

اوراکل‌های تصادفی و مبهم‌سازی کد

پویا فرشیم

(کار مشترک با کریستینا پروژکا و آرنو میتل‌باخ)

دانشگاه کوئینز بلفاست - اکول نرمال سوپریور

جنبه‌های ریاضی علوم کامپیوتری - مبانی رمز



Random Oracles and Obfuscation

Pooya Farshim

(Joint work with Christina Brzuska and Arno Mittelbach)

Queen's University Belfast/École Normale Supérieure

Mathematical Aspects of Computer Science – Foundations of Cryptography



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 $\text{Enc}^{\mathcal{O}}$ s.t.

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

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

- 1 Interpret M as the **description** of a hash function Hash.
- 2 If $O(x) = \text{Hash}(x)$ for $x = 1 \dots n$ 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.

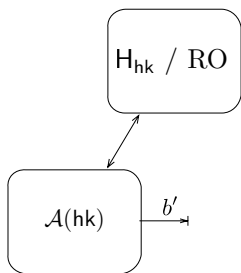
- 1 $\text{Enc}^{\mathcal{RO}}$ is secure.
- 2 Enc^{Hash} is insecure for **any** concrete Hash.

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

- 1 Interpret M as the **description** of a hash function Hash.
 - 2 If $O(x) = \text{Hash}(x)$ for $x = 1 \dots n$ append K to ciphertexts.
 - 3 Else return a normal/good encryption of M .
- Lack of a **definition** formalizing “RO-like” behavior.

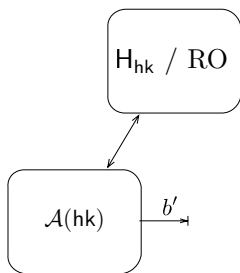
(Very) Naïve Attempt at Modeling ROs

Call a hash function “IND-RO” if:



(Very) Naïve Attempt at Modeling ROs

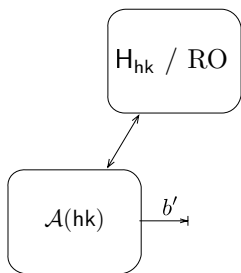
Call a hash function “IND-RO” if:



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

(Very) Naïve Attempt at Modeling ROs

Call a hash function “IND-RO” if:



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

Clearly uninstantiable:

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

But observe:

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

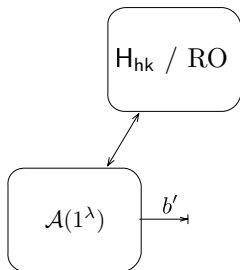
Can We Fix the Naïve Model?

Can We Fix the Naïve Model?

Let's hide hk :

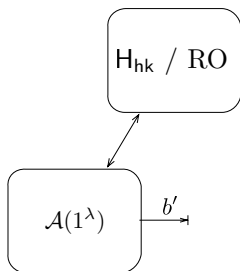
Can We Fix the Naïve Model?

Let's hide hk : PRF security:



Can We Fix the Naïve Model?

Let's hide hk : PRF security:

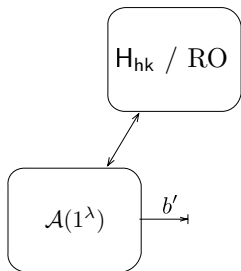


$$\mathbf{Adv}_{H, \mathcal{A}}^{\text{prf}}(\lambda) := 2 \cdot \Pr [b' = b] - 1$$

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

Can We Fix the Naïve Model?

Let's hide hk : PRF security:



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

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

First idea:

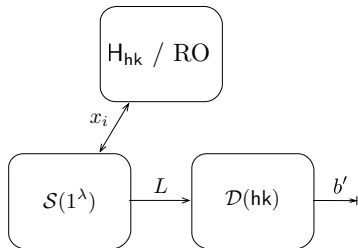
Split $\mathcal{A}$: one part gets hk and the other gets oracle access.

Modeling ROs via Split Adversaries

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

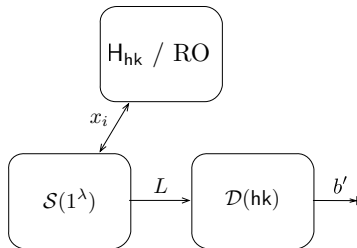
Modeling ROs via Split Adversaries

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



Modeling ROs via Split Adversaries

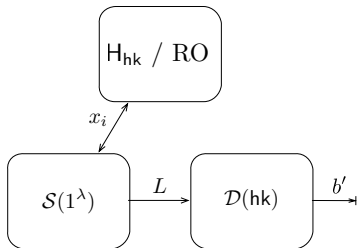
Call the two components of $\mathcal{A}$ the **source** $\mathcal{S}$ and the **distinguisher** $\mathcal{D}$:



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

Modeling ROs via Split Adversaries

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



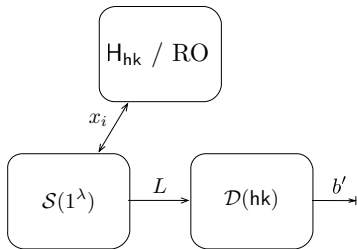
$$\mathbf{Adv}_{H, \mathcal{S}, \mathcal{D}}^{\text{uce}}(\lambda) := 2 \cdot \Pr[b' = b] - 1$$

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** $\mathcal{D}$:



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

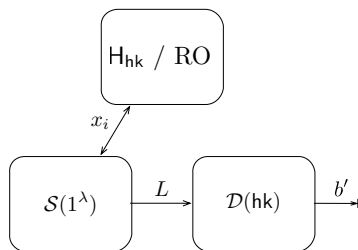
Still uninstantiable:

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

Second idea:

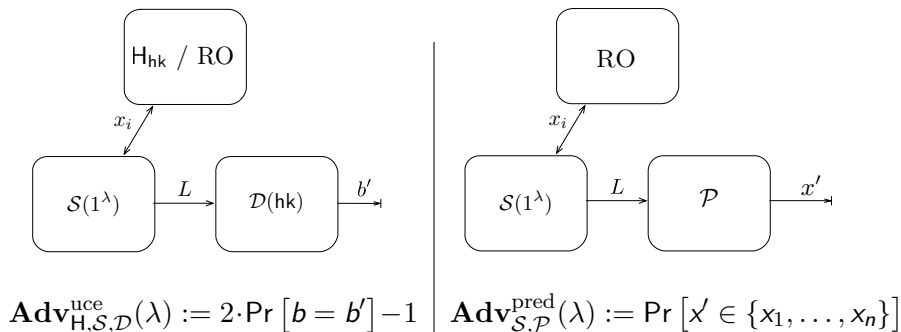
Restrict L : it must not leak any of $\mathcal{S}$'s queries.

Universal Computational Extractors (UCEs) [BHK13a]

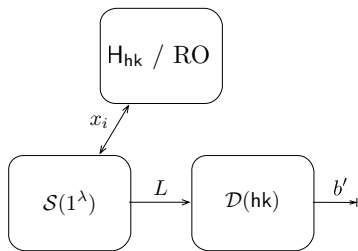


$$\mathbf{Adv}_{H, \mathcal{S}, \mathcal{D}}^{\text{uce}}(\lambda) := 2 \cdot \Pr[b = b'] - 1$$

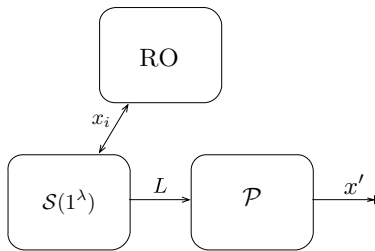
Universal Computational Extractors (UCEs) [BHK13a]



Universal Computational Extractors (UCEs) [BHK13a]



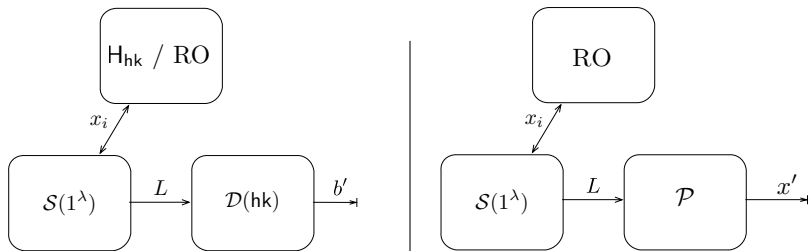
$$\text{Adv}_{\text{H}, \mathcal{S}, \mathcal{D}}^{\text{uce}}(\lambda) := 2 \cdot \Pr[b = b'] - 1$$



$$\text{Adv}_{\mathcal{S}, \mathcal{P}}^{\text{pred}}(\lambda) := \Pr[x' \in \{x_1, \dots, x_n\}]$$

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

Universal Computational Extractors (UCEs) [BHK13a]



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

$$\mathbf{Adv}_{S,P}^{\text{pred}}(\lambda) := \Pr[x' \in \{x_1, \dots, x_n\}]$$

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

H is **UCE1 secure** iff: $\mathbf{Adv}_{H,S,D}^{\text{uce}}(\lambda) \in \text{Negl}$ for any **unpredictable** 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

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

- $\mathcal{D}$: 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

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

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

Recall we need unpredictability for a nontrivial attack:

$\text{Obf}(H(\cdot, x))$ **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

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

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

Recall we need unpredictability for a nontrivial attack:

$\text{Obf}(H(\cdot, x))$ **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

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

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

Recall we need unpredictability for a nontrivial attack:

$\text{Obf}(H(\cdot, x))$ **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:

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

[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:

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

[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

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

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

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

where x is random and y is oracle's response on x .

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

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

The iO Attack

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

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

where x is random and y is oracle's response on x .

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

- Oracle = H : $y = H(hk, x) \implies$ 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

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

where x is random and y is oracle's response on x .

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

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

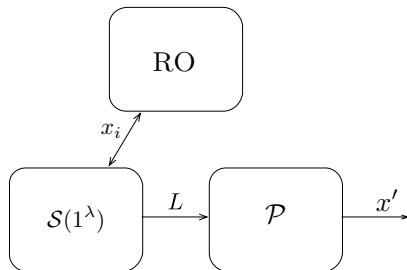
Advantage of $(\mathcal{S}, \mathcal{D})$ is:

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

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

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

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

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

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

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

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

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

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

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

with overwhelming probability. Now by the security of iO

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

and the latter is independent of x .

Dropping $|KSp| \ll |Rng|$

Consider instead iO-ing the circuit

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

where x_i are random and y_i are the corresponding responses.¹

¹Alternative view: range extension followed by iO attack.

Dropping $|KSp| \ll |Rng|$

Consider instead iO-ing the circuit

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

where x_i are random and y_i are the corresponding responses.¹

Theorem

Under the existence of iO, UCE security is uninstantiable.

Remark: Does not contradict UCE instantiability in the RO model.

¹Alternative view: range extension followed by iO attack.

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

Independently suggested by [BHK13c].

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

Independently suggested by [BHK13c].

Does **not** hold for D-PKE, MLE, OAEP, and hard-core functions:

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

Deterministic PKEs

D-PKEs are similar to PKEs except that their Enc is deterministic.

Deterministic PKEs

D-PKEs are similar to PKEs except that their Enc is deterministic.

Security: IND wrt. **high-entropy and pk-independent** messages:

$$\text{DetEnc}(\text{pk}, m_0) \stackrel{c}{\approx} \text{DetEnc}(\text{pk}, m_1) ,$$

Deterministic PKEs

D-PKEs are similar to PKEs except that their Enc is deterministic.

Security: IND wrt. **high-entropy and pk-independent** messages:

$$\text{DetEnc}(\text{pk}, m_0) \stackrel{c}{\approx} \text{DetEnc}(\text{pk}, m_1) ,$$

where

$$(m_0, m_1) \leftarrow_{\$} \mathcal{M}(1^\lambda)$$

and both m_0 and m_1 are required to have high min-entropy.

Encrypt-with-Hash (EwH)

EwH converts a normal PKE to a D-PKE:

$$\text{DetEnc}(\text{pk}, m) := \text{Enc}(\text{pk}, m; \mathcal{RO}(\text{pk}|m))$$

Encrypt-with-Hash (EwH)

EwH converts a normal PKE to a D-PKE:

$$\text{DetEnc}(\text{pk}, m) := \text{Enc}(\text{pk}, m; \mathcal{RO}(\text{pk}|m))$$

Theorem (Bellare, Boldyreva and O'Neill [BBO07])

If PKE is CPA secure, $\text{EwH}^{\mathcal{RO}}[\text{PKE}]$ is a secure D-PKE in ROM.

Encrypt-with-Hash (EwH)

EwH converts a normal PKE to a D-PKE:

$$\text{DetEnc}(\text{pk}, m) := \text{Enc}(\text{pk}, m; \mathcal{RO}(\text{pk}|m))$$

Theorem (Bellare, Boldyreva and O'Neill [BBO07])

If PKE is CPA secure, $\text{EwH}^{\mathcal{RO}}[\text{PKE}]$ is a secure D-PKE in ROM.

The analogous UCE theorem is

Theorem (Bellare, Hoang and Keelveedhi [BHK13])

If PKE is CPA secure, $\text{EwH}^H[\text{PKE}]$ is a secure D-PKE if H is UCE secure.

Encrypt-with-Hash (EwH)

EwH converts a normal PKE to a D-PKE:

$$\text{DetEnc}(\text{pk}, m) := \text{Enc}(\text{pk}, m; \mathcal{RO}(\text{pk}|m))$$

Theorem (Bellare, Boldyreva and O'Neill [BBO07])

If PKE is CPA secure, $\text{EwH}^{\mathcal{RO}}[\text{PKE}]$ is a secure D-PKE in ROM.

The analogous UCE theorem is

Theorem (Bellare, Hoang and Keelveedhi [BHK13])

If PKE is CPA secure, $\text{EwH}^H[\text{PKE}]$ is a secure D-PKE if H is UCE secure.

Our attacks say: this theorem is vacuous. This raises the questions:

Encrypt-with-Hash (EwH)

EwH converts a normal PKE to a D-PKE:

$$\text{DetEnc}(\text{pk}, m) := \text{Enc}(\text{pk}, m; \mathcal{RO}(\text{pk}|m))$$

Theorem (Bellare, Boldyreva and O'Neill [BBO07])

If PKE is CPA secure, $\text{EwH}^{\mathcal{RO}}[\text{PKE}]$ is a secure D-PKE in ROM.

The analogous UCE theorem is

Theorem (Bellare, Hoang and Keelveedhi [BHK13])

If PKE is CPA secure, $\text{EwH}^H[\text{PKE}]$ is a secure D-PKE if H is UCE secure.

Our attacks say: this theorem is vacuous. This raises the questions:

Can EwH instantiated under some other UCE-type assumption?

Or is EwH uninstantiable to start with?

Uninstantiability

Steps to show the uninstantiability of EwH:

- Start with an CPA scheme PKE.
- **Tweak** this scheme to a new scheme PKE^* .
- Prove that PKE^* is still CPA secure.
- Show that using PKE^* in EwH results in an **insecure** D-PKE for **any** hash function used to instantiate the random oracle.

Uninstantiability

Steps to show the uninstantiability of EwH:

- Start with an CPA scheme PKE.
- **Tweak** this scheme to a new scheme PKE^* .
- Prove that PKE^* is still CPA secure.
- Show that using PKE^* in EwH results in an **insecure** D-PKE for **any** hash function used to instantiate the random oracle.

So what's the tweak?

Pushing the Attack down to EwH

Recall the circuit used to attack UCEs:

$P[x, y](\cdot) : \text{if } H(\cdot, x) = y \text{ return } 1 \text{ else return } 0 .$

Pushing the Attack down to EwH

Recall the circuit used to attack UCEs:

$P[x, y](\cdot) : \text{if } H(\cdot, x) = y \text{ return } 1 \text{ else return } 0 .$

In the D-PKE/EwH context $x = \text{pk} \| m$ and $y = r$:

$P[\text{pk}, m, r](\cdot) : \text{if } H(\cdot, \text{pk} \| m) = r \text{ return } 1 \text{ else return } 0 .$

Pushing the Attack down to EwH

Recall the circuit used to attack UCEs:

$$P[x, y](\cdot) : \text{if } H(\cdot, x) = y \text{ return } 1 \text{ else return } 0 .$$

In the D-PKE/EwH context $x = \text{pk} \| m$ and $y = r$:

$$P[\text{pk}, m, r](\cdot) : \text{if } H(\cdot, \text{pk} \| m) = r \text{ return } 1 \text{ else return } 0 .$$

Now let's look at a **universal** variant of this circuit:

$$P'[\text{pk}, m, r](H(\text{hk}, \cdot)) : \text{if } H(\text{hk}, \text{pk} \| m) = r \text{ return } \boxed{m} \text{ else return } 0 .$$

Pushing the Attack down to EwH

Recall the circuit used to attack UCEs:

$$P[x, y](\cdot) : \text{if } H(\cdot, x) = y \text{ return } 1 \text{ else return } 0 .$$

In the D-PKE/EwH context $x = \text{pk} \| m$ and $y = r$:

$$P[\text{pk}, m, r](\cdot) : \text{if } H(\cdot, \text{pk} \| m) = r \text{ return } 1 \text{ else return } 0 .$$

Now let's look at a **universal** variant of this circuit:

$$P'[\text{pk}, m, r](H(\text{hk}, \cdot)) : \text{if } H(\text{hk}, \text{pk} \| m) = r \text{ return } \boxed{m} \text{ else return } 0 .$$

The tweaked scheme PKE^* is as follows:

- Gen^* : identical to that of the base scheme PKE.
- Enc^* : compute c as in PKE, but also output iO of $P'[\text{pk}, m, r]$:
 Enc^* is playing the role of a **UCE source**.
- Dec^* : ignore the iO component and PKE-decrypt.

Does This Work?

Does This Work?

Almost:

- **Attack**: run the circuit on the description of the hash function used in instantiation to recover m in the clear.
- **CPA security of PKE^*** : try to argue as before that

$$\text{iO}(P') \stackrel{c}{\approx} \text{iO}(Z) .$$

Does This Work?

Almost:

- **Attack**: run the circuit on the description of the hash function used in instantiation to recover m in the clear.
- **CPA security of PKE^*** : try to argue as before that

$$\text{iO}(P') \stackrel{c}{\approx} \text{iO}(Z) .$$

Cannot apply the union-bound argument directly:

There are potentially many (possibly infinity) more hash functions than hash digests.

Tweaking the Circuit

Consider a PKE that uses a **sparse** set of coins:

Coins r are first mapped to $\text{PRG}(r)$.
They are then used within the scheme.

Adapt the circuit accordingly:

$$P[\text{pk}, m, \text{PRG}(r)] \left(H(\text{hk}, \cdot) \right) : \text{if } \text{PRG}(H(\text{hk}, \text{pk} \| m)) = \text{PRG}(r) \text{ return } m \\ \text{else return } 0$$

How Do We Argue Now?

How Do We Argue Now?

Two game hops:

How Do We Argue Now?

Two game hops:

- 1 **PRG hop:** The **description** of $P[\text{pk}, m, \text{PRG}(r)]$ is ind. from $P[\text{pk}, m, s]$ for a truly random s .

How Do We Argue Now?

Two game hops:

- 1 **PRG hop:** The **description** of $P[\text{pk}, m, \text{PRG}(r)]$ is ind. from $P[\text{pk}, m, s]$ for a truly random s .
- 2 **iO hop** : $P[\text{pk}, m, s]$ is functionally equivalent to Zero as s will be **outside the range** of PRG with overwhelming probability.

How Do We Argue Now?

Two game hops:

- 1 **PRG hop**: The **description** of $P[\text{pk}, m, \text{PRG}(r)]$ is ind. from $P[\text{pk}, m, s]$ for a truly random s .
- 2 **iO hop** : $P[\text{pk}, m, s]$ is functionally equivalent to Zero as s will be **outside the range** of PRG with overwhelming probability.

Structurally,

$$\begin{array}{ccccccc} P_{\text{PRG}(r)} & \overset{c}{\approx} & P_s & \equiv & Z \\ \\ \text{iO}(P_{\text{PRG}(r)}) & \overset{c}{\approx} & \text{iO}(P_s) & \overset{c}{\approx} & \text{iO}(Z) \end{array}$$

Syntactic Checks

P takes as input a hash-function description:

- Assuming iO for **circuits**, we get uninstantiability wrt **a priori bounded size** hash functions.

Syntactic Checks

P takes as input a hash-function description:

- Assuming iO for **circuits**, we get uninstantiability wrt **a priori bounded size** hash functions.
- Assuming iO for **Turing machines**, we get full uninstantiability wrt **any** hash function.

Syntactic Checks

P takes as input a hash-function description:

- Assuming iO for **circuits**, we get uninstantiability wrt **a priori bounded size** hash functions.
- Assuming iO for **Turing machines**, we get full uninstantiability wrt **any** hash function.

Assumptions:

- [GGHRSW14] constructs iO for circuits.

Syntactic Checks

P takes as input a hash-function description:

- Assuming iO for **circuits**, we get uninstantiability wrt **a priori bounded size** hash functions.
- Assuming iO for **Turing machines**, we get full uninstantiability wrt **any** hash function.

Assumptions:

- [GGHRSW14] constructs iO for circuits.
- iO for TMs can be based on (pubic-coin) **differing-inputs** obfuscation and a few other high-tech gadgets [IPS15,ABGSZ13,BCP13].

Flexibility of the Technique

- Generalizes to a wider class of “admissible” transforms.
- Applies to the classical **Fujisaki–Okamoto** transform [FO98] for converting CPA to CCA security.
- Can be adapted to ROM transforms for **KDM** security, **message-locked** encryption, and **hedged** PKEs.
- Schemes specifically designed to foil our simple iO attack fall prey to more complex “double-iO” attacks:

Run one obfuscated circuit on the description
of **another** obfuscated circuit.

Final Thoughts

- UCEs bring added provable-security guarantees to practical protocols.
- Progress 15 years after [CGH98].
- This work brings cryptanalytic validation & refinements to UCEs:
 - ▶ A natural computational UCE notion is uninstantiable.
 - ▶ Our weaker statistical UCE is robust and still useful.
- iO attack extends to many prominent ROM transforms:
 - ▶ D-PKE, Hedged PKE, Fujisaki–Okamoto, KDM, MLEs, ...

Final Thoughts

- UCEs bring added provable-security guarantees to practical protocols.
- Progress 15 years after [CGH98].
- This work brings cryptanalytic validation & refinements to UCEs:
 - ▶ A natural computational UCE notion is uninstantiable.
 - ▶ Our weaker statistical UCE is robust and still useful.
- iO attack extends to many prominent ROM transforms:
 - ▶ D-PKE, Hedged PKE, Fujisaki–Okamoto, KDM, MLEs, ...

Questions:

- Give transforms that do not suffer from iO attacks.
- Characterize the power of iO attacks more precisely.
- Extend modeling of ROs.

Final Thoughts

- UCEs bring added provable-security guarantees to practical protocols.
- Progress 15 years after [CGH98].
- This work brings cryptanalytic validation & refinements to UCEs:
 - ▶ A natural computational UCE notion is uninstantiable.
 - ▶ Our weaker statistical UCE is robust and still useful.
- iO attack extends to many prominent ROM transforms:
 - ▶ D-PKE, Hedged PKE, Fujisaki–Okamoto, KDM, MLEs, ...

Questions:

- Give transforms that do not suffer from iO attacks.
- Characterize the power of iO attacks more precisely.
- Extend modeling of ROs.

Thank you.