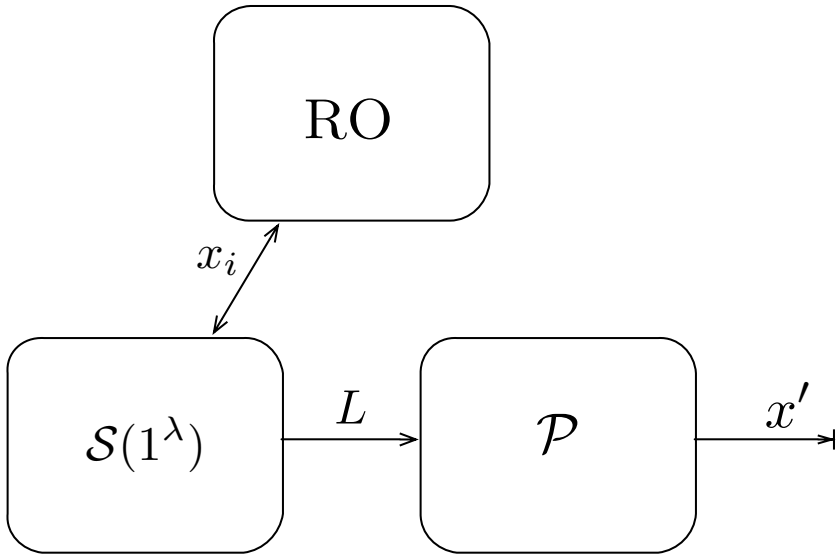
$$1 - 2^{-|y|} \ .$$

Is $\mathcal{S}$ unpredictable?

Proving Unpredictability

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

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



Proving Unpredictability

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

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

For any x, by the union bound we have

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

Proving Unpredictability

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

unpredictability was defined wrt. **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

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

with overwhelming probability.

Proving Unpredictability

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

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

For any x, by the union bound we have

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

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

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

with overwhelming probability. Now by the security of iO

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

and the latter is independent of x.

Dropping $|\mathsf{KSp}| \ll |\mathsf{Rng}|$

Consider instead iO-ing the circuit

$$\mathsf{H}(\cdot, x_1) \stackrel{?}{=} y_1 \ \wedge \ \cdots \ \wedge \ \mathsf{H}(\cdot, x_n) \stackrel{?}{=} y_n$$

where x_i are random and y_i are the corresponding responses.

Dropping $|\mathsf{KSp}| \ll |\mathsf{Rng}|$

Consider instead iO-ing the circuit

$$\mathsf{H}(\cdot, x_1) \stackrel{?}{=} y_1 \ \wedge \ \cdots \ \wedge \ \mathsf{H}(\cdot, x_n) \stackrel{?}{=} y_n$$

where x_i are random and y_i are the corresponding responses.

Theorem

Under the existence of iO, UCE security is uninstantiable.

Salvaging UCE: Statistical Unpredictability

iO is only **computationally** secure. Restrict the source class by

Allowing the **predictors** to run in **unbounded** time.

This raises the question: does **statistical iO** exist?

Salvaging UCE: Statistical Unpredictability

iO is only **computationally** secure. Restrict the source class by

Allowing the **predictors** to run in **unbounded** time.

This raises the question: does **statistical iO** exist?

Theorem (Goldwasser and Rothblum [GR07])

If statistical iO exists then the polynomial hierarchy collapses.

Salvaging UCE: Statistical Unpredictability

iO is only **computationally** secure. Restrict the source class by

Allowing the **predictors** to run in **unbounded** time.

This raises the question: does **statistical iO** exist?

Theorem (Goldwasser and Rothblum [GR07])

If statistical iO exists then the polynomial hierarchy collapses.

Remark for researchers working on negative results:

Example of an impossibility result that is
used positively to define a security model.

Statistical UCE and Applications

Definition

H is **statistically UCE** secure if UCE advantage is negligible for all sources that are unpredictable even wrt **unbounded** predictors.

Statistical UCE and Applications

Definition

H is **statistically UCE** secure if UCE advantage is negligible for all sources that are unpredictable even wrt **unbounded** predictors.

Applications of statistical UCEs:

- RKA and KDM security
- Proof-of-storage protocol
- point-function obfuscation, correlated-input secure hashing, and garbling schemes

Statistical UCE and Applications

Definition

H is **statistically UCE** secure if UCE advantage is negligible for all sources that are unpredictable even wrt **unbounded** predictors.

Applications of statistical UCEs:

- RKA and KDM security
- Proof-of-storage protocol
- point-function obfuscation, correlated-input secure hashing, and garbling schemes

Does **not** hold for some other applications:

Reductions to UCE rely on L that only **computationally** hide x.

Final Thoughts

- UCEs bring added provable-security guarantees to practical protocols.
- Reliance on idealized objects,relpaced by UCEs.
- This work brings refinements to UCEs:
  - Computational UCE notion is uninstantiable under iO.
  - Our weaker statistical UCE is robust and still useful.
- iO attack extends to many other ROM schemes:
  - D-PKE, Hedged PKE, Fujisaki–Okamoto, KDM, MLEs, ...

Final Thoughts

- UCEs bring added provable-security guarantees to practical protocols.
- Reliance on idealized objects,relpaced by UCEs.
- This work brings refinements to UCEs:
  - Computational UCE notion is uninstantiable under iO.
  - Our weaker statistical UCE is robust and still useful.
- iO attack extends to many other ROM schemes:
  - D-PKE, Hedged PKE, Fujisaki–Okamoto, KDM, MLEs, ...

Thank you.