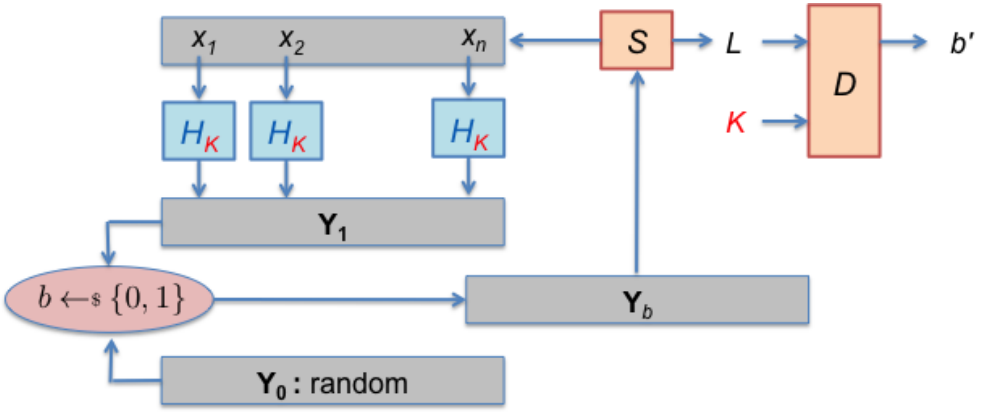
Instantiating Random Oracles via UCEs

Mihir Bellare

Joint work with Sriram Keelveedhi

UCSD



Disclaimer

The work reported on here is very much work in progress. The UCE definition has evolved since this talk and will evolve further, and what is here should not be taken as its definitive or final form. Theorems stated here have been strengthened, and we have other results as well. A paper is not yet available but we hope to have one soon.

Contributions in brief

UCE (Universal Computational Extractors): A new DEFINITION of security for hash functions.

Instantiating ROs: UCE hash functions can provably instantiate ROs in a variety of existing schemes: DE, HE, MLE, PKE (OAEP), KDM, RKA, …

Generalization: UCE extends and unifies existing definitions like hardcore functions, extractors, correlated-input functions, …

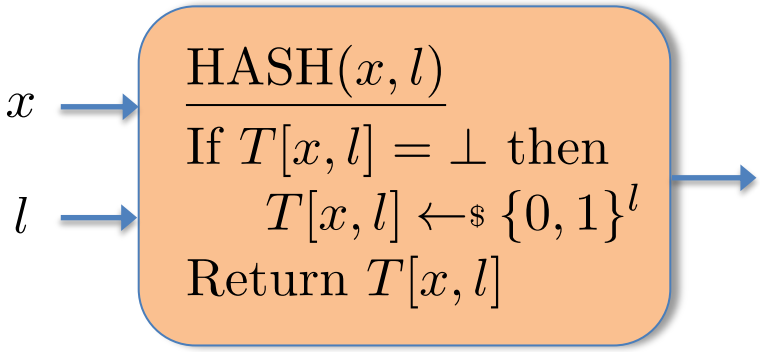
Modular design: Instantiate RO in (existing and) practical ROM schemes, not design new schemes!

Modular proofs: (Instantiable RO paradigm): Proofs of instantiated schemes use results on the ROM security of the scheme in a blackbox way.

Random Oracle Model (ROM) and Paradigm

[BR93]

Access to this procedure given to scheme algorithms as well as to the adversary.



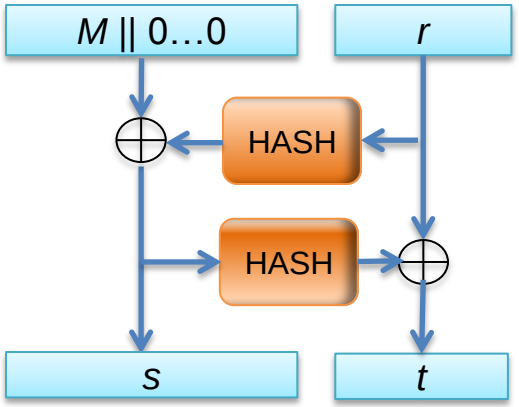
Step 1: Prove security of scheme in the ROM

Step 2: Instantiate HASH via a cryptographic hash function *H* in an implementation

The instantiated scheme is then secure as long as *H* behaves "like a random oracle."

RSA-OAEP [BR94]

$\underline{\mathcal{E}_{N,e}(M)} \;/\!/\; |M| = k_1$
$r \leftarrow\!\!\$\ \{0,1\}^{k_0}$
$s \leftarrow \mathrm{HASH}(r, k_2) \oplus M \| 0^{k_2 - k_1}$
$t \leftarrow \mathrm{HASH}(s, k_0) \oplus r$
Return $(s \| t)^e \bmod N$



Theorem: In the ROM, RSA-OAEP is

- IND-CPA secure if RSA is 1-way [BR94]
- IND-CCA secure if RSA is partial-domain 1-way [FOPS01]

In PKCS#1: Implemented with

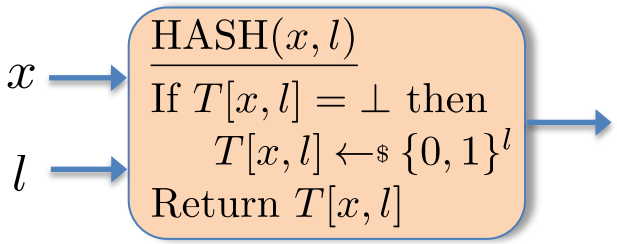
$$\mathrm{HASH}(x, l) = [\mathrm{SHA1}(x\|1)\|\mathrm{SHA1}(x\|2)\|\ldots]_{1..l}$$

ROM Features

Yields schemes that are
- Practical and
- Proven secure

ROM paradigm is
- Widely employed
- Works in practice

ROM Critiques



Step 1: Prove security of scheme in the ROM.

Step 2: Instantiate HASH via a cryptographic hash function *H* in an implementation.

The instantiated scheme is then secure as long as *H* behaves "like a random oracle."

The ROM proof does not apply to the instantiated scheme

It is not clear what this means.

Counter-examples

Theorem: [CGH98,MRH04] There exist encryption schemes that are

- Secure in the ROM
- Insecure under any instantiation of HASH

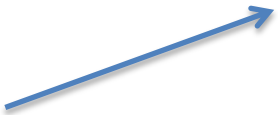
$\underline{\mathcal{E}_{pk}(M)}$

View M as executable code

$y \leftarrow M(x)$

If $y = \mathrm{HASH}(x, |y|)$ then do BUG!

Else encrypt M securely under pk



The scheme in which HASH is instantiated by *H* can be broken if a program implementing *H* is the message *M* encrypted.

Counter-examples that are (perhaps?) more realistic: Ni02, GT03, BBP04, CGH04, BDWY12, BCPT13, …

The debate continues …

The RO paradigm is theoretically unsound.

It can yield insecure instantiated schemes.

Your counter-examples are artificial.
The paradigm has not failed in practice.
You have some alternative?

It's too easy.

We developed all this nice, deep, complex theory, and you want to replace it with a noise box.

Google allintitle: "without random oracles"

Scholar About 263 results (**0.03** sec)

Google "without random oracles"

Scholar About 3,320 results (**0.05** sec)

RSA+ROM

q-ABDHE assumption

The core problem: Lack of a definition

Step 1: Prove security of scheme in the ROM.

Step 2: Instantiate HASH via a cryptographic hash function *H* in an implementation.

The instantiated scheme is then secure as long as *H* behaves "like a random oracle."

The ROM proof does not apply to the instantiated scheme

It is not clear what this means.

We lack a formal **DEFINITION** of what it means for a hash function to behave "like a random oracle."

The core problem: Lack of a definition

Step 1: Prove security of scheme in the ROM.

Step 2: Instantiate HASH via a cryptographic hash function *H* in an implementation.

The instantiated scheme is then secure as long as *H* behaves "like a random oracle."

The ROM proof does not apply

Cryptographers don't mind strong assumptions. But they like to know exactly what they are assuming.

We lack a formal **DEFINITION** of what it means for a hash function to behave "like a random oracle."

Want: Instantiable RO Paradigm

Step 1: Prove security of scheme in the ROM

Step 2: Instantiate HASH via a cryptographic hash function *H* in an implementation

Exploiting the result of Step 1 in a blackbox way!

Step 3: Prove that the instantiated scheme is secure as long as *H* meets definition X.

Game-based, falsifiable definition in the standard style. Tells us when an attack on *H* is successful.

We cannot hope to find (achievable) X where this works for ALL schemes. But we would like to cover interesting sub-classes of schemes.

Previous work

X = PRF: [GGM86]

Works for symmetric cryptography, where adversary does NOT have access to HASH oracle.

X = POWHF (Perfectly One-Way Hash Functions): [C97,CMR98]

X = Non-malleable hash functions: [BCFW09]

X = CIH (Correlated-input-secure hash functions): [GOR11]

Limited applicability.

Contributions in brief

UCE (Universal Computational Extractors): A new DEFINITION of security for hash functions.

Instantiating ROs: UCE hash functions can provably instantiate ROs in a variety of existing schemes: DE, HE, MLE, PKE (OAEP), KDM, RKA, …

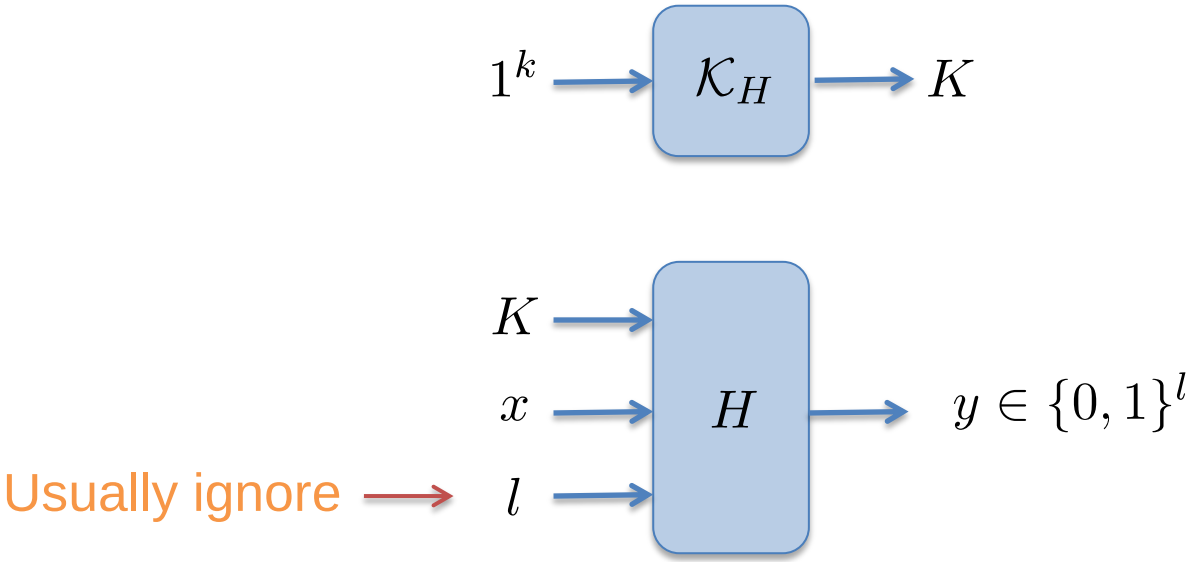
Generalization: UCE extends and unifies existing definitions like hardcore functions, extractors, correlated-input functions, …

Modular design: Instantiate RO in (existing and) practical ROM schemes, not design new schemes!

Modular proofs: (Instantiable RO paradigm): Proofs of instantiated schemes use results on the ROM security of the scheme in a blackbox way.

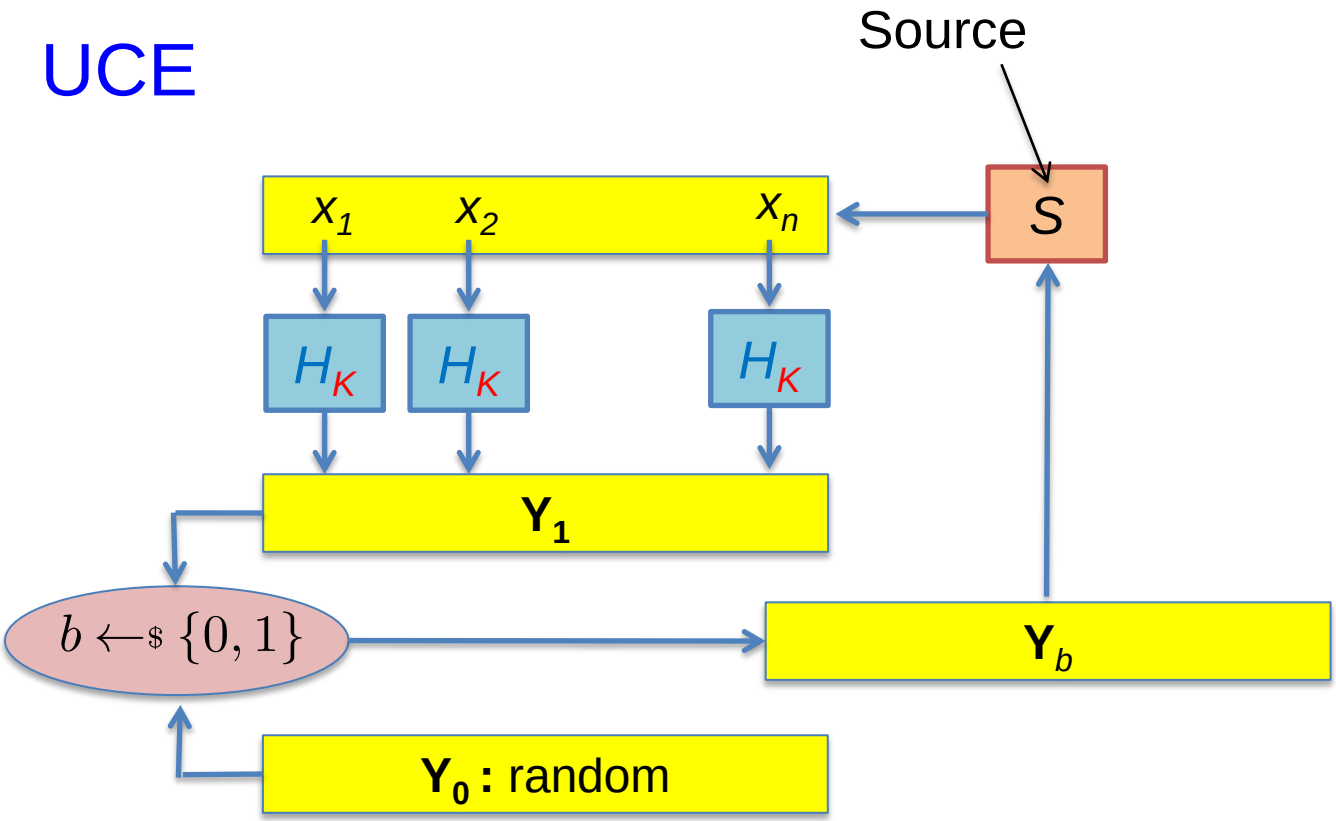
Syntax

A family of hash functions is a pair $(\mathcal{K}_H, H)$ of poly-time algorithms.



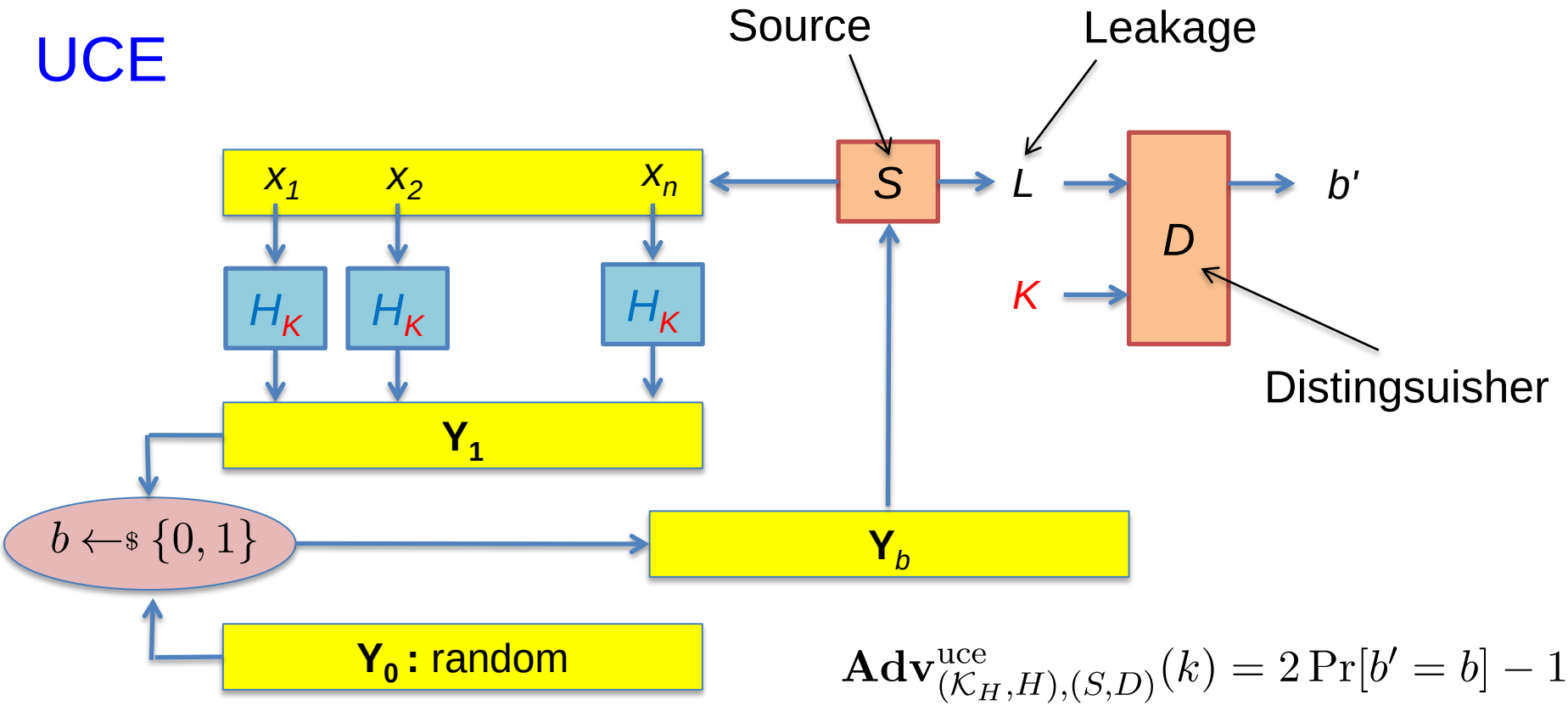
Think HMAC-SHA1, not SHA1.

UCE



Informally: $\mathbf{Y_1}$ looks random.

UCE

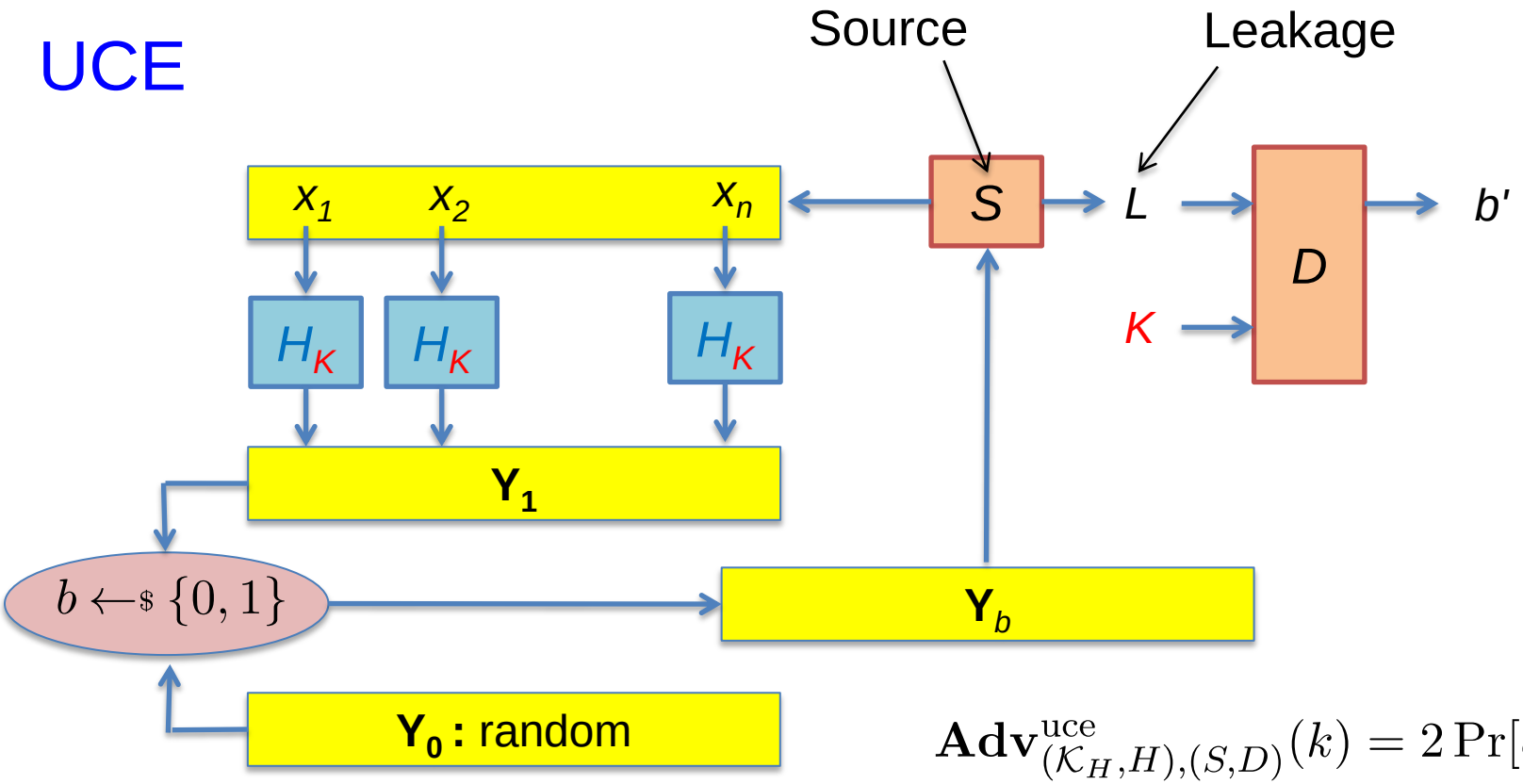


$$\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) = 2\Pr[b' = b] - 1$$

$(\mathcal{K}_H, H)$ is *UCE-secure* if $\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(\cdot)$ is negligible for every poly-time *S* and every poly-time *D*.

Informally: $\mathbf{Y_1}$ looks random given *L*.

UCE

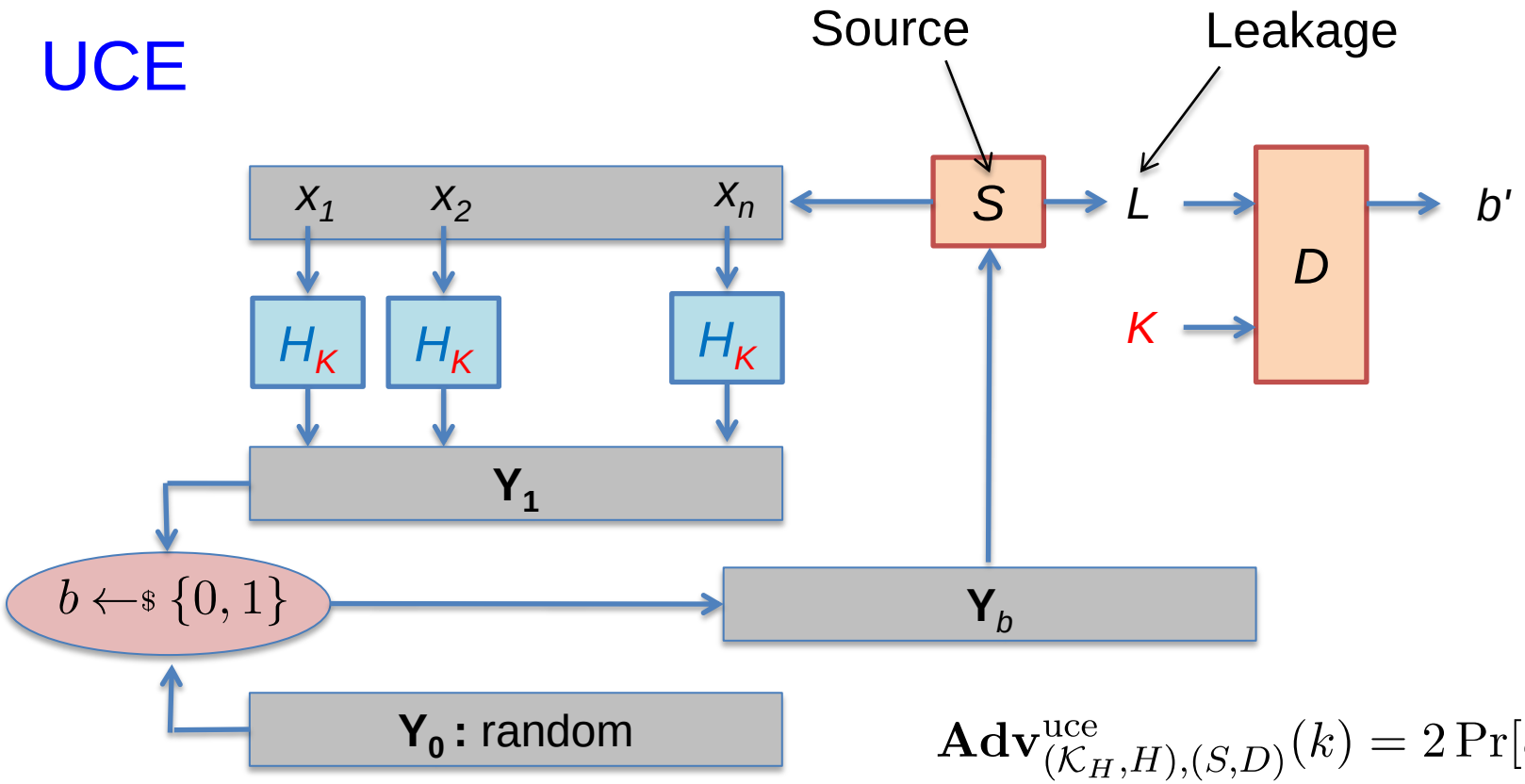


$$\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) = 2\Pr[b' = b] - 1$$

$(\mathcal{K}_H, H)$ is *UCE-secure* if $\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(\cdot)$ is negligible for every poly-time *S* and every poly-time *D*.

Not possible! *L* could contain x_1 and *D* could check whether $H_K(x_1)$ equals the first component of vector $\mathbf{Y}_b$.

UCE



$$\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) = 2\Pr[b' = b] - 1$$

$(\mathcal{K}_H, H)$ is *UCE-secure* if $\mathbf{Adv}^{\mathrm{uce}}_{(\mathcal{K}_H,H),(S,D)}(\cdot)$ is negligible for every poly-time unpredictable S and every poly-time D.

Computing any x_i given L is hard.

Informally: $\mathbf{Y_1}$ looks random given L as long as you cannot compute any x_i.

UCE: Formally

Game UCE

$\underline{\text{MAIN}()}$
$K \leftarrow\!\!\$\ \mathcal{K}_H(1^k)\,;\ b \leftarrow\!\!\$\ \{0,1\}$
$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^k)$
$b' \leftarrow\!\!\$\ D(1^k, K, L)$
Return $(b = b')$

$\underline{\text{HASH}(x, l)}$
If not $T[x, l]$ then
  If $b = 1$ then $T[x, l] \leftarrow H(K, x, l)$
  Else $T[x, l] \leftarrow\!\!\$\ \{0,1\}^l$
Return $T[x, l]$

$$\mathbf{Adv}^{\text{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) = 2\Pr[\text{UCE}^{(S,D)}_{(\mathcal{K}_H,H)}(k) \Rightarrow \text{true}] - 1$$

$(\mathcal{K}_H, H)$ is *UCE-secure* if $\mathbf{Adv}^{\text{uce}}_{(\mathcal{K}_H,H),(S,D)}(\cdot)$ is negligible for every poly-time unpredictable *S* and every poly-time *D*.

The unpredictability condition: Formally

Game Pred

$\underline{\text{MAIN}()}$

$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^k)$

done $\leftarrow$ true

$x \leftarrow\!\!\$\ U^{\text{HASH}}(1^k, L)$

Return $(x \in X)$

$\underline{\text{HASH}(x, l)}$

If not $T[x, l]$ then $T[x, l] \leftarrow\!\!\$\ \{0,1\}^l$

If not done then $X \leftarrow X \cup \{x\}$

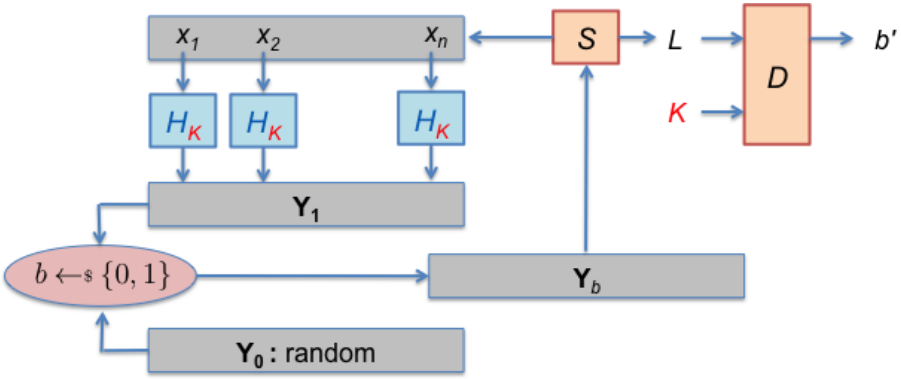
Return $T[x, l]$

$$\mathbf{Adv}^{\text{pred}}_{S,U}(k) = \Pr[\text{Pred}^{(S,U)}(k) \Rightarrow \text{true}]$$

S is *unpredictable* if $\mathbf{Adv}^{\text{pred}}_{S,U}(\cdot)$ is negligible for every poly-time *U*.

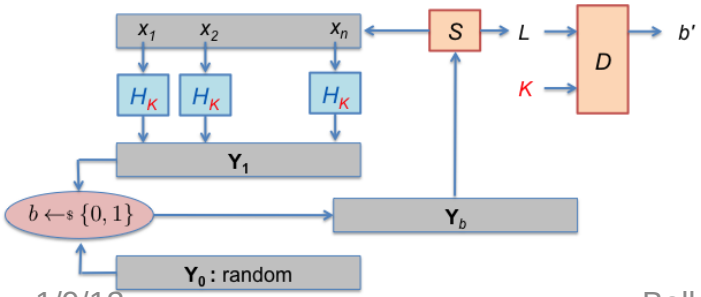
Note: The hash function *H* appears nowhere; unpredictability is in the ROM.

UCE generalizes existing definitions



UCE generalizes existing definitions

Definition	Leakage	Unpredictability	Indistinguishability	Correlated inputs
Hardcore functions [BlMi81,Y82]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	No
Hardcore functions on correlated inputs [FOR12]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	Yes
Randomness extractors [NZ93,DORS08]	$\mathbf{Y}_b$, $F(x_i)$	Statistical	Statistical	No
Computational extractors [FPZ08,K10,DGKM12]	$\mathbf{Y}_b$	Statistical	Computational	No
Correlated-input secure functions [GOR11]	$\mathbf{Y}_b$	Statistical	Computational	Yes
UCE [BK13]	Any	Computational	Computational	Yes



UCE generalizes existing definitions

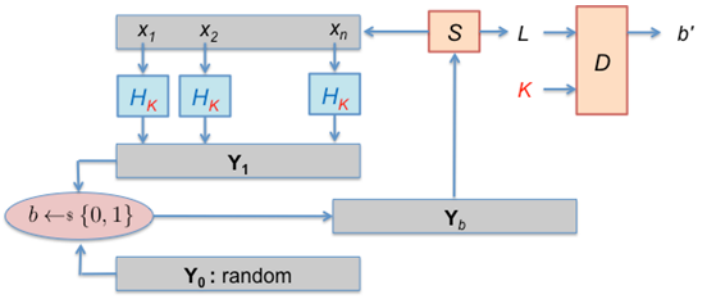
Definition	Leakage	Unpredictability	Indistinguishability	Correlated inputs
Hardcore functions [BlMi81,Y82]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	No
Hardcore functions on correlated inputs [FOR12]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	Yes
Randomness extractors [NZ93,DORS08]	$\mathbf{Y}_b$, $F(x_i)$	Statistical	Statistical	No
Computational extractors [FPZ08,K10,DGKM12]	$\mathbf{Y}_b$	Statistical	Computational	No
Correlated-input secure functions [GOR11]	$\mathbf{Y}_b$	Statistical	Computational	Yes
UCE [BK13]	Any	Computational	Computational	Yes

Definition	Leakage	Unpredictability	Privacy	Correlated inputs
DE [BBO07,BFOR08]	$\mathbf{Y}_b$	Statistical	Computational	Yes
DE with auxiliary inputs [BrSe11]	$\mathbf{Y}_b$, $F(x)$	Computational	Computational	Yes

UCE extends standard definitions

Definition	Leakage	Unpredictability	Indistinguishability	Correlated inputs
Hardcore functions [BIMi81,Y82]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	No
Hardcore functions on correlated inputs [FOR12]	$\mathbf{Y}_b$, $F(x_i)$	Computational	Computational	Yes
Randomness extractors [NZ93,DORS08]	$\mathbf{Y}_b$, $F(x_i)$	Statistical	Statistical	No
Computational extractors [FPZ08,K10,DGKM12]	$\mathbf{Y}_b$	Statistical	Computational	No
Correlated-input secure functions [GOR11]	$\mathbf{Y}_b$	Statistical	Computational	Yes
UCE [BK13]	Any	Computational	Computational	Yes

Definition	Leakage	Unpredictability	Privacy	Correlated inputs
DE [BBO07,BFOR08]	$\mathbf{Y}_b$	Statistical	Computational	Yes
DE with auxiliary inputs [BrSe11]	$\mathbf{Y}_b$, $F(x)$	Computational	Computational	Yes



UCE:

- Considers a more general form of leakage than other primitives
- Goes "computational" on all fronts.

UCE: Features and Limitations

Allows instantiation of ROs when the inputs to which the good parties apply the RO are unknown (unpredictable) to the adversary.

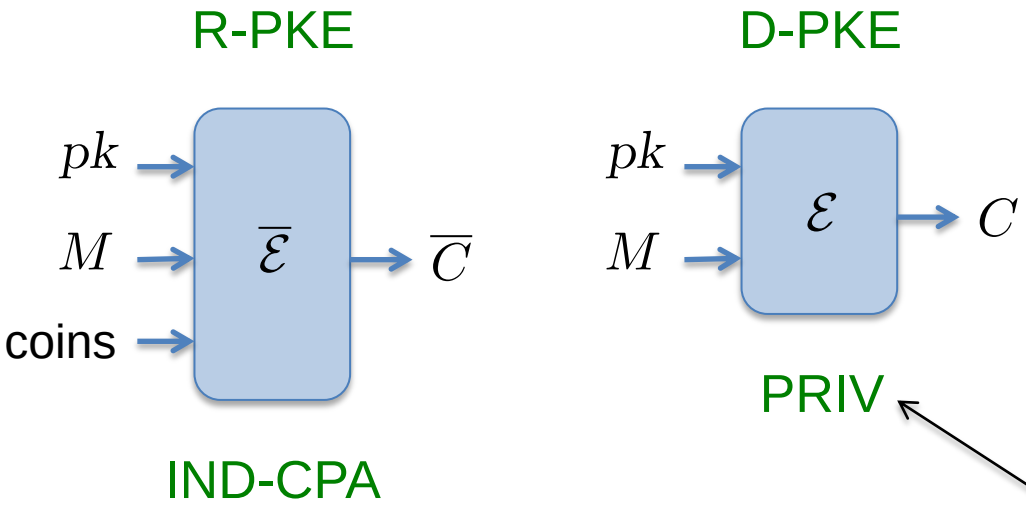
Good for many forms of encryption: DE, HE, MLE, PKE, KDM, RKA, …

Does not handle CCA.

Does not handle signatures, IBE.

The hard part is usually to prove unpredictability of the source. But this is a ROM claim!

Deterministic PKE [BBO07]



Semantic-security on unpredictable, *pk*-independent messages.

D-PKE provides efficiently searchable encryption of records in outsourced databases.

Instantiating EwH

ROM EwH D-PKE scheme [BBO07]:

$$\mathcal{E}_{pk}(M) = \overline{\mathcal{E}}_{pk}(M; \mathrm{HASH}(pk\|M))$$

D-PKE R-PKE coins

Theorem [BBO07]**:** If $\overline{\mathcal{E}}$ is IND-CPA-secure and HASH is a RO then $\mathcal{E}$ is PRIV-secure.

Theorem [BK13]**:** If $\overline{\mathcal{E}}$ is IND-CPA-secure and *H* is UCE-secure then the instantiated scheme

$$\mathcal{E}_{(pk,K)}(M) = \overline{\mathcal{E}}_{pk}(M; H(K, pk\|M))$$

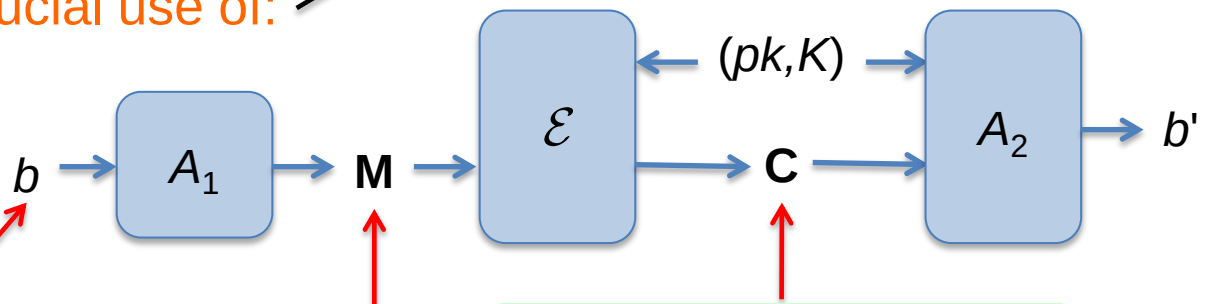
is PRIV-secure.

First standard-model PRIV-secure D-PKE scheme.
Previous standard model schemes achieved security for restricted sources [BoFeON08,BrSe11,FuONRe12].

Why it works

Theorem [BFOR08]: PRIV ≡ IND

We will make crucial use of:



Random challenge bit.

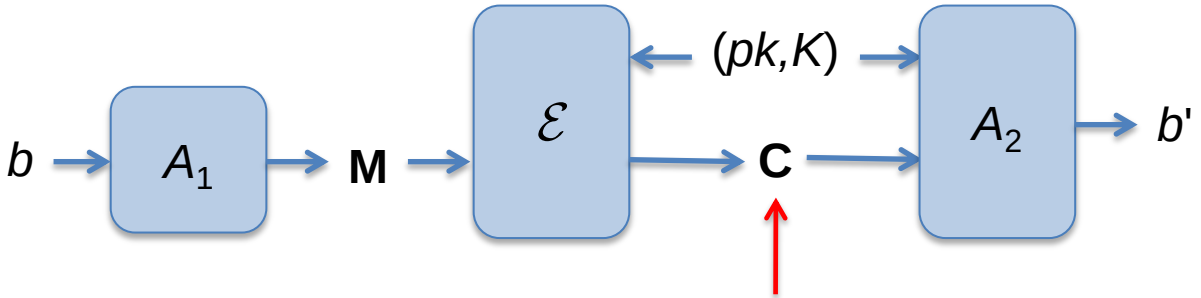
Vector of messages, each component message having high min-entropy.

$\overline{\mathcal{E}}pk(\mathbf{M}; H(K, pk||\mathbf{M}))$

Vector of ciphertexts formed by component-wise encryption.

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{E},(A_1,A_2)}(k) = 2\Pr[b' = b] - 1$$

Why it works



Source S

Pick (pk, sk)
Pick random challenge bit b
Run $A_1(b)$ to get message vector $\mathbf{M}$
Let $\mathbf{C} \leftarrow \overline{\mathcal{E}}_{pk}(\mathbf{M}; \text{HASH}(pk||\mathbf{M}, l))$
Return leakage $L \leftarrow (\mathbf{C}, b, pk)$

Distinguisher $D(K, L)$

Parse L as $(\mathbf{C}, b, pk)$
Run $A_2((pk, K), \mathbf{C})$ to get b'
If $(b' = b)$ then return 1
Else return 0

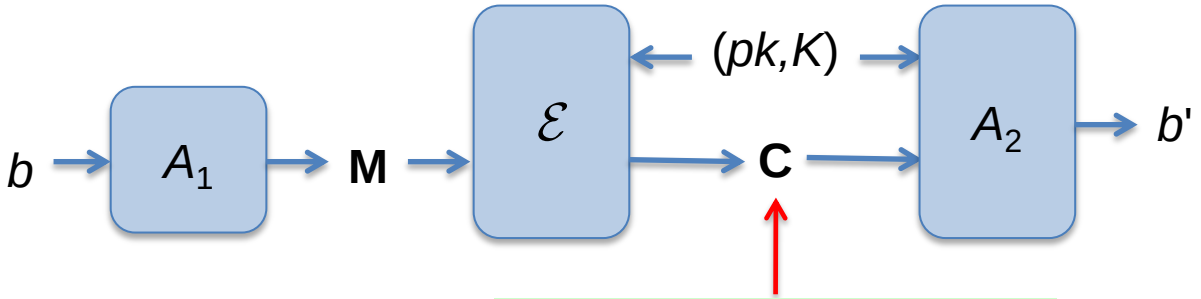
Claim 1: S is unpredictable

If not, one-wayness of the ROM scheme is violated.

Claim 2: $\mathbf{Adv}^{\text{ind}}_{\mathcal{E},(A_1,A_2)}(k) \leq 2 \cdot \mathbf{Adv}^{\text{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) + \text{negl}(k)$

By PRIV-security of the ROM scheme.

Why it works



$\overline{\mathcal{E}}pk(\mathbf{M}; H(K, pk||\mathbf{M}))$

Source S

Pick (pk, sk)
Pick random challenge bit b
Run $A_1(b)$ to get message vector $\mathbf{M}$
Le
Re

Proofs of both Claims exploit in a blackbox way the known PRIV-security of the ROM scheme from [BBO07].

If $(b' = b)$ then return 1
Else return 0

Claim 1: S is unpredictable

If not, one-wayness of the ROM scheme is violated.

Claim 2: $\mathbf{Adv}^{\text{ind}}_{\mathcal{E},(A_1,A_2)}(k) \leq 2 \cdot \mathbf{Adv}^{\text{uce}}_{(\mathcal{K}_H,H),(S,D)}(k) + \text{negl}(k)$

By PRIV-security of the ROM scheme.

More instantiations in the same vein …

Hedged PKE [BBNRSSY09] provides the best possible privacy in the face of system RNG failures.

We can instantiate the RO in the Hedged PKE scheme of [BBNRSSY09].

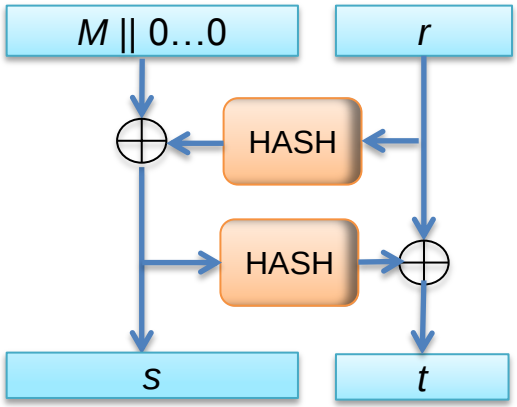
Message-Locked Encryption (MLE) [BKR12] allows secure de-duplicated storage. CE [DABST02] is a practical MLE scheme proven secure in the ROM by [BKR12].

Presentation Friday 3:15pm!

We can instantiate the RO in the CE MLE scheme.

OAEP [BR94]

$\underline{\mathcal{E}_f(M)}$ // $|M| = k_1$
$r \leftarrow\!\!\$\ \{0,1\}^{k_0}$
$s \leftarrow \text{HASH}(r, k_2) \oplus M\|0^{k_2-k_1}$
$t \leftarrow \text{HASH}(s, k_0) \oplus r$
Return $f(s\|t)$



$\underline{\mathcal{E}_{(f,K)}(M)}$ // $|M| = k_1$
$r \leftarrow\!\!\$\ \{0,1\}^{k_0}$
$s \leftarrow H(K, r, k_2) \oplus M\|0^{k_2-k_1}$
$t \leftarrow H(K, s, k_0) \oplus r$
Return $f(s\|t)$

Theorem: In the ROM, OAEP is

- IND-CPA secure if *f* is one-way [BR94]
- IND-CCA secure if *f* is partial-domain one-way [FOPS01]

Theorem [BK13]**:** If *H* is UCE-secure then instantiated OAEP is

- IND-CPA' secure if *f* is partial-domain one-way

IND-CPA for messages not depending on the public key.

Note: One-way and partial-domain one-way are equivalent for RSA [FOPS01].

Previous work: [KOS10] show RSA-OAEP is IND-CPA-secure under a weaker assumption on *H* (*t*-wise independence) and a stronger assumption on RSA (ϕ-hiding).

Multi-key UCE

We also give a definition of UCE-security for the case that hashing is being performed with many different keys.

We do not know whether single-key UCE-security implies multi-key UCE-security in general. (The hybrid argument breaks down.)

However, we can show that single key UCE-security implies multi-key UCE-security in the case that the number of keys is a constant.

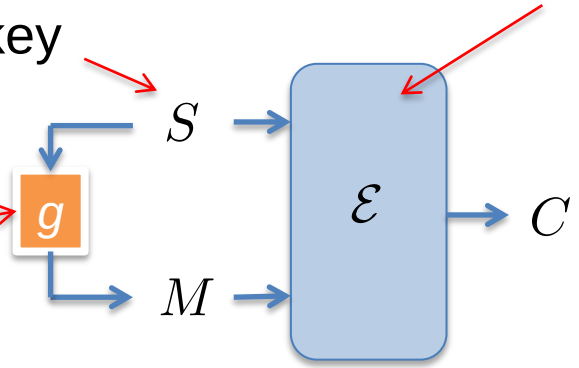
We exploit multi-key security in a crucial way for instantiating the RO in the [BRS03] KDM-secure encryption scheme and in RKA-secure encryption.

KDM-secure encryption [CL01,BRS02]

Symmetric encryption.
Randomized.

Encryption key

Definition of security for key-dependent messages [BRS02] in which this is chosen by the adversary



BRS02 scheme:

$\underline{\mathcal{E}_L(M)}$
Pick R at random
Return $C \leftarrow (R, \text{HASH}(R\|L) \oplus M)$

Theorem [BRS02]**:** If HASH is a RO then $\mathcal{E}$ is KDM-secure.

Instantiated scheme, first try:

$\underline{\mathcal{E}_{(L,K)}(M)}$
Pick R at random
Return $C \leftarrow (R, H(K, R\|L) \oplus M)$

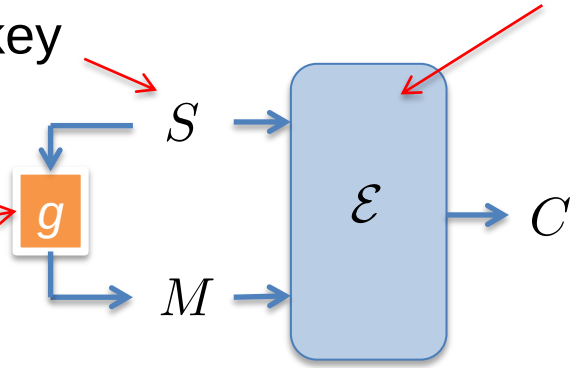
This won't work!
For UCE, *M* cannot depend on *K*.
But for KDM, it must.

KDM-secure encryption [CL01,BRS02]

Symmetric encryption. Randomized.

Encryption key

Definition of security for key-dependent messages [BRS02] in which this is chosen by the adversary



BRS02 scheme:

$\underline{\mathcal{E}_L(M)}$
Pick R at random
Return $C \leftarrow (R, \text{HASH}(R\|L) \oplus M)$

Theorem [BRS02]**:** If HASH is a RO then $\mathcal{E}$ is KDM-secure.

Our instantiated scheme:

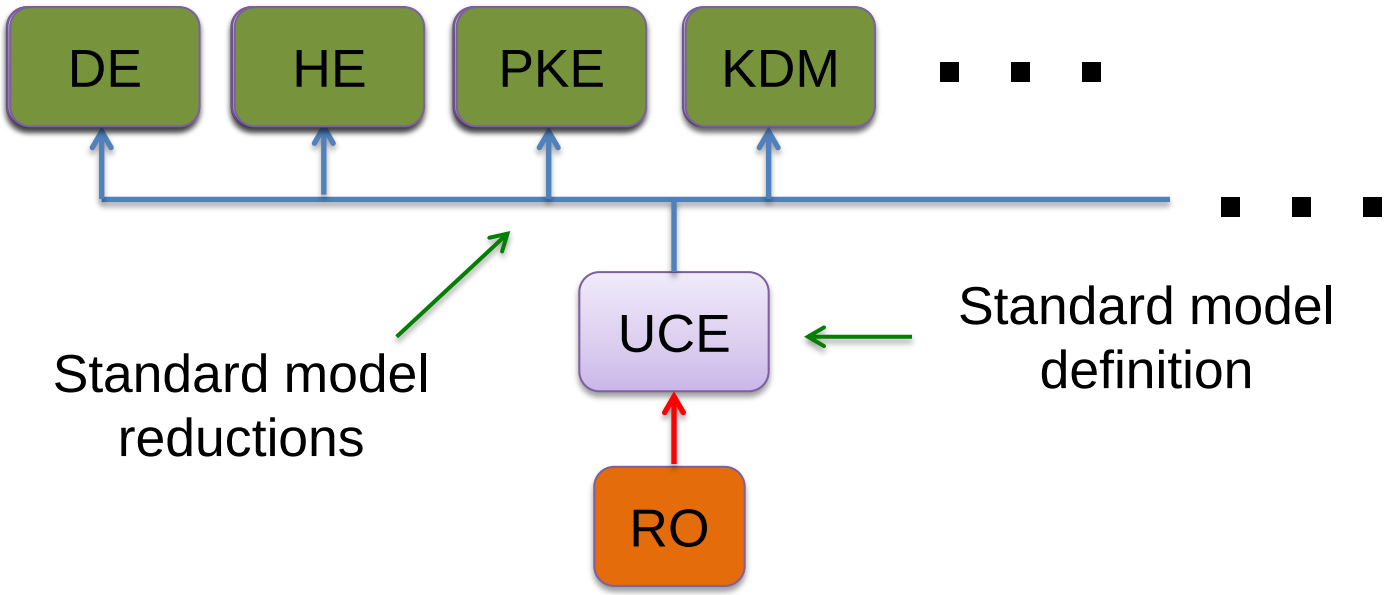
$\underline{\mathcal{E}_L(M)}$
$K \leftarrow\!\!\$\ \mathcal{K}_H$
Return $C \leftarrow (K, H(K, L) \oplus M)$

Theorem [BK13]**:** If *H* is multi-key UCE-secure then $\mathcal{E}$ is non-adaptive KDM-secure.

UCE hash functions from ROs

Theorem [BK13]**:** If HASH is a RO then $H(K, M, l) = \mathrm{HASH}(K \| M, l)$ is (multi-key) UCE-secure.

In practice, instantiate via a cryptographic hash function.



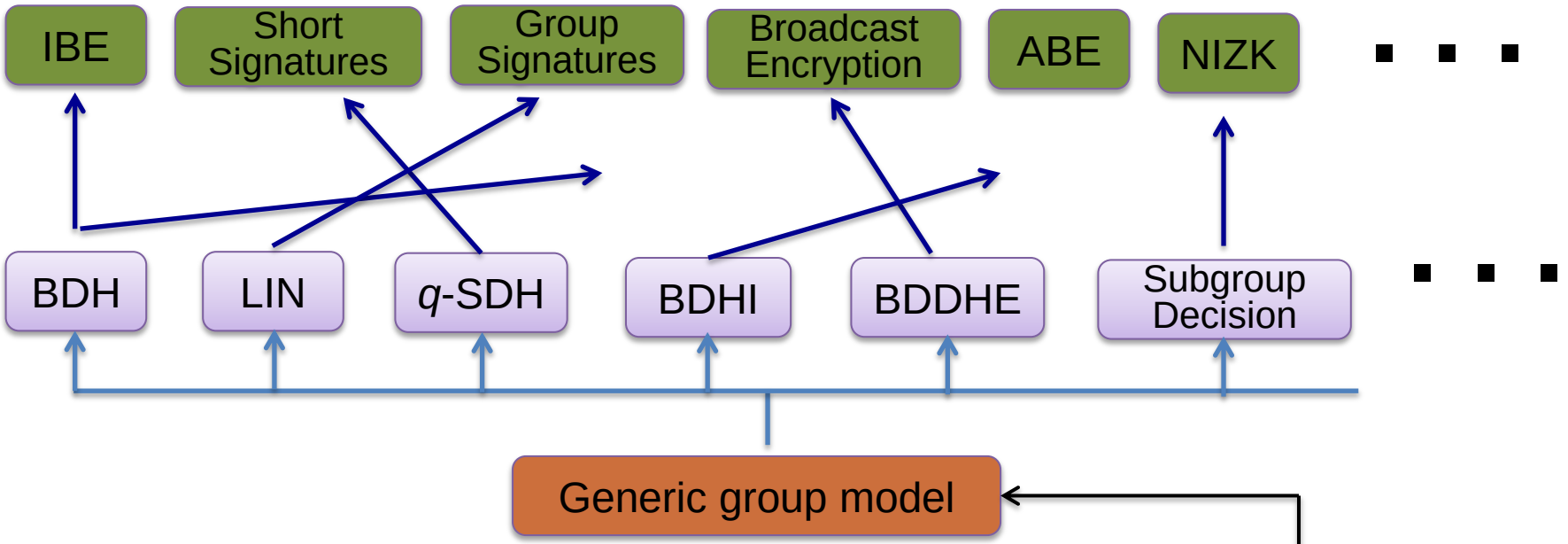
Strength of UCE

Is UCE too ``close'' to the applications derived from it?

- Perhaps true in some cases (DE, HE, MLE) yet the proofs are not completely direct
- Less so in other cases (OAEP, KDM)

We did not realize prior to UCE that all these applications relied on the same properties of the RO.

Pairing-based cryptography

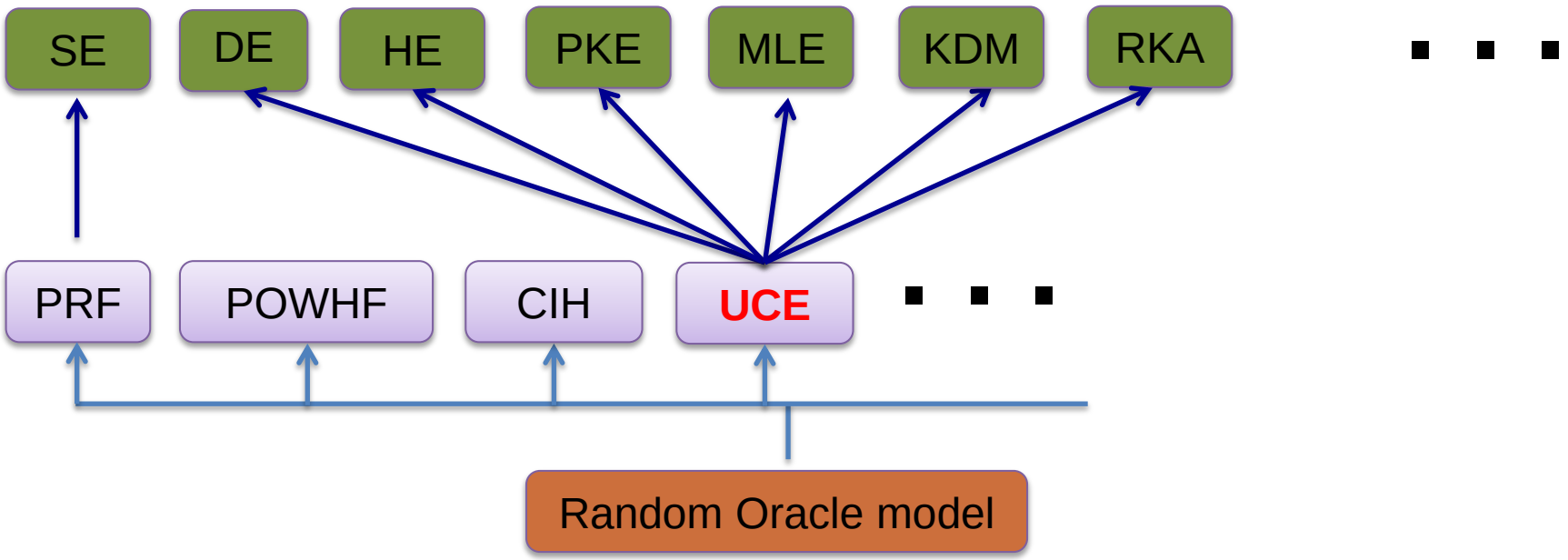


This paradigm works!
Get new capabilities.
Understand what we assume.
Use falsifiable, standard-model assumptions.

Issues and counter-examples similar to (or worse than!) those for the ROM [Fi00,De02]

Why has ROM-based cryptography not taken a similar direction?

A vision of ROM-based cryptography



UCE hash functions from ``standard'' assumptions?

Natural target: Independent inputs; block sources. Potentially useful:

- Lossy TDFs [PW08] as used in [BoFeON08,BrSe11]
- Augmented Crooked LHL [KOS10]

Goldreich-Levin hardcore bits fail for correlated inputs.

We expect achieving (full) UCE-security under standard assumptions to be challenging since it implies several things not yet known to be achievable under standard assumptions.

UCE hash functions from ``standard'' assumptions?

[RSS11] draws attention to how the number of stages of an adversary can affect achievability of notions of security.

A UCE adversary is multi-stage.

[Wi13] implies that proving UCE based on an assumption that is a challenger-adversary game may be hard. This

- Rules out achieving UCE based on assumptions with single-stage adversaries
- But does not rule out achieving it from assumptions that involve multi-stage adversaries.

Contributions in brief

UCE (Universal Computational Extractors): A new DEFINITION of security for hash functions.

Instantiating Ros: UCE hash functions can provably instantiate ROs in a variety of existing schemes: DE, HE, MLE, PKE (OAEP), KDM, RKA, …

Generalizaton: UCE extends and unifies existing definitions like hardcore functions, extractors, correlated-input functions, …

Modular design: Instantiate RO in (existing and) practical ROM schemes, not design new schemes!

Modular proofs: (Instantiable RO paradigm): Proofs of instantiated schemes use results on the ROM security of the scheme in a blackbox way.