The Pseudorandom Oracle Model and Ideal Obfuscation

Aayush Jain Rachel Lin Ji Luo Daniel Wichs

Carnegie Mellon University

University of Washington

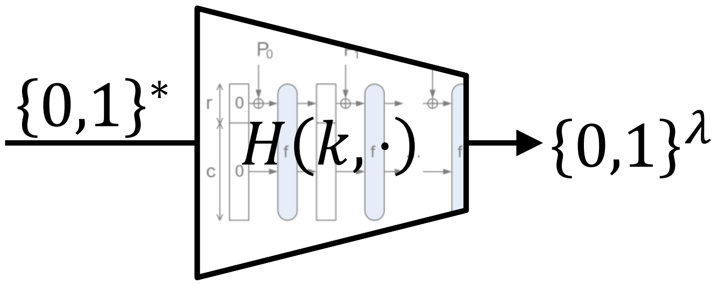
Northeastern University

NTT Research

Outline

- Motivation and Result
- **P**seudo**r**andom **O**racle **M**odel ($\mathrm{Pr}\mathcal{O}\mathrm{M}$)
- Ideal Obfuscation in $\mathrm{Pr}\mathcal{O}\mathrm{M}$

Hash Functions



basic, useful primitive

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

$$\mathrm{Sign}(\mathrm{sk}, m) \overset{\mathrm{def}}{=} \mathrm{Sign}_\lambda\big(\mathrm{sk}_\lambda, H(k, m)\big)$$

✓ collision resistance

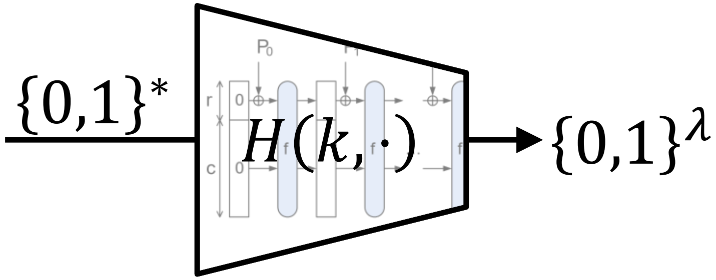
$$\mathrm{Sign}(\mathrm{sk}, m) \overset{\mathrm{def}}{=} \mathrm{TDP}^{-1}\big(\mathrm{sk}, H(k, m)\big)$$

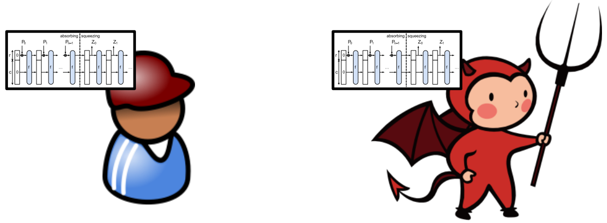
What assumption for H?

desired security not always captured by simple complexity assumptions

Hash Functions: Idealization

Standard Model

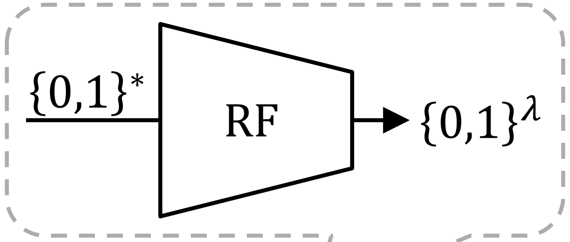


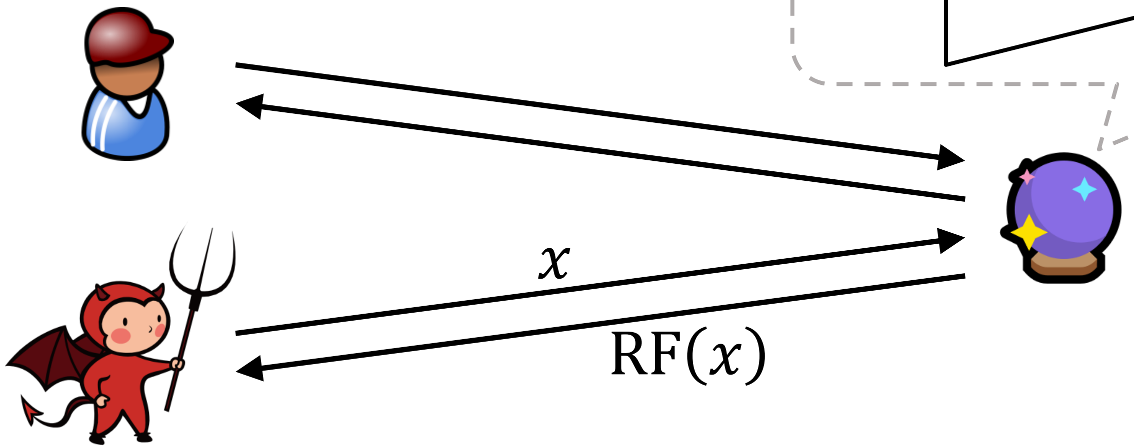


- one-wayness
- second-preimage resistance
- collision resistance

- correlation intractability

Random Oracle Model[BR]





✓ **practically used** and **more efficient** schemes

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

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

✓ proof in ROM **>>** no proof

🚩 if can**not** write proof in ROM

✓ **precursor** to standard-model version

Obfuscation (for Circuits)

```
basicFun = Function[{Typed[pixel0, "ComplexReal64"]},
    Module[{iters = 1, maxIters = 1000, pixel = pixel0},
      While[iters < maxIters && Abs[pixel] < 2,
        pixel = pixel^2 + pixel0;
        iters++
      ];
      iters]];
```

$\mathrm{Obf}(\cdot)$ →

$$\mathrm{Obf}(C)\,(x) = C(x)$$

$\mathrm{Obf}(C)$ is "**unintelligible**"

Indistinguishability Obfuscation

$$C_0 \equiv C_1,\ |C_0| = |C_1| \quad \Longrightarrow \quad \mathrm{Obf}(C_0) \approx \mathrm{Obf}(C_1)$$

✓ **already very powerful**

deniable encryption, short signatures, injective TDF, NIZK, OT[SW]

FHE[ABFGGTW] FE for P[GGHRSW] 2-round MPC[GGHR]

succinct garbled RAM[CH] (+ many more!)

✓ **from well-studied assumptions**[JLS]

⚠ **weak, unintuitive security guarantee**

⚠ **complicated design in applications**

Simulation-Secure Obfuscation (Idealized)

Ideal Obfuscation.

$$\exists S \quad \forall C:$$

$$\mathrm{Obf}(C) \approx S^C\left(1^{|C|}, 1^{|x|}\right)$$

✗ **impossible for unlearnable circuits**

Virtual Black-Box.

$$\forall \text{ 1-bit } A \;\; \exists S_A \;\; \forall C:$$

$$A(\mathrm{Obf}(C)) \approx S_A^C\left(1^{|C|}, 1^{|x|}\right)$$

⚠ **VBB not possible in general**[BGIRSVY]

(contrived "self-eating" programs)

✓ **strong, intuitive security guarantee**

✓ **simple, intuitive designs in applications**

✓ **only path to certain (plausible) applications**

doubly efficient PIR[BIPW] FHE for RAM[HHWW]

VGB obfuscation[BC] time-optimal FE for RAM[ACFQ]

OT from binary erasure channel[AIKNPPR]

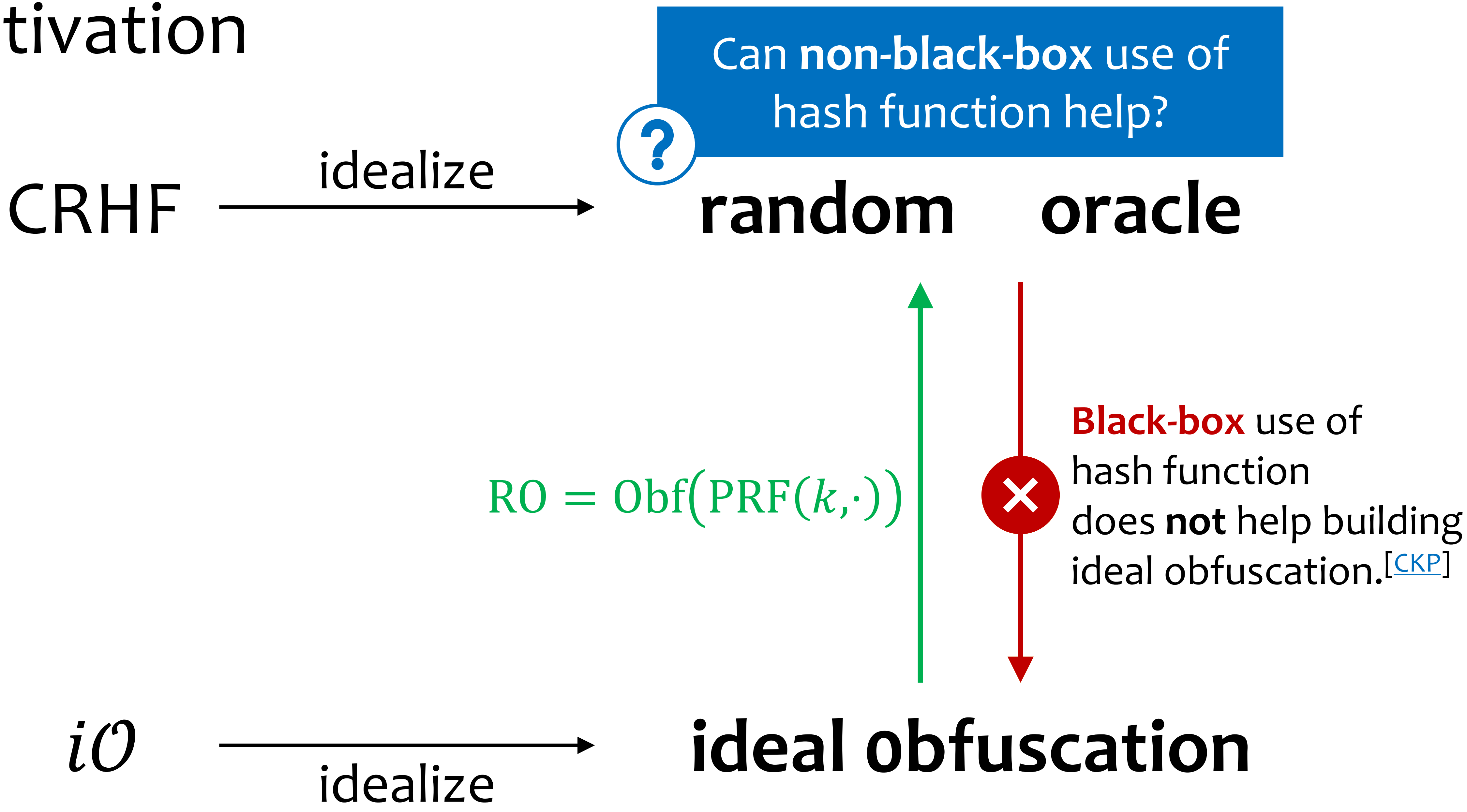
public-coin d$i\mathcal{O}$, unbounded-input obfuscation for TM[IPS]

input-hiding obfuscation for evasive functions[BBCKPS]

extractable WE[GKPVZ] wiretap-channel coding[IKLS]

refuting dream XOR lemma[BIKSW]

Motivation


Can **non-black-box** use of hash function help?
CRHF
idealize
random oracle
$\mathrm{RO} = \mathrm{Obf}(\mathrm{PRF}(k,\cdot))$
Black-box use of hash function does **not** help building ideal obfuscation.[CKP]
$i\mathcal{O}$
idealize
ideal obfuscation

Our Result

CRHF —idealize→ **pseudorandom oracle** (morally ROM but more **flexible**)

Theorem

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

- **Justifies** all downstream applications.
  - OK to use $i\mathcal{O}$ in the applications ($i\mathcal{O}$ is best-possible[GR]).
- $i\mathcal{O}$ from **polynomially secure** FE.

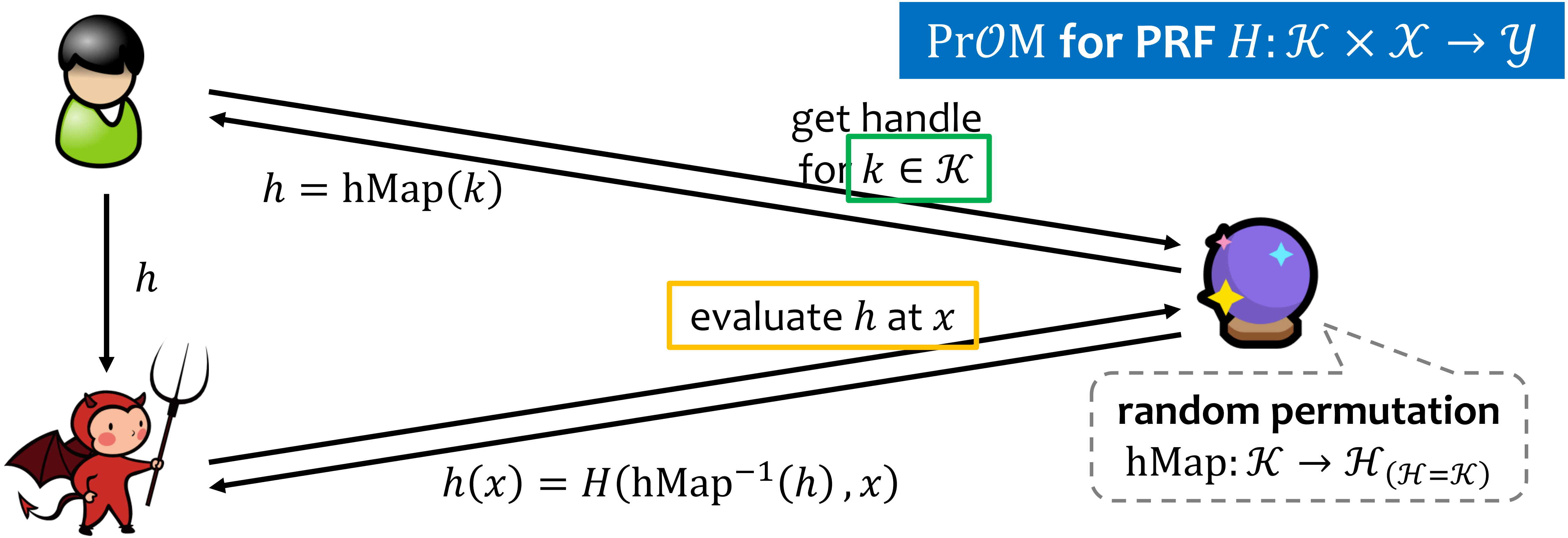
pseudorandom oracle —+ FE→ ideal obfuscation

$i\mathcal{O}$ —idealize→ **ideal obfuscation**

Pseudorandom **Oracle** **Model** (Pr$\mathcal{O}$M)

Two aspects of the model.

- H looks like a **random function** **(using h is like ROM)**
- H has (short) **code** **(using k is like usual PRF)**



Two Ways of Using H in Pr$\mathcal{O}$M

- with h – call oracle to evaluate H **only use h: just ROM**
- with k – use code of H **only use k: just PRF**

flexibility of Pr$\mathcal{O}$M: use in both ways, simultaneously

But when k is seen by adversary,
h is just a usual function, **not** a random oracle!

Basic Recipe of Using Pr$\mathcal{O}$M

$h \leftarrow \mathrm{hMap}(k \leftarrow \$)$, $\mathrm{FHE/GC/FE}(k)$

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

⚠ h **cannot be monitored or programmed**

$\approx$

$h \leftarrow \mathrm{hMap}(k \leftarrow \$)$, $\mathrm{Sim}(\{H(k, x_i)\}_i)$

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

(FHE/GC/FE security)

$\approx$

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

$h(x) = \mathrm{RF}(x)$

(PRF security of H)

✓ h **can be monitored and programmed**

Limited Use of Code in Pr$\mathcal{O}$M

- with h – call oracle to evaluate H
- with k – use code of H

only use h: just ROM
only use k: just PRF

flexibility of Pr$\mathcal{O}$M: use in both ways, simultaneously

But when k is seen by adversary,
h is just a usual function, not a random oracle!

In hybrid with k removed, h becomes a random oracle!

limited use of code: must $\approx$ hybrid without non-black-box use

The Random Oracle Paradigm[BR]

1. Formally define Π in the ROM.
2. Design a scheme $P^{\mathcal{O}}$ for Π in the ROM.
3. Prove $P^{\mathcal{O}}$ satisfies the definition for Π in the ROM.
4. Instantiate $\mathcal{O}$ by a real hash function.

✓ models intuition of hash functions being public random-looking functions
✓ excludes generic attacks
✓ heuristically justifies security of **practically used** / **more efficient** schemes
⚠ forbids use of code of hash function

impossibility results	simulation-secure FE[AKW]	VBB obfuscation[CKP]
***ad hoc* code use in ROM**	recursive proof composition[BCMS]	

⚠ (contrived) uninstantiability results

The Pseudorandom Oracle Paradigm

1. Formally define Π in the $\mathrm{Pr}\mathcal{O}\mathrm{M}$.
2. Design a scheme $P^{\mathcal{O}}$ for Π in the $\mathrm{Pr}\mathcal{O}\mathrm{M}$.
3. Prove $P^{\mathcal{O}}$ satisfies the definition for Π in the $\mathrm{Pr}\mathcal{O}\mathrm{M}$.
4. Instantiate $\mathcal{O}$ by a real hash function.

✓ models intuition of hash functions being public random-looking functions **with code**
✓ excludes generic attacks
✓ heuristically justifies **goals beyond previous reach** (this work – **ideal obfuscation**)
✓ allows limited use of code

⚠ (contrived) uninstantiability results

How to Instantiate Pr$\mathcal{O}$M

$$\text{good for ROM} \implies \text{good for Pr}\mathcal{O}\text{M}$$

Example. $h = k$ are random strings, $H(k, x) = \mathrm{SHA3}(k \parallel x)$

Rationale.

- Good hash function is "**self-obfuscated PRF**".
- In ROM instantiation, oracle calls are replaced by code of hash function anyway. Using it for Pr$\mathcal{O}$M **does not morally demand more** from this hash function **than for ROM**.

Ideal Obfuscation: Definition

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

$\mathrm{Obf}^{\mathcal{O}}(C) \to \tilde{C}^{\bullet}$: obfuscator

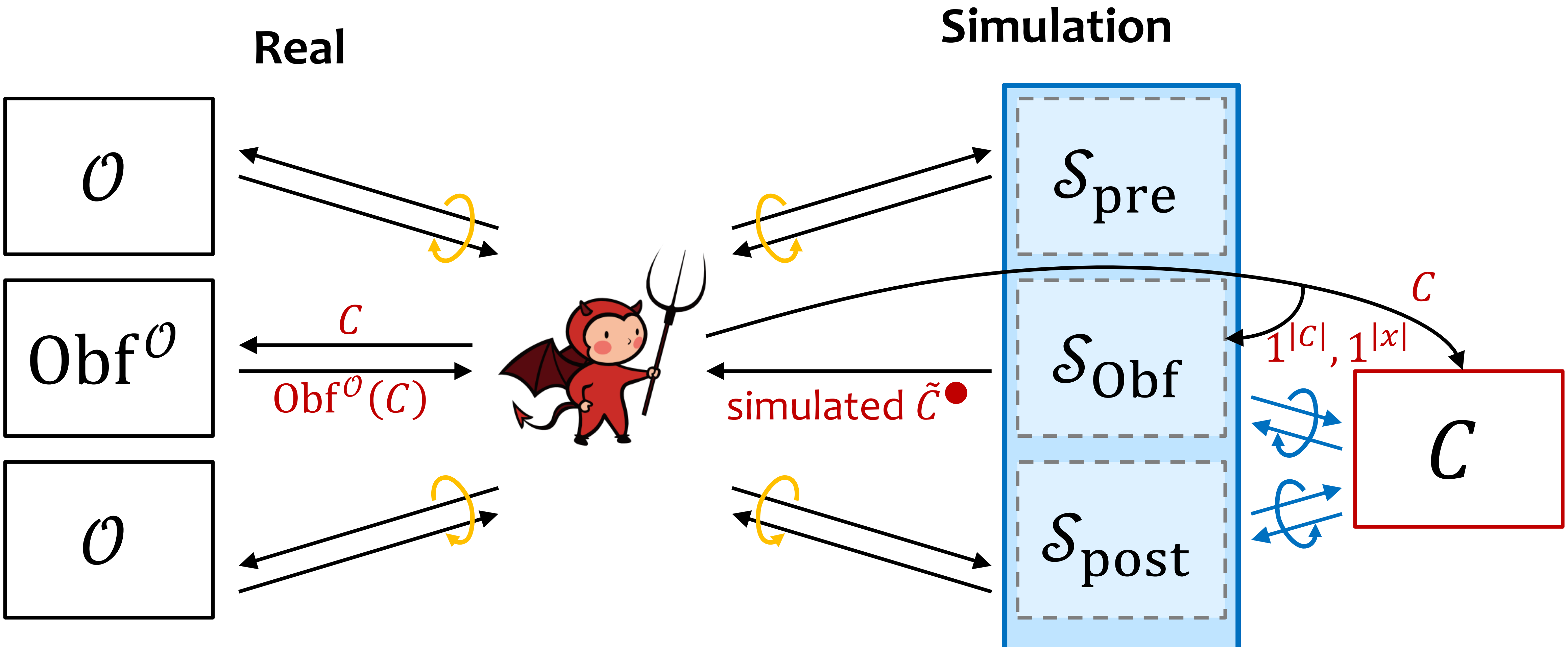
C has **no** **oracle access**

$\tilde{C}^{\bullet}$ has **oracle access**

$$\forall C, x: \quad \Pr\left[\tilde{C}^{\bullet} \leftarrow \mathrm{Obf}^{\mathcal{O}}(C) \;:\; \tilde{C}^{\mathcal{O}}(x) = C(x)\right] = 1.$$

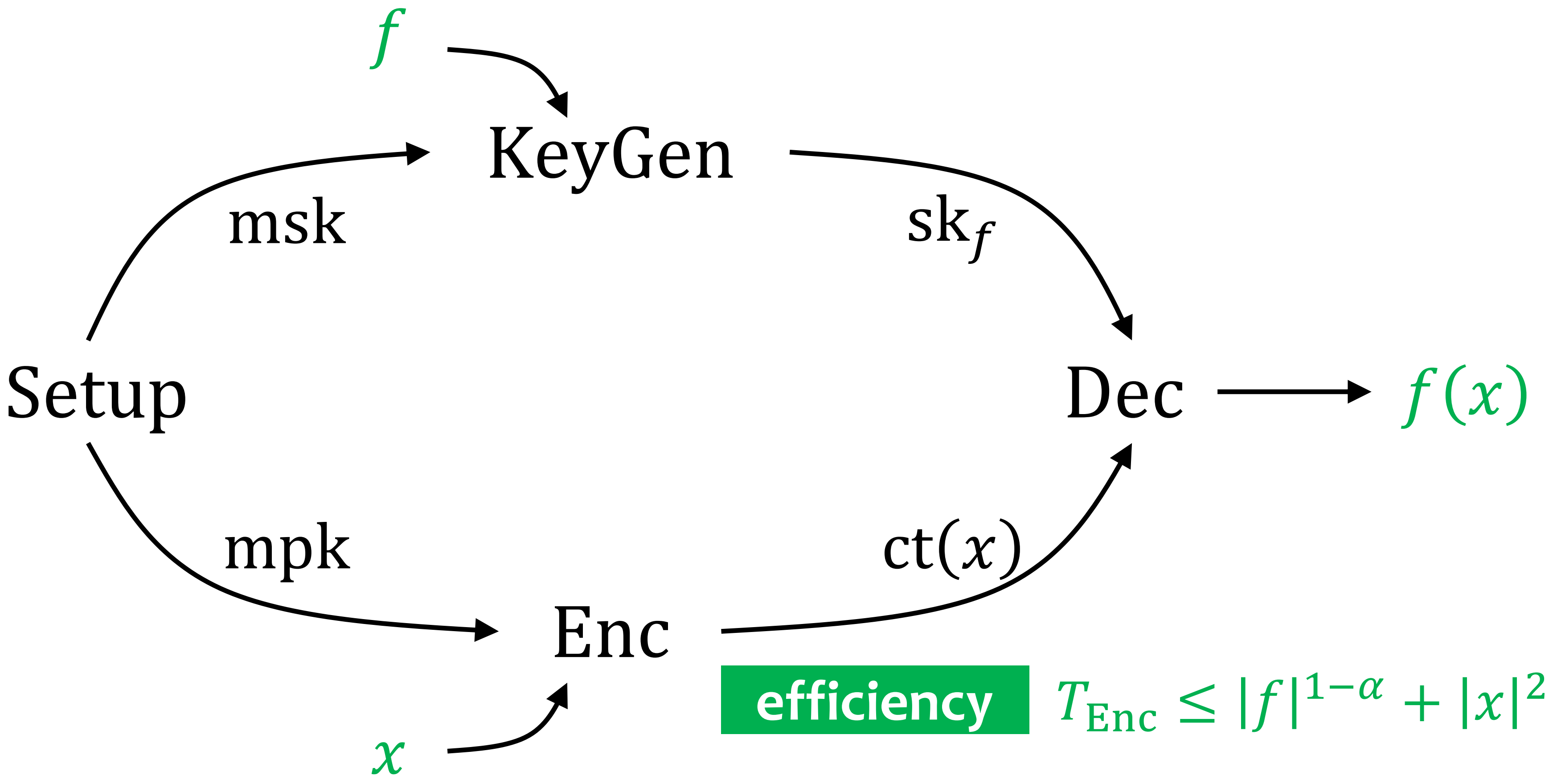
Security is **indifferentiability**-based.[MRH]

Ideal Obfuscation: Security Definition



$$\left(\mathcal{O}, \mathrm{Obf}^{\mathcal{O}}(C), \mathcal{O}\right) \approx \left(\mathcal{S}_{\mathrm{pre}}, \mathcal{S}_{\mathrm{Obf}}^{C}, \mathcal{S}_{\mathrm{post}}^{C}\right)$$

Tool: (Standard-Model) Functional Encryption



efficiency $T_{\mathrm{Enc}} \le |f|^{1-\alpha} + |x|^{2-\alpha}$

adaptive x_0, x_1 chosen after seeing $\mathrm{mpk}, \mathrm{sk}_f$

Security. $\left(\mathrm{mpk}, \mathrm{sk}_f, \mathrm{ct}(x_0)\right) \approx \left(\mathrm{mpk}, \mathrm{sk}_f, \mathrm{ct}(x_1)\right)$ if $f(x_0) = f(x_1)$

Main Theorem

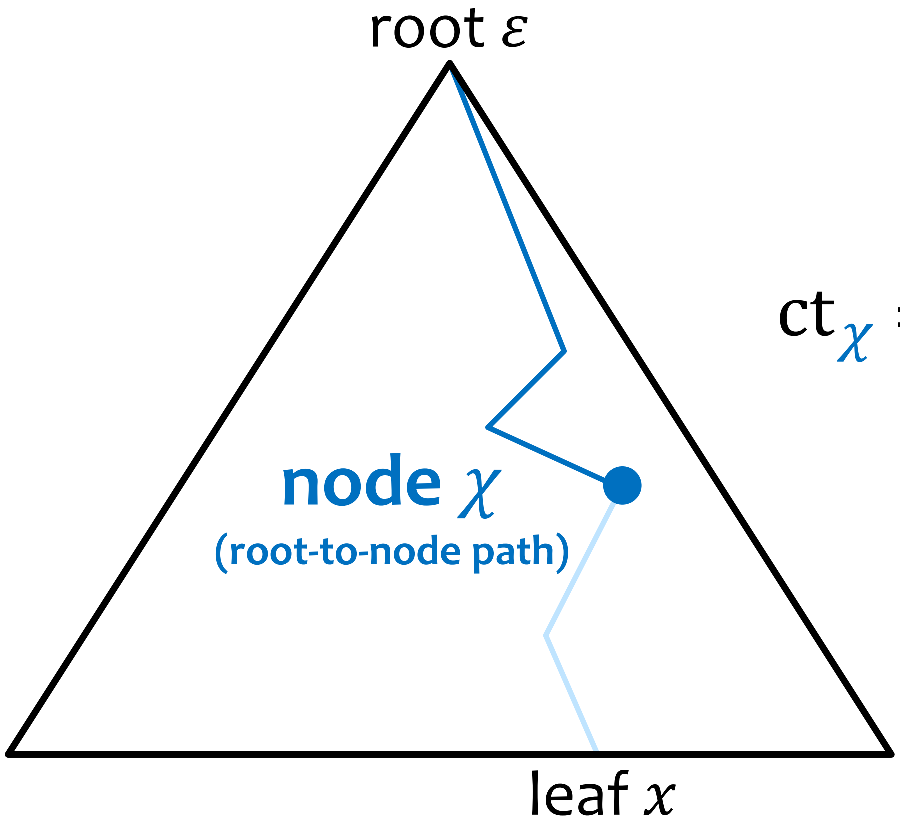
Assuming polynomially secure **succinct** FE for all circuits,
there exists ideal obfuscation **for all circuits**
in the pseudorandom oracle model **for any PRF** H.

Function-revealing encryption (f in Setup, no KeyGen) is sufficient.

Fact. The FE/FRE we need can be based on polynomially secure selective 1-key sublinearly succinct public-key FE, thus on well-founded assumptions.[ABSV, GVW, GS, LM, AJS, BV, AS, KNTY, JLL, JLS]

$i\mathcal{O}$ from FE: Quick Recap

perfect binary tree (leaf ⇔ input)

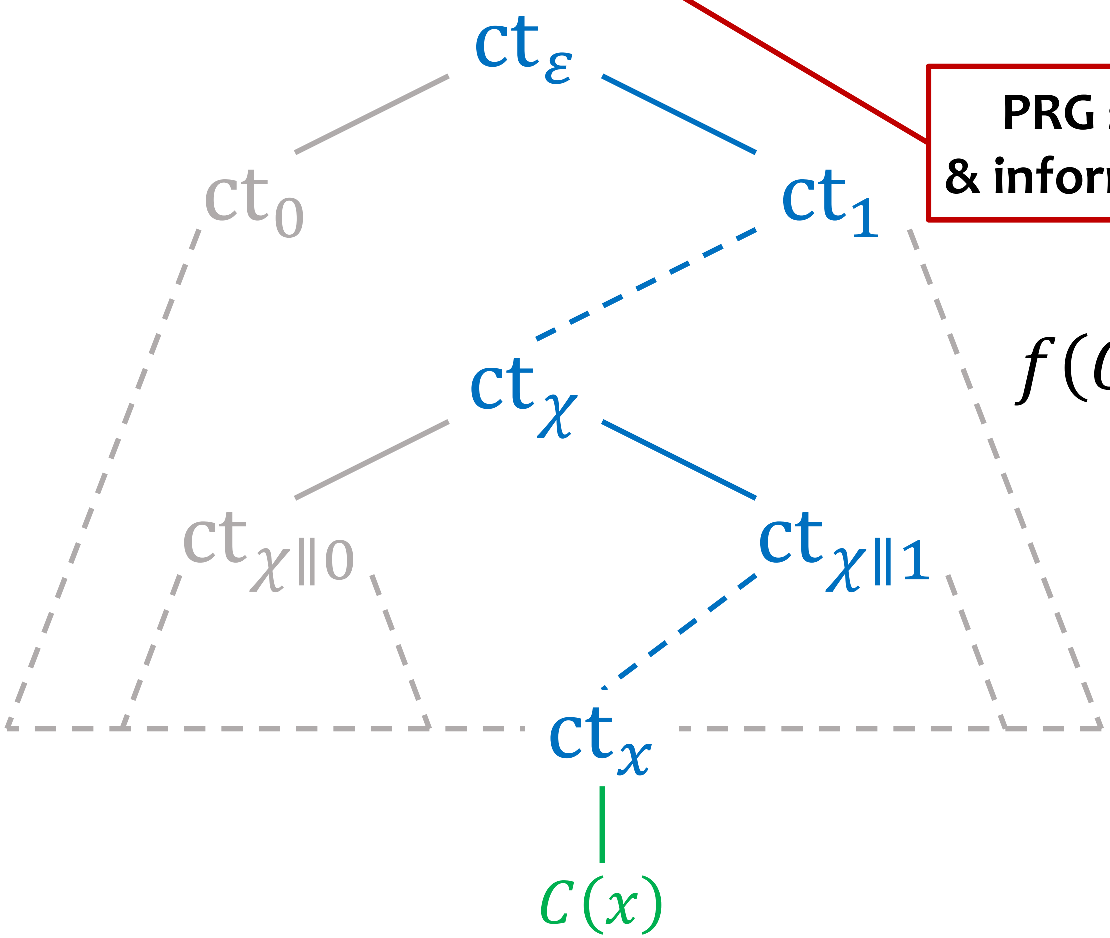


ct_{χ} = FE ciphertext of $(C, \chi, \ldots)$ for $\chi \in \{0,1\}^{\leq D}$

$i\mathcal{O}$ from FE: Traversal and Evaluation

$\mathrm{ct}_\chi(C, \chi, \ldots)$

traverse/evaluate = decrypt using FE key sk_f



PRG seed / PRF key
& information for proof

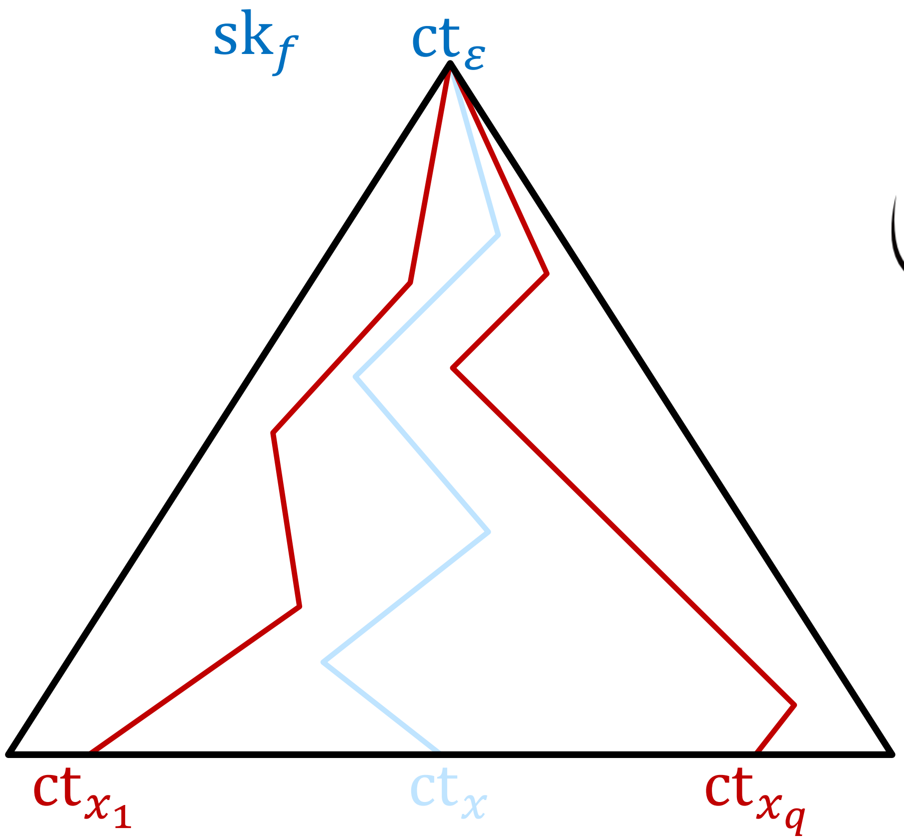
$f(C, \chi, \ldots) = (\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1})$ when $|\chi| < D$

needs $\mathrm{FE.Enc}$ **randomness**

$f(C, \chi, \ldots) = C(\chi)$ when $|\chi| = D$

$i\mathcal{O}$ from FE

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



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

$$\mathrm{Obf}(C) = \left(\mathrm{sk}_f, \mathrm{ct}_\varepsilon\right)$$

← could evaluate at many inputs.
Do **not know where** it explores.
Reduction goes over **every** input.
(**exponential loss**)

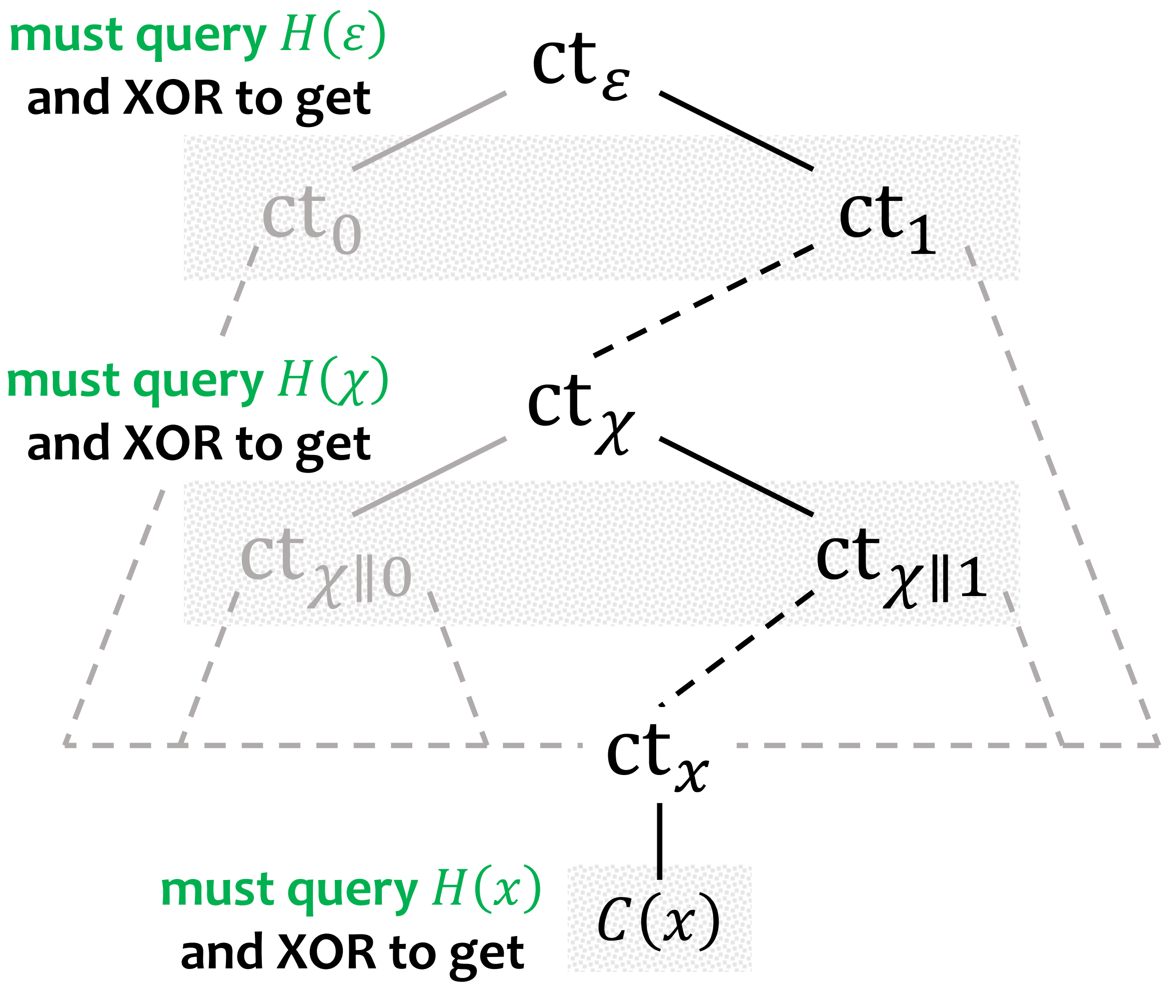
Idealized models could help!

Simplified Idea in ROM

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

$\mathsf{sk}_f : (C, \chi) \mapsto \boxed{H(\chi)} \oplus \big((\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1}) / C(\chi)\big)$

✗ **cannot call RO in circuit sent to** $\mathsf{FE.KeyGen}$



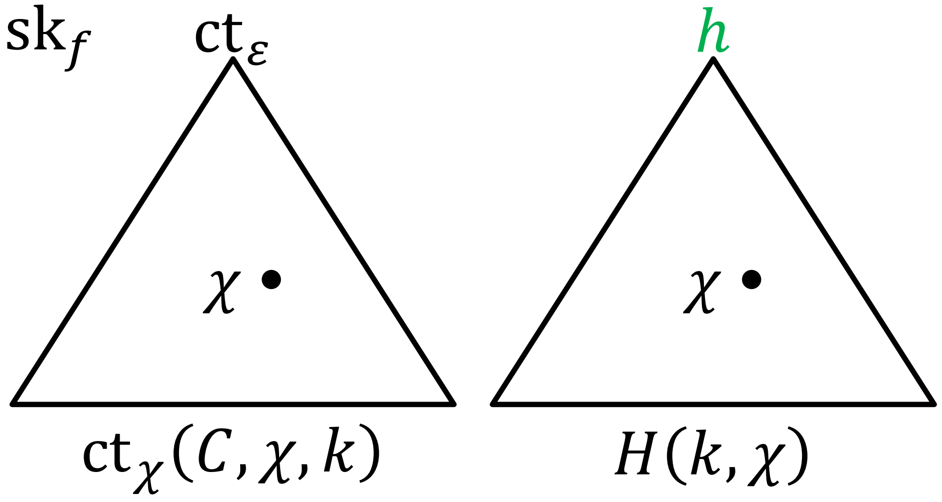
$\mathsf{FE.Dec}(\mathsf{sk}_f, \mathsf{ct}_\chi)$ is **random**
if $H(\chi)$ is **not queried**

observe & program RO
in simulation

First Attempt in Pr$\mathcal{O}$M

use code and k

$\mathrm{ct}_{\chi}(C, \chi, k)$ $\quad$ $\mathrm{sk}_f : (C, \chi, k) \mapsto H(k, \chi) \oplus \left((\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1})/C(\chi)\right)$



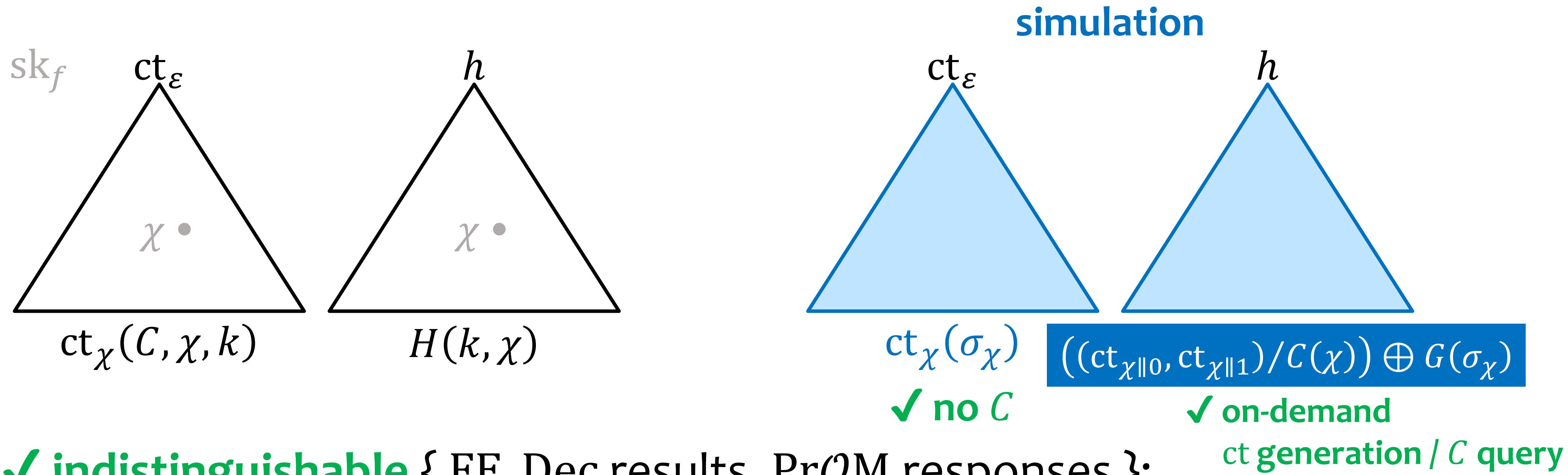
$$\mathrm{Dec}\left(\mathrm{sk}_f, \mathrm{ct}_{\chi}\right) \oplus h(\chi) = (\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1})/C(\chi)$$

call evaluation oracle with h

First Attempt: Simulation

$\mathsf{ct}_\chi \begin{cases} C, \chi, k \\ \boxed{\sigma_\chi} \end{cases}$ **fresh seed for each χ**

$\mathsf{sk}_f \begin{cases} (C, \chi, k) \mapsto H(k, \chi) \oplus ((\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1})/C(\chi)) \\ \boxed{\sigma_\chi \mapsto G(\sigma_\chi)} \end{cases}$ **PRG**



✓ **indistinguishable** { FE. Dec results, Pr$\mathcal{O}$M responses }:

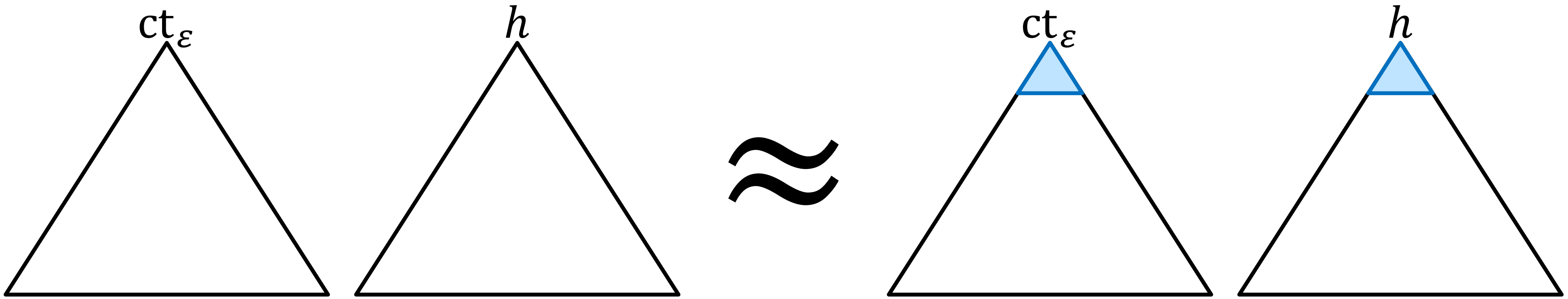
$$\{H(k, \chi) \oplus (\cdots), H(k, \chi)\}_{\chi \in \{0,1\}^{\leq D}} \approx \{G(\sigma_\chi), (\cdots) \oplus G(\sigma_\chi)\}_{\chi \in \{0,1\}^{\leq D}}$$

First Attempt: Proof

1. separate functions for each level

NOT making H puncturable PRF (not good model of hash functions)

$\mathrm{ct}_\chi \begin{cases} C, \chi, k \\ \sigma_\chi \end{cases}$ $\quad \mathrm{sk}_f \begin{cases} H(k,\chi) \oplus (\cdots) \\ G(\sigma_\chi) \end{cases}$ $\quad h \begin{cases} H(k,\chi) \\ G(\sigma_\chi) \oplus (\cdots) \end{cases}$



$\mathrm{ct}_\varepsilon(C, \varepsilon, k)$ $\quad h(\varepsilon) = H(k, \varepsilon)$ $\quad \approx \quad$ $\mathrm{ct}_\varepsilon(\sigma_\varepsilon)$ $\quad h(\varepsilon) = G(\sigma_\varepsilon) \oplus (\mathrm{ct}_0, \mathrm{ct}_1)$

$\mathrm{ct}_0(C, 0, k)$ $\quad h(0) = H(k, 0)$ $\quad\quad$ $\mathrm{ct}_0(C, 0, k)$ $\quad h(0) = H(k, 0)$

$\mathrm{ct}_1(C, 1, k)$ $\quad h(1) = H(k, 1)$ $\quad\quad$ $\mathrm{ct}_1(C, 1, k)$ $\quad h(1) = H(k, 1)$

$\vdots$ $\quad \vdots$ $\quad\quad \vdots$

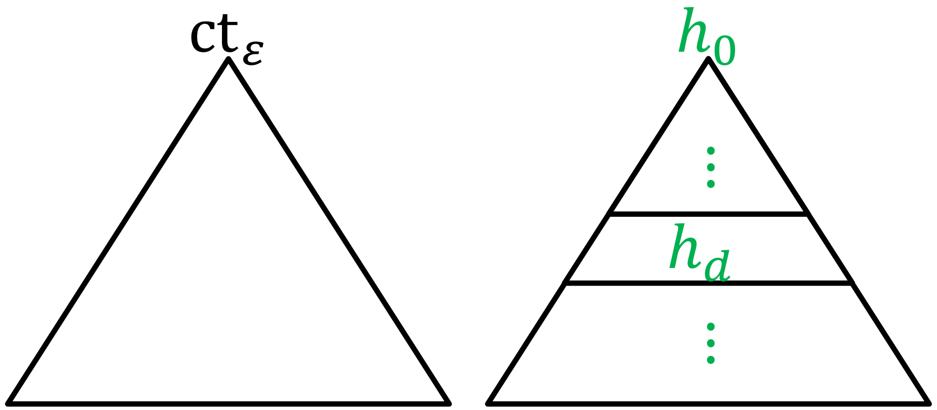
✗ k exists in $\mathrm{ct}_{\neq\varepsilon}$

✓ $h(\varepsilon)$ pseudorandom in both

Second Attempt

1. separate functions for each level

$$\mathrm{ct}_\chi \begin{cases} C, \chi, k_{\geq|\chi|} \\ \sigma_\chi \end{cases} \qquad \mathrm{sk}_f \begin{cases} H(k_{|\chi|}, \chi) \oplus (\cdots) \\ G(\sigma_\chi) \end{cases}$$

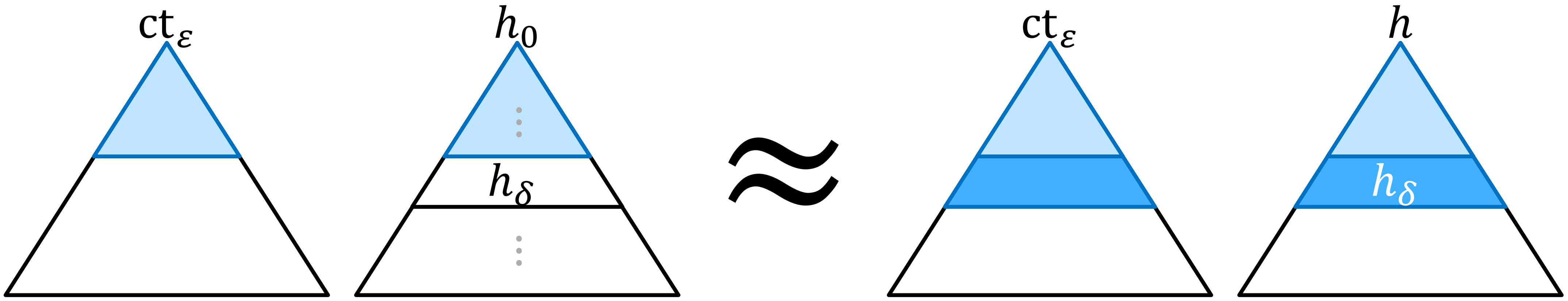


$$(\text{sim.})\ h_d(\chi) \leftarrow \begin{cases} \$, & |\chi| \neq d; \\ G(\sigma_\chi) \oplus (\cdots), & |\chi| = d. \end{cases}$$

Second Attempt: Proof

1. separate functions for each level

$\mathrm{ct}_\chi \begin{cases} C, \chi, k_{\geq|\chi|} \\ \sigma_\chi \end{cases}$ $\quad \mathrm{sk}_f \begin{cases} H(k_{|\chi|}, \chi) \oplus (\cdots) \\ G(\sigma_\chi) \end{cases}$ $\quad h_d \begin{cases} H(k_d, \chi) \\ \$/G(\sigma_\chi) \oplus (\cdots) \end{cases}$



$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$ $\quad h_{<\delta}: \$/\cdots$ $\quad \mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$ $\quad h_{<\delta}: \$/\cdots$

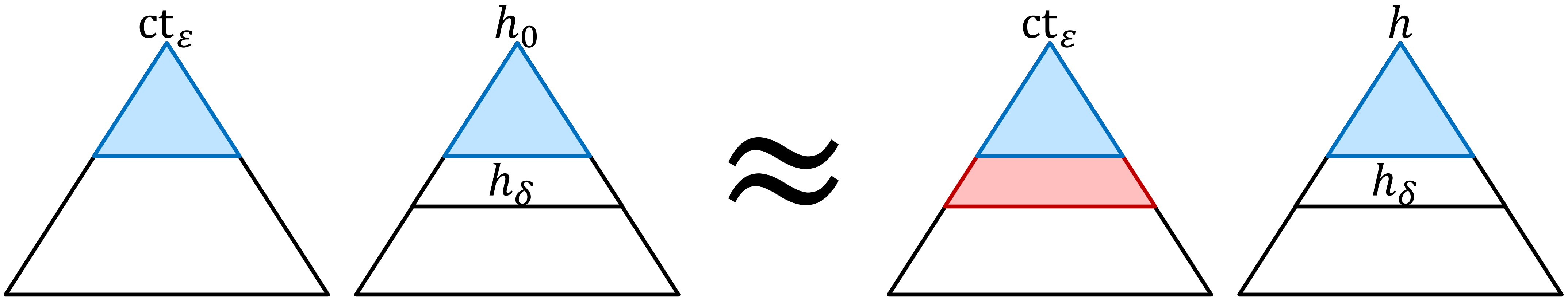
$\mathrm{ct}_{|\chi|=\delta}(C, \ldots)$ $\quad h_\delta: H(k_\delta, \chi)$ $\quad \mathrm{ct}_{|\chi|=\delta}(\sigma_\chi)$ $\quad h_\delta: \$/\cdots$

$\mathrm{ct}_{|\chi|>\delta}(C, \ldots)$ $\quad h_{d>\delta}: H(k_d, \chi)$ $\quad \mathrm{ct}_{|\chi|>\delta}(C, \ldots)$ $\quad h_{d>\delta}: H(k_d, \chi)$

Second Attempt: Hardwiring

1. separate functions for each level

$\mathrm{ct}_\chi \begin{cases} C, \chi, k_{\geq|\chi|} \\ \sigma_\chi \end{cases}$ $\mathrm{sk}_f \begin{cases} H(k_{|\chi|}, \chi) \oplus (\cdots) \\ G(\sigma_\chi) \end{cases}$ $h_d \begin{cases} H(k_d, \chi) \\ \$/G(\sigma_\chi) \oplus (\cdots) \end{cases}$



$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$ $h_{<\delta}: \$/\cdots$

$\mathrm{ct}_{|\chi|=\delta}(C, \ldots)$ $h_\delta: H(k_\delta, \chi)$

$\mathrm{ct}_{|\chi|>\delta}(C, \ldots)$ $h_{d>\delta}: H(k_d, \chi)$

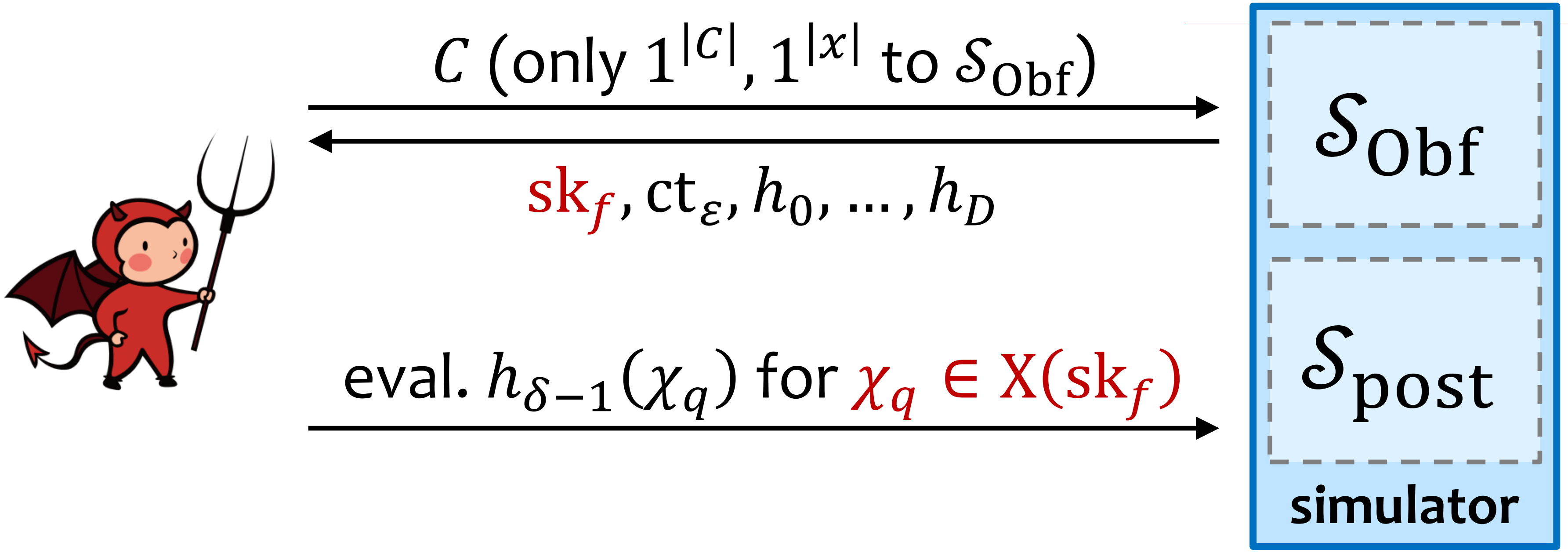
$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$ $h_{<\delta}: \$/\cdots$

$\mathrm{ct}_{|\chi|=\delta} \mapsto H(k_\delta, x) \oplus (\cdots)$ $h_\delta: H(k_\delta, \chi)$

$\mathrm{ct}_{|\chi|>\delta}(C, \ldots)$ $h_{d>\delta}: H(k_d, \chi)$

Hardwire $H(k_\delta, \chi) \oplus (\cdots)$ into sk_f?

Problem with Hardwiring into sk_f



querying $h_{\delta-1}(\chi_q) = G(\sigma_{\chi_q}) \oplus (\mathsf{ct}_{\chi_q \| 0}, \mathsf{ct}_{\chi_q \| 1})$
forces $\mathcal{S}_{\mathsf{post}}$ **to generate** $\mathsf{ct}_{\chi_q \| 0}, \mathsf{ct}_{\chi_q \| 1}$

put $H(k_\delta, \chi) \oplus (\cdots)$
into $\mathsf{ct}_{|\chi|=\delta}$

1. **Cannot predict** (**adaptively** chosen) χ_q's dependent on sk_f.
2. **No space** to hardwire $H(k_\delta, \chi_q \| 0) \oplus (\cdots), H(k_\delta, \chi_q \| 1) \oplus (\cdots)$ into sk_f for **all** $\chi_q \in \mathrm{X}(\mathsf{sk}_f)$.

Problem with Hardwiring into ct_χ

1. **separate functions for each level**
2. **hardwire in ct of adaptive FE**

$$\mathrm{ct}_\chi \leftarrow \mathrm{FE.Enc}\Big(\mathrm{mpk}, H(k_\delta, \chi) \oplus \big(\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1}\big)\Big)$$

$|\mathrm{ct}|$**-bit ciphertext**

$2|\mathrm{ct}|$**-bit plaintext**

chop $\left(\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1}\right)$ into blocks & hardwire one block at a time

$B = \Theta\left(\sqrt{|\mathrm{ct}|}\right)$ **blocks & each block is B bits**

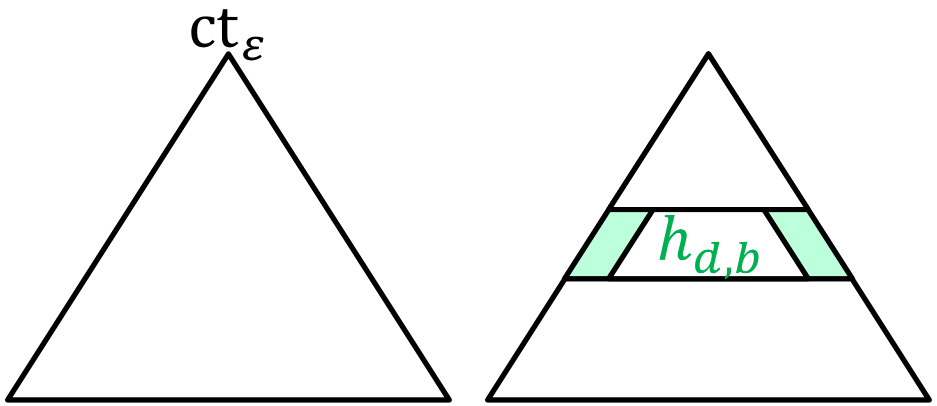
OK if $|\mathrm{ct}| \leq |\mathrm{plaintext}|^{2-\alpha}$

Ideal Obfuscation in Pr$\mathcal{O}$M

1. **separate functions for each level**
2. **hardwire in ct of adaptive FE**
3. **hardwire block by block**

$\mathrm{ct}_\chi\left(C, \chi, \sigma_{\chi,\leq\beta}, k_{\geq|\chi|,>\beta}\right)$ $\quad$ $\mathrm{sk}_f : G(\sigma_{\chi,\leq\beta}) \parallel \left(H(k_{|\chi|,>\beta}, \chi) \oplus (\cdots)\right)$

stitching first β blocks (sim.) and last $(B-\beta)$ blocks

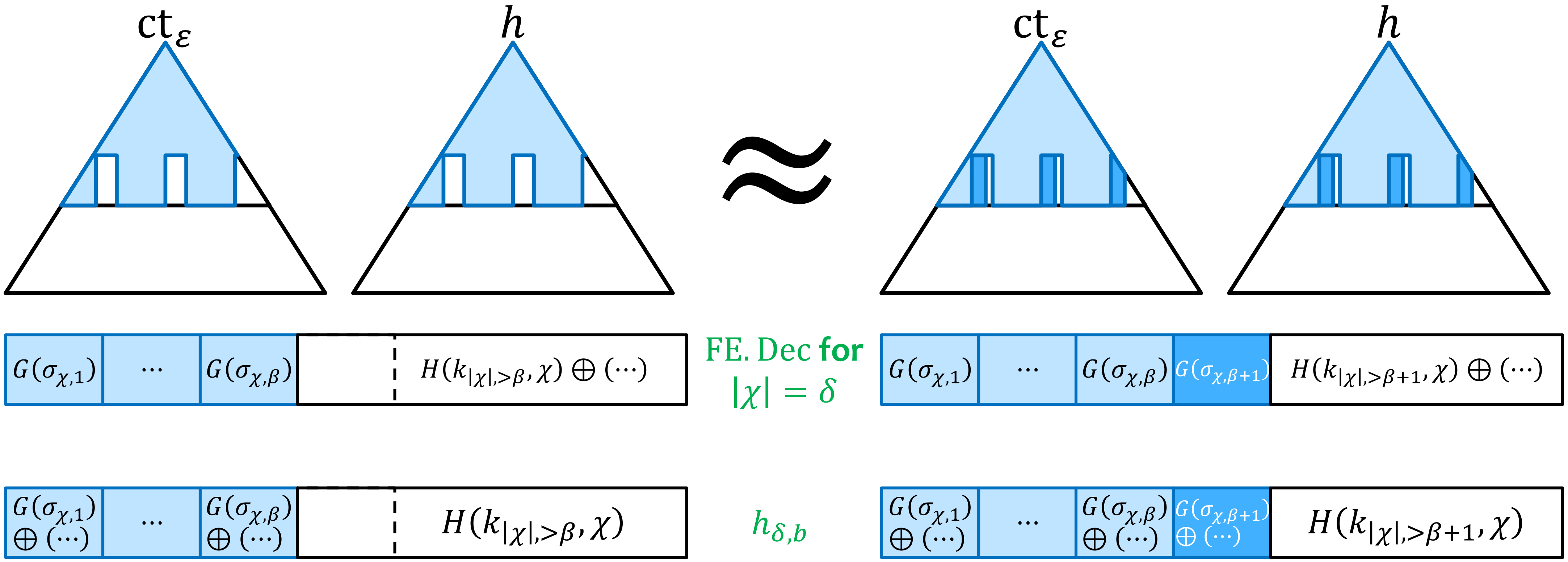


$$(\text{sim.})\ h_{d,b}(\chi) \leftarrow \begin{cases} \$, & |\chi| \neq d; \\ G(\sigma_{\chi,b}) \oplus (\cdots), & |\chi| = d. \end{cases}$$

Ideal Obfuscation: Hybrid

1. separate functions for each level
2. hardwire in ct of adaptive FE
3. hardwire block by block

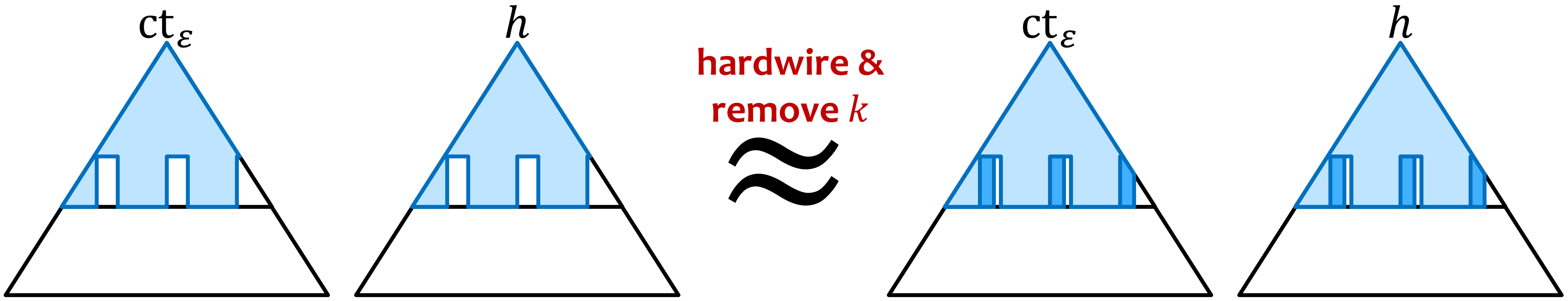
$$\mathrm{ct}_\chi\left(C, \chi, \sigma_{\chi,\le\beta}, k_{\ge|\chi|,>\beta}\right) \qquad \mathrm{sk}_f : G(\sigma_{\chi,\le\beta}) \parallel \left(H(k_{|\chi|,>\beta}, \chi) \oplus (\cdots)\right) \qquad h_{d,b} \begin{cases} H(k_{d,b}, \chi) \\ \$/G(\sigma_{\chi,b}) \oplus (\cdots) \end{cases}$$



Ideal Obfuscation: Proof

1. **separate functions for each level**
2. **hardwire in ct of adaptive FE**
3. **hardwire block by block**

$\mathrm{ct}_\chi(C, \chi, \sigma_{\chi,\le\beta}, k_{\ge|\chi|,>\beta})$ $\quad \mathrm{sk}_f: G(\sigma_{\chi,\le\beta}) \parallel (H(k_{|\chi|,>\beta}, \chi) \oplus (\cdots))$ $\quad h_{d,b} \begin{cases} H(k_{d,b}, \chi) \\ \$/G(\sigma_{\chi,b}) \oplus (\cdots) \end{cases}$



$\mathrm{ct}_{|\chi|<\delta}(\sigma_{\chi,\le B})$ $\quad h_{<\delta,b}: \$/\cdots$ $\quad \mathrm{ct}_{|\chi|<\delta}(\sigma_{\chi,\le B})$ $\quad h_{<\delta,b}: \$/\cdots$

$h_{\delta,<\beta}: \$/\cdots$ $\quad h_{\delta,<\beta}: \$/\cdots$

$\mathrm{ct}_{|\chi|=\delta}(\sigma_{\chi,\le\beta}, k_{\ge|\chi|,>\beta})$ $\quad h_{\delta,\beta}: H(k_\delta, \chi)$ $\quad \mathrm{ct}_{|\chi|=\delta}(\sigma_{\chi,\le\beta+1}, k_{\ge|\chi|,>\beta+1})$ $\quad h_{\delta,\beta}: \$/\cdots$

$h_{\delta,>\beta}: H(k_\delta, \chi)$ $\quad h_{\delta,>\beta}: H(k_\delta, \chi)$

$\mathrm{ct}_{|\chi|>\delta}(\sigma_{\chi,\le 0}, k_{\ge|\chi|,>0})$ $\quad h_{d>\delta,b}: H(k_{d,b}, \chi)$ $\quad \mathrm{ct}_{|\chi|>\delta}(\sigma_{\chi,\le 0}, k_{\ge|\chi|,>0})$ $\quad h_{d>\delta,b}: H(k_{d,b}, \chi)$

Pseudorandom Oracle Model

(new model for random-looking hash functions with code)

$\Rightarrow$

Ideal Obfuscation

FE

Thank you!

ia.cr/2022/1204

luoji@cs.washington.edu / luoji.bio