ideal obfuscation 理想混淆 与 pseudorandom oracle model 伪随机谕言模型

Aayush Jain ✉

Rachel Lin 林蕙佳 ✉

罗辑 ✉

Daniel Wichs ✉

Carnegie Mellon University

UNIVERSITY of WASHINGTON

Northeastern University

NTT Research

报告大纲

- 程序混淆　　讨论定义
- 散列函数　　标准模型、随机谕言模型
- 动机、问题、成果
- 伪随机谕言模型　定义、用法

- 理想混淆　　定义、工具、构造、安全证明

电路混淆 (circuit obfuscation)

C

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

$\mathrm{Obf}(\cdot)$

$\tilde{C}$

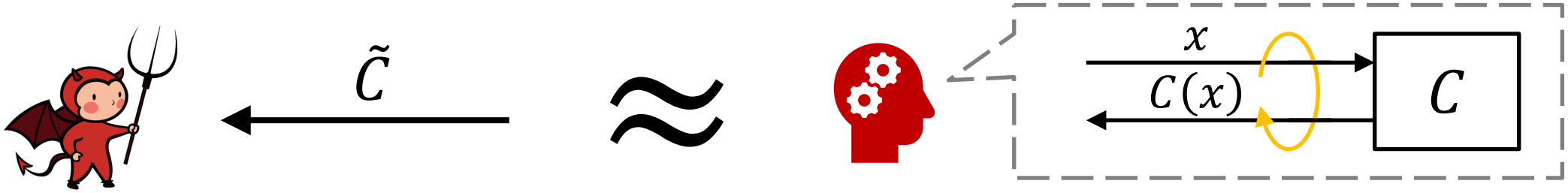
正确

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

安全

$\mathrm{Obf}(C)$ 是**不可理解**的代码

程序混淆的**模拟安全** (simulation security)



ideal obfuscation

理想混淆.

$$\exists S \quad \forall C:$$
$$\mathrm{Obf}(C) \approx S^{C}\left(1^{|C|}, 1^{|x|}\right).$$

unlearnable

✗ **对无法学习的电路不能实现**

virtual black-box

等效黑盒混淆.

$$\forall \text{一位输出的 } A \quad \exists S_A \quad \forall C:$$
$$A(\mathrm{Obf}(C)) \approx S_A^{C}\left(1^{|C|}, 1^{|x|}\right).$$

⚠ **VBB 不能一般地实现 [BGIRSVY]**
（不自然反例：输入自己的代码时表现异常）

✓ **安全刻画直观且安全性强**

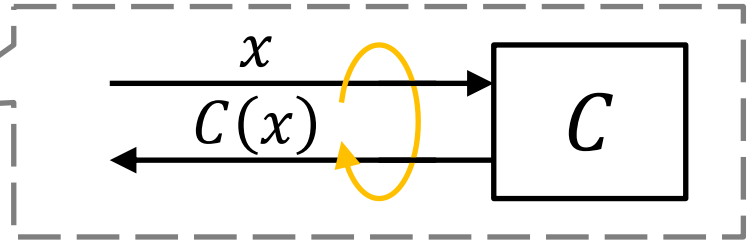
✓ **用法直观且简单**

从私钥加密得到公钥加密 [DH]

$$\mathrm{sk} = \mathrm{skeK}$$
$$\mathrm{pk} = \mathrm{Obf}(\mathrm{skeEnc}(\mathrm{skeK}, \cdot))$$

程序混淆的**模拟安全**（续）

$\tilde{C}$ ⟵ ≈



ideal obfuscation

理想混淆.

$\exists S \quad \forall C:$

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

unlearnable

✗ **对无法学习的电路不能实现**

virtual black-box

等效黑盒混淆.

$\forall$ **一位**输出的 $A \quad \exists S_A \quad \forall C:$

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

⚠ **VBB 不能一般地实现 [BGIRSVY]**

（不自然反例：输入自己的代码时表现异常）

✓ **安全刻画直观且安全性强**

✓ **用法直观且简单**

✓ **目前某些（本身可能的）应用的必由之路**

plausible

DE-PIR [BIPW] 适用于 RAM 的 FHE [HHWW]

等效灰盒混淆 [BC] 解密时间最优的泛函加密 [ACFQ]

从二值擦除信道得到 OT [AIKNPPR]

公开随机数的 $di\mathcal{O}$、输入长度无界的 Turing 机程序混淆 [IPS]

输出几乎全为零函数的输入隐藏混淆 [BBCKPS]

可提取的证据加密 [GKPVZ] 窃听信道编码 [IKLS]

证否 XOR 引理梦中情理版本的反例 [BIKSW]

不可区分安全的程序混淆 (indistinguishability obfuscation)

$$|C_0| = |C_1| \text{ 且 } \forall x: C_0(x) = C_1(x) \quad \Longrightarrow \quad \mathrm{Obf}(C_0) \approx \mathrm{Obf}(C_1)$$

✅ **已经相当强力了**

抵赖加密、短签名、陷门单射、非交互式零知识证明、安全选择 [SW]
全同态加密 [ABFGGTW]　　多项式规模电路的泛函加密 [GGHRSW]
两轮多方安全计算 [GGHR]　随机访问机的短乱码化 [CH]　　加其他更更多！

best-possible
✅ **最佳混淆 [GR]**

$i\mathcal{O}(\mathrm{padded}(C)) \approx i\mathcal{O}(\mathrm{Obf}(C))$
任何 Obf 所隐藏的，左边也隐藏
即 $i\mathcal{O}$ **不**比任何 Obf 更**不**安全

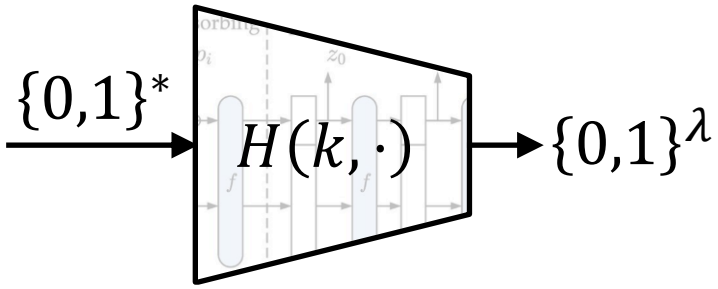
✅ **可基于久经考验的假设构造 [JLS]**

⚠️ **安全刻画不直观、安全性弱**

⚠️ **使用时需搭配繁复的技巧**

😃 **尚有太多可期**

散列函数 (hash functions)



基础而有用的密码学工具

- 单向
- 抗第二原像
- 抗碰撞

$$\mathrm{Sign}(\mathrm{sk}, m) \stackrel{\text{def}}{=} \mathrm{Sign}_\lambda\big(\mathrm{sk}_\lambda, H(K, m)\big)$$

✅ 不可伪造 ⟸ 不可伪造 + 抗碰撞

$$\mathrm{Sign}(\mathrm{sk}, m) \stackrel{\text{def}}{=} \mathrm{TDP}^{-1}\big(\mathrm{sk}, H(K, m)\big)$$

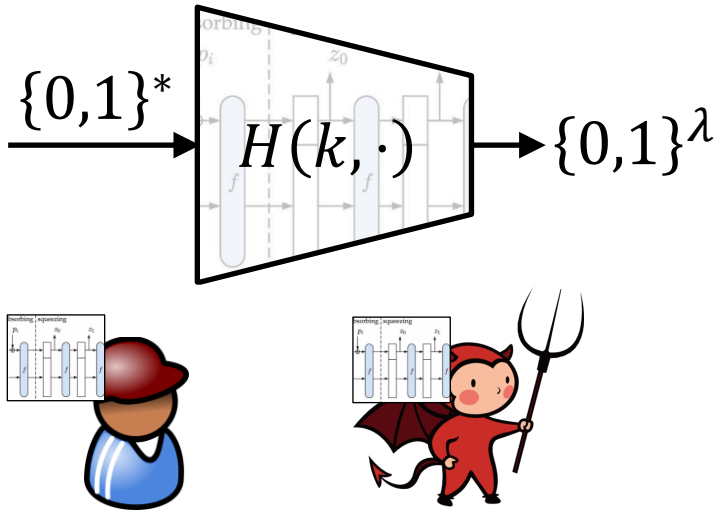
🧐 归约到 H 的什么复杂度假设?

对散列函数“安全性”感性的认知

不总是可以通过简单的复杂度假设刻画

散列函数：理想模型

标准模型

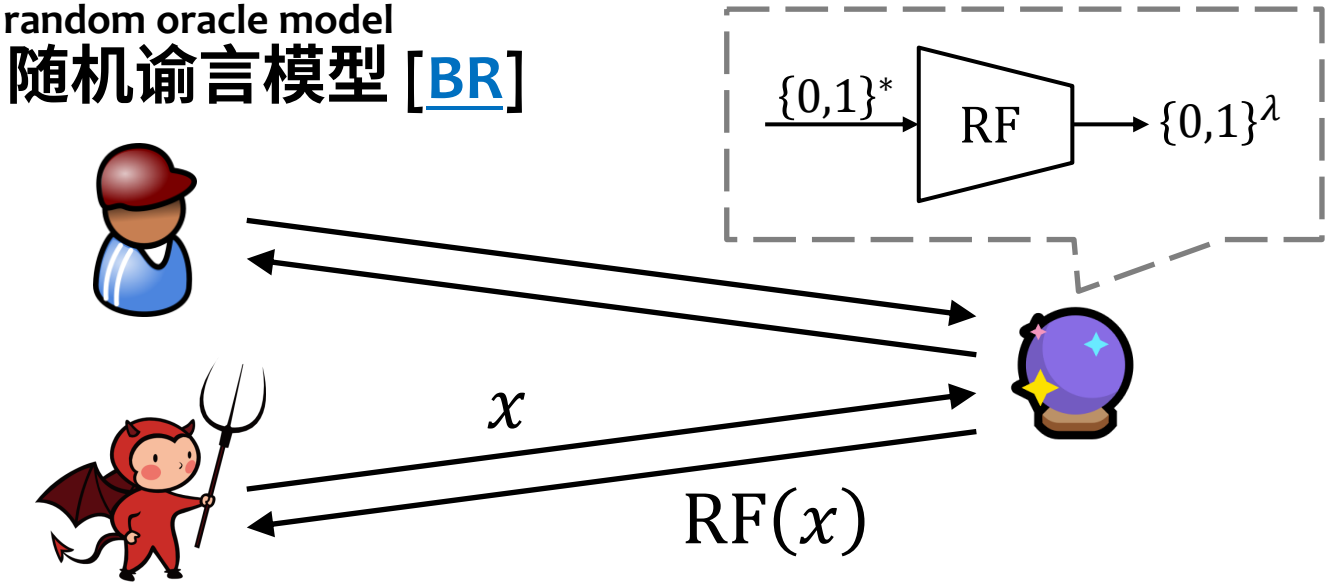


- 单向
- 抗第二原像
- 抗碰撞

correlation intractability

- 关系难解性

random oracle model
随机谕言模型 [BR]



- ✅ **实用**、**高效**的方案
  Schnorr 签名[Schnorr] RSA-OAEP[BR] TLS[KPW,DFGS,DJ]
- ⚠️ **（不自然）反例**（模型中安全，但用任意散列函数都不安全）
- ✅ ROM 里的安全证明**好过**没有任何证明
  🚩 在 ROM 里**写不出**证明是**大问题**
- ✅ 标准模型版本**之母**

动机与问题

? 用散列函数的代码有用吗?

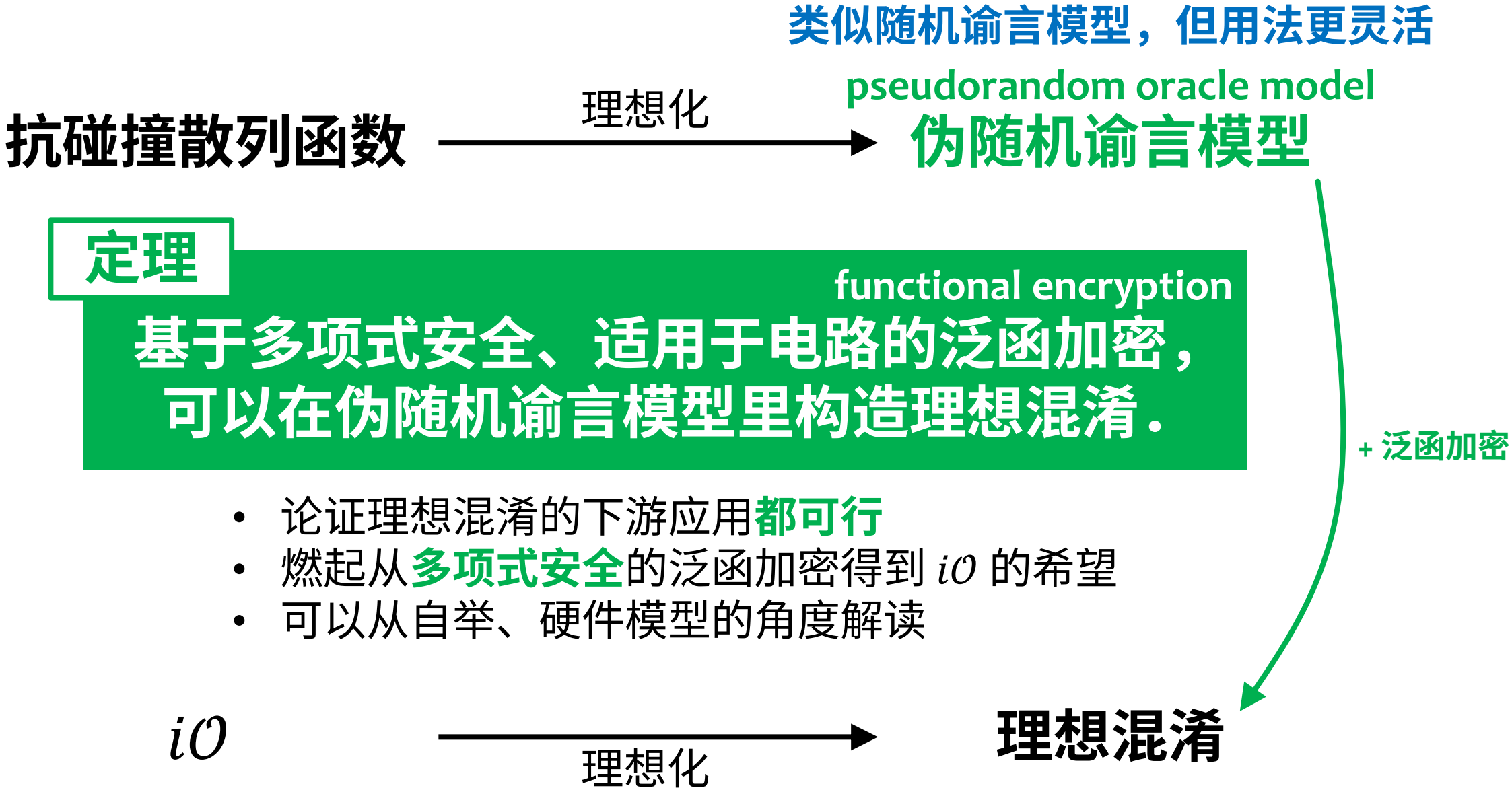
抗碰撞散列函数 —理想化→ **随机谕言模型**

$\mathrm{RO}(\cdot) = \mathrm{Obf}(\mathrm{PRF}(k, \cdot))$

黑盒使用
散列函数
无助于构造
理想混淆[CKP]

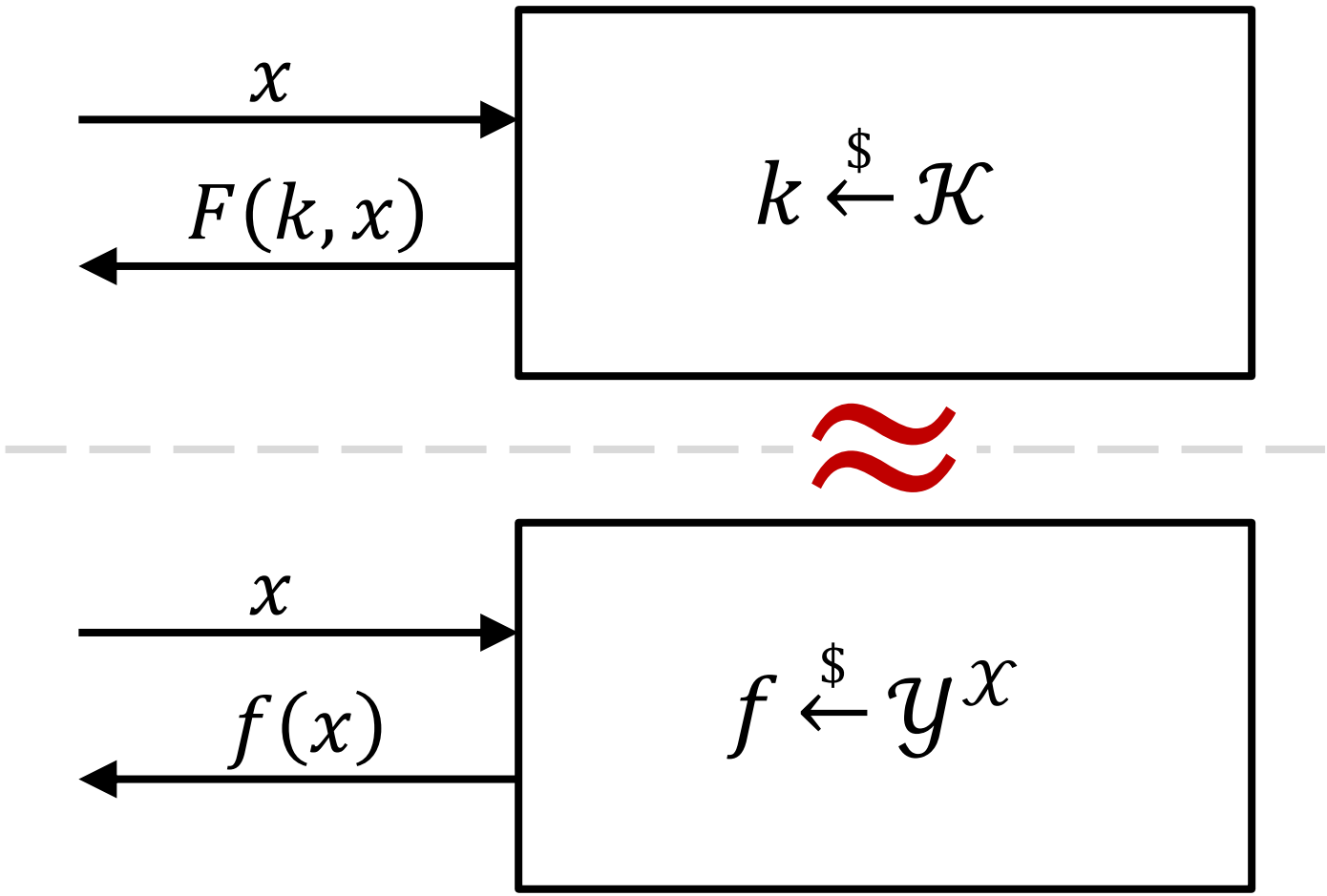
$i\mathcal{O}$ —理想化→ **理想混淆**

本作成果


类似随机谕言模型，但用法更灵活
pseudorandom oracle model
抗碰撞散列函数
理想化
伪随机谕言模型
定理
functional encryption
基于多项式安全、适用于电路的泛函加密，
可以在伪随机谕言模型里构造理想混淆。
+ 泛函加密
• 论证理想混淆的下游应用都可行
• 燃起从多项式安全的泛函加密得到 iO 的希望
• 可以从自举、硬件模型的角度解读
iO
理想化
理想混淆

伪随机函数 (pseudorandom function)

函数族 $F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$



伪随机谕言模型 (pseudorandom oracle model, $\mathrm{Pr}\mathcal{O}\mathrm{M}$)

模型的两面.

- H 像**随机函数**（h 对应 ROM）
- H 有（短）**代码实现**（k 对应通常的 PRF）

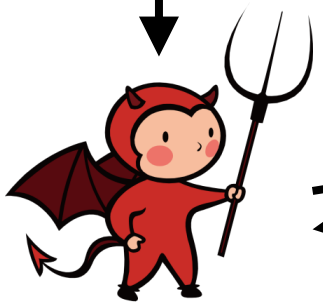
伪随机函数 $H\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ 的 $\mathrm{Pr}\mathcal{O}\mathrm{M}$

获取 $k \in \mathcal{K}$ 的柄把

$h = \mathrm{hMap}(k)$

h

以 h 对应的函数作用于 x



$h(x) = H(\mathrm{hMap}^{-1}(h), x)$

随机置换（柄把映射）

$\mathrm{hMap}\colon \mathcal{K} \to \mathcal{H}_{(\mathcal{H}=\mathcal{K})}$

使用 $\mathrm{Pr}\mathcal{O}\mathrm{M}$ 中 H 的两种方式

- 用 h—通过谕言查询 H　　**只用 h 的话就是 ROM**
- 用 k—利用 H 的代码　　**只用 k 的话就是 PRF**

$\mathrm{Pr}\mathcal{O}\mathrm{M}$ 灵活之处在于可同时以两种方式使用 H

adversary

但如果 k 让使坏者“知道”，

则 h 只能对应普通 PRF 实例，**不是**随机谕言！

使用 Pr$\mathcal{O}$M 的基本套路

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

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

⚠ h 的谕言查询不可监控、不可编程

$\approx$

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

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

（FHE/GC/FE 安全性）

$\approx$

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

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

（H 的 PRF 安全性）

✓ h 的谕言查询可以监控、可以编程

$\mathrm{Pr}\mathcal{O}\mathrm{M}$ 中 H 代码的使用限制

- 用 h—通过谕言查询 H　　**只用 h 的话就是 ROM**
- 用 k—利用 H 的代码　　**只用 k 的话就是 PRF**

$\mathrm{Pr}\mathcal{O}\mathrm{M}$ 灵活之处在于可同时以两种方式使用 H

adversary

但如果 k 让使坏者“知道”，

则 h 只能对应普通 PRF 实例，不是随机谕言！

先过渡到删除 k，再把 h 变成随机谕言！

使用代码的限制：必须与纯黑盒使用 h 的过渡博弈不可区分

随机谕言范式 (paradigm) [BR]

1. 在 ROM 中形式化定义 Π.
2. 在 ROM 中设计 Π 的实现 $P^{\mathcal{O}}$.
3. 在 ROM 中证明 $P^{\mathcal{O}}$ 满足 Π 的定义.
4. 把 $\mathcal{O}$ 替换为真实的散列函数，作为 Π 在现实世界的实现.

✅ 把散列函数理想化建模为公开随机函数

✅ 排除泛用攻击

✅ 论证**实用高效**方案的安全性

⚠️ 禁止使用散列函数的代码

ROM 中不能实现　模拟安全的 FE [AKW]　VBB 混淆 [CKP]

强行使用 RO 代码　证明的递归复合 [BCMS]

⚠️ （不自然）反例

伪随机谕言范式

1. 在 $\mathrm{Pr}\mathcal{O}\mathrm{M}$ 中形式化定义 Π.
2. 在 $\mathrm{Pr}\mathcal{O}\mathrm{M}$ 中设计 Π 的实现 $P^{\mathcal{O}}$.
3. 在 $\mathrm{Pr}\mathcal{O}\mathrm{M}$ 中证明 $P^{\mathcal{O}}$ 满足 Π 的定义.
4. 把 $\mathcal{O}$ 替换为真实的散列函数，作为 Π 在现实世界的实现.

- ✅ 把散列函数理想化建模为**有代码实现**的公开随机函数
- ✅ 排除泛用攻击
- ✅ 论证**此前未能实现的目标**的可能性（本作：**理想混淆**）
- ✅ **允许**使用散列函数的代码，有一定限制

- ⚠️ （不自然）反例

实现 $\mathrm{Pr}\mathcal{O}\mathrm{M}$

经验法则. 适合实现 ROM $\Longrightarrow$ 适合实现 $\mathrm{Pr}\mathcal{O}\mathrm{M}$

例. 令 $h = k$ 为随机串且 $H(k, x) = \mathrm{SHA3}(k \parallel x)$

理由.

- 优秀的散列函数是“**自混淆 PRF**”. (self-obfuscated)
- 实现 $\mathrm{Pr}\mathcal{O}\mathrm{M}$ 的要求**不比实现 ROM 的要求高**

——都不过是把谕言查询替换为代码.

休息，
休息一会儿！

画面上缺少两个形象，是什么？
答案：一休、晴天娃娃.

报告大纲（回顾与补充）

- 程序混淆　　讨论定义
- 散列函数　　标准模型、随机谕言模型
- 动机、问题、成果
- 伪随机谕言模型　定义、用法

- 理想混淆
  - 定义
  - 工具（泛函加密）
  - 构造
  - 安全证明中的技巧亮点

理想混淆：定义

$\mathcal{O}$ 是理想化模型里的谕言，如 $\mathrm{Pr}\mathcal{O}^H$

$\mathrm{Obf}^{\mathcal{O}}(C) \to \tilde{C}^{\bullet}$ 是混淆算法

混淆前的 C 是**无**谕言程序

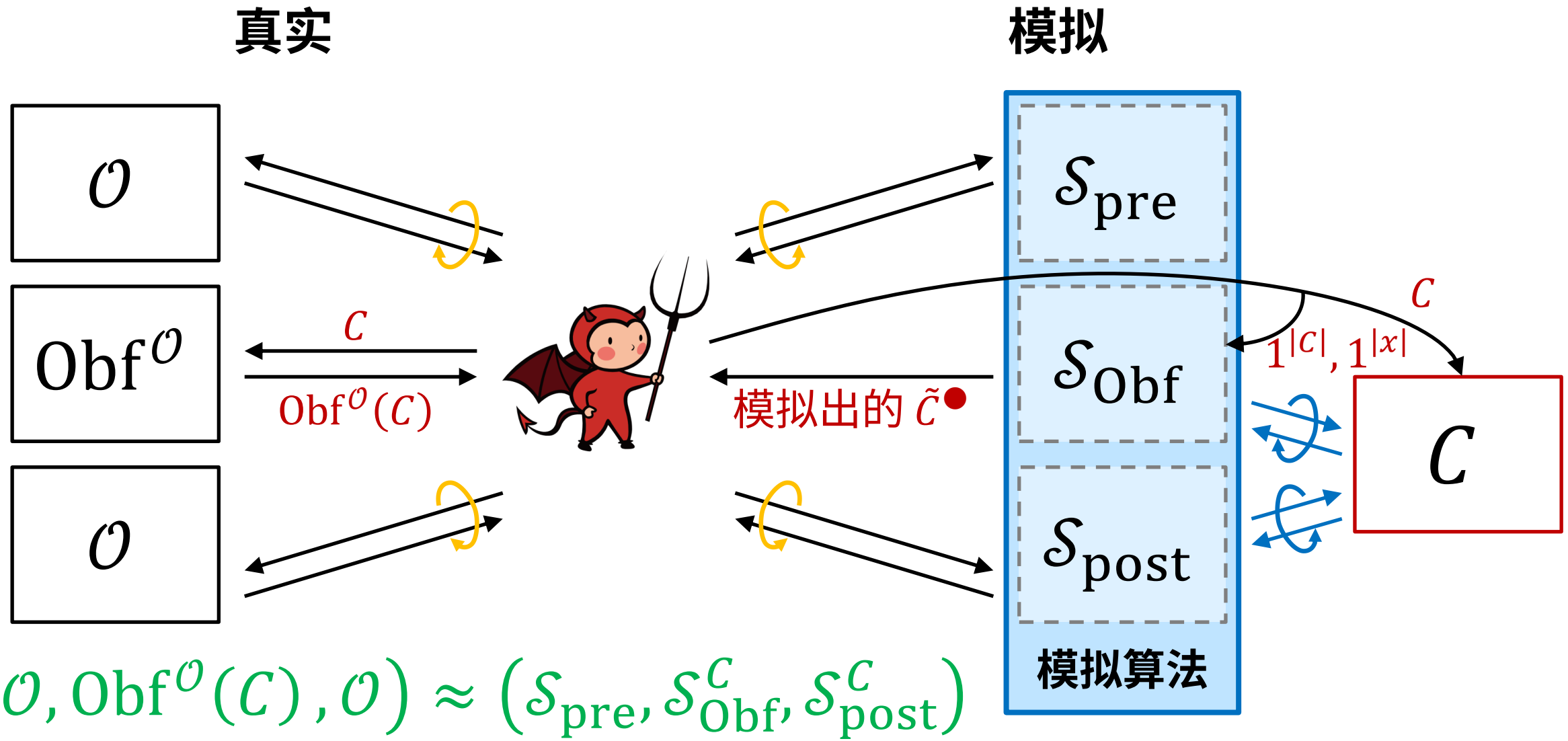
混淆后的 $\tilde{C}^{\bullet}$ 可以访问谕言

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

indifferentiability

安全定义受**无差别替代性**启发 [MRH]

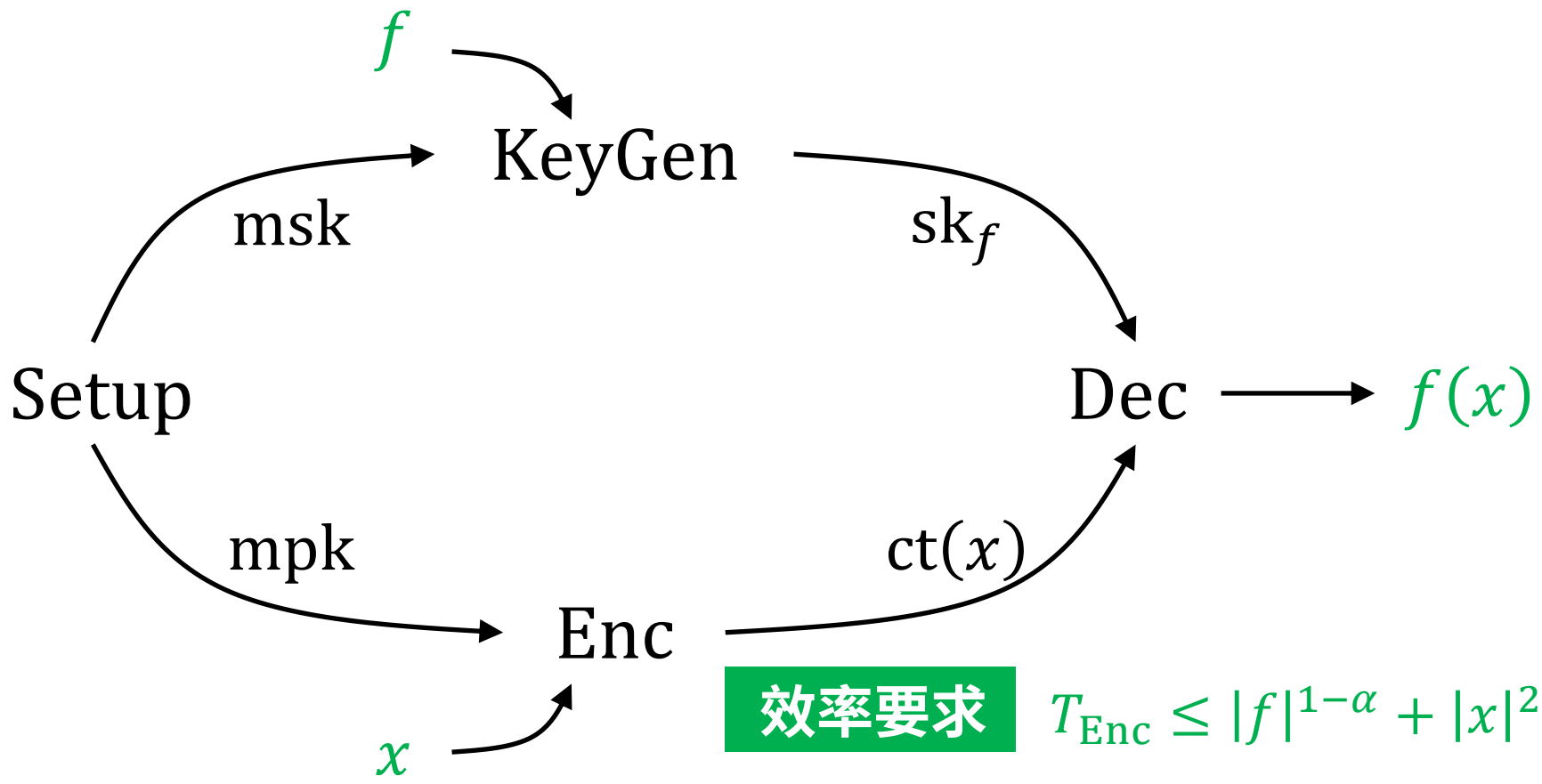
理想混淆：安全性



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

functional encryption

工具：（标准模型下的）泛函加密



效率要求 $T_{\mathrm{Enc}} \leq |f|^{1-\alpha} + |x|^{2-\alpha}$

适应性安全 x_0, x_1 可根据 $\mathrm{mpk}, \mathrm{sk}_f$ 选择

安全性. 若 $f(x_0) = f(x_1)$ 则 $\left(\mathrm{mpk}, \mathrm{sk}_f, \mathrm{ct}(x_0)\right) \approx \left(\mathrm{mpk}, \mathrm{sk}_f, \mathrm{ct}(x_1)\right)$

主要定理

基于多项式安全、**满足效率要求**、适用于所有电路的泛函加密，
在**任意 PRF** H 的伪随机谕言模型下，
可构造**适用于所有电路**的理想混淆.

function-revealing encryption
实际上**函数揭示加密**就足够了（Setup 时选定 f 且无 KeyGen）.

已知. 这里所需要的 FE/FRE 可以从多项式安全、选择性安全、单密钥、次线性效率的公钥 FE 得到，进而可以从久经考验的假设得到. [ABSV, GVW, GS, LM, AJS, BV, AS, KNTY, JLL, JLS]

主要定理：解读

基于多项式安全、**满足效率要求**、适用于所有电路的泛函加密，
在**任意 PRF** H 的伪随机谕言模型下，
可构造**适用于所有电路**的理想混淆.

首选解读.

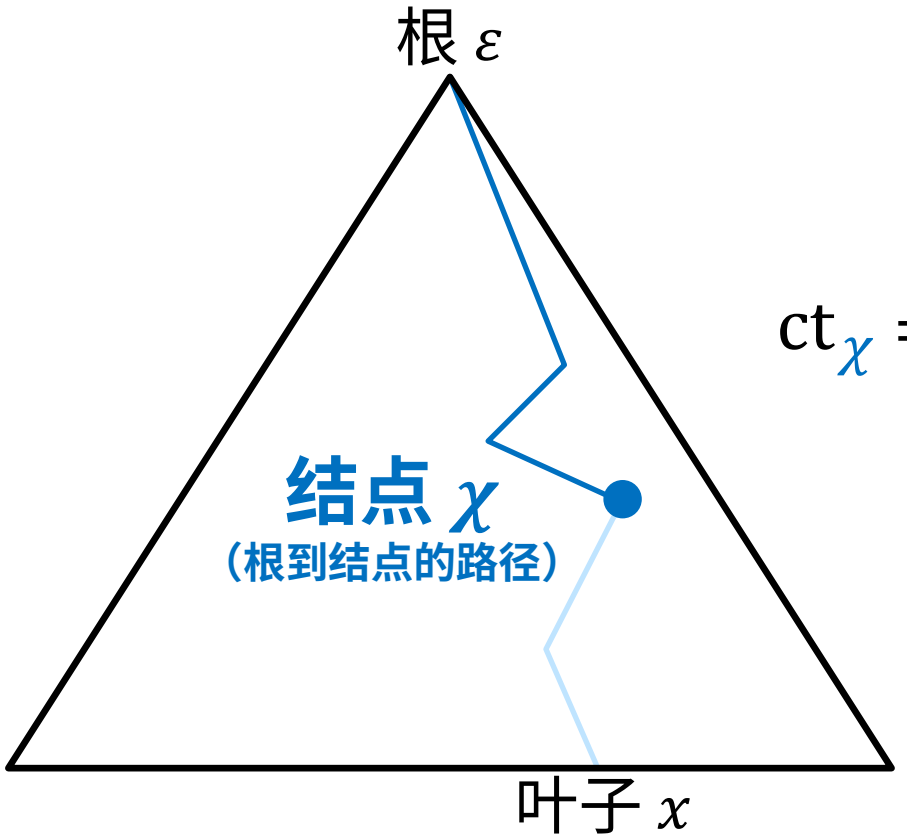
理想散列函数“感性存在且可被 Pr$\mathcal{O}$M 刻画”$\Longrightarrow$ 理想混淆“感性存在”

另解.

- 从 PRF 理想混淆（Pr$\mathcal{O}$M 所刻画的理想散列函数）
  自举得到到所有电路的理想混淆
- 简单的**硬件模型**（通过令牌黑盒访问 PRF）可实现 Pr$\mathcal{O}$M，
  也可实现所有电路的理想混淆

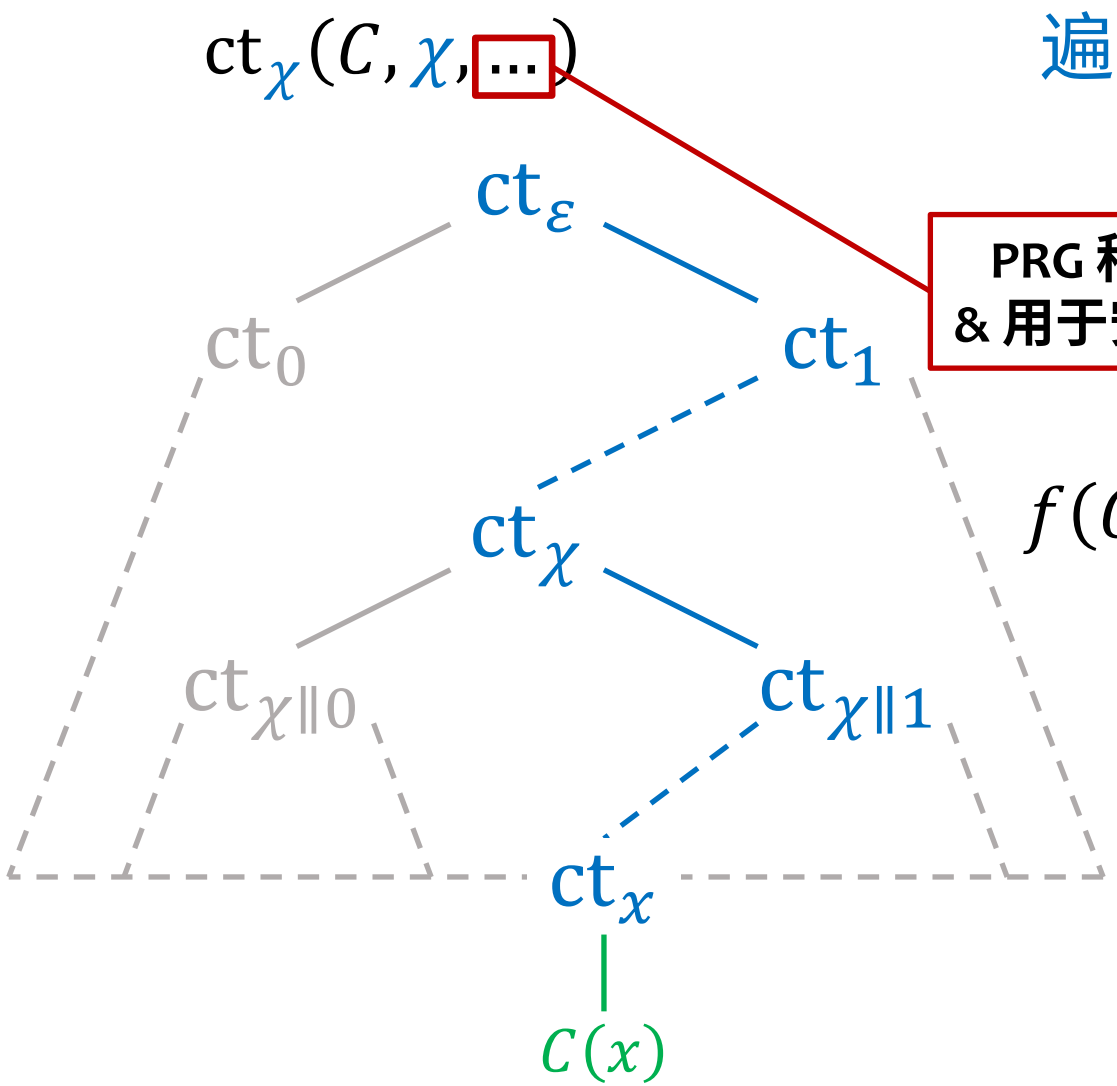
FE 变 $i\mathcal{O}$：二叉树构造

完美二叉树（叶子 ⇔ 输入值）



$\mathrm{ct}_\chi = (C, \chi, \dots)$ 的 FE 密文，其中 $\chi \in \{0,1\}^{\leq D}$

FE 变 $i\mathcal{O}$：遍历、求值



遍历/求值 = 用 FE 密钥 sk_f 解密

PRG 种子 / PRF 密钥
& 用于安全证明的信息

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

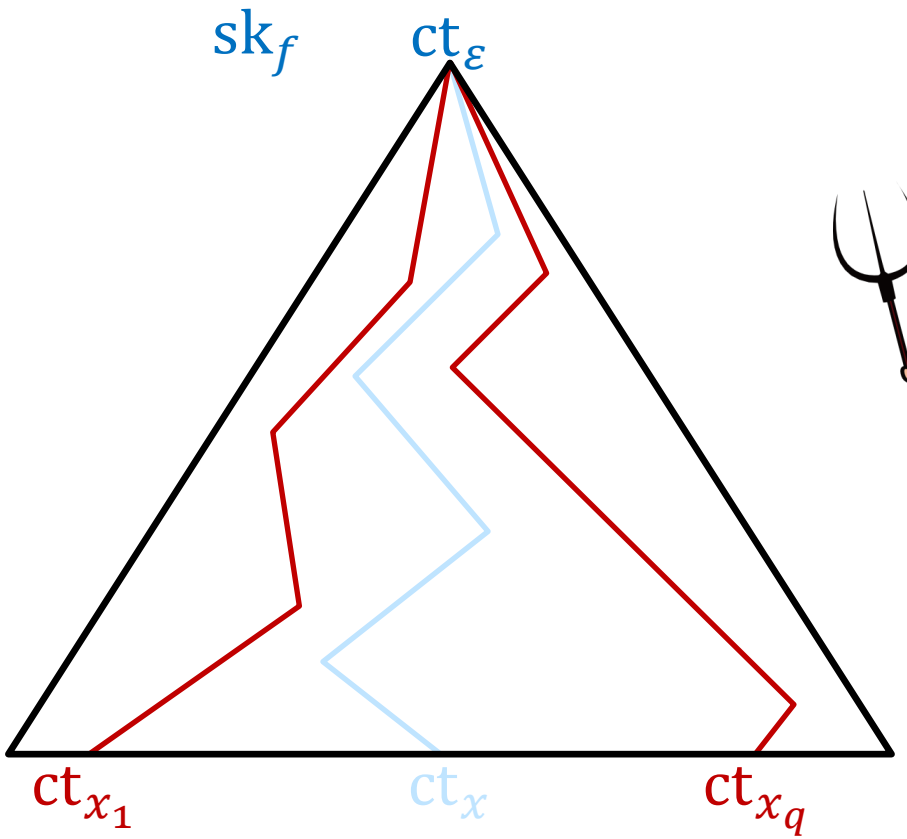
FE. Enc 要用到随机数

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

FE 变 $i\mathcal{O}$

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

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



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

← 可能多次运行混淆后的程序.
不知道它“对**哪些**路径感兴趣”.
归约算法必须考虑**每一个**输入.
(产生**指数归约损失**)

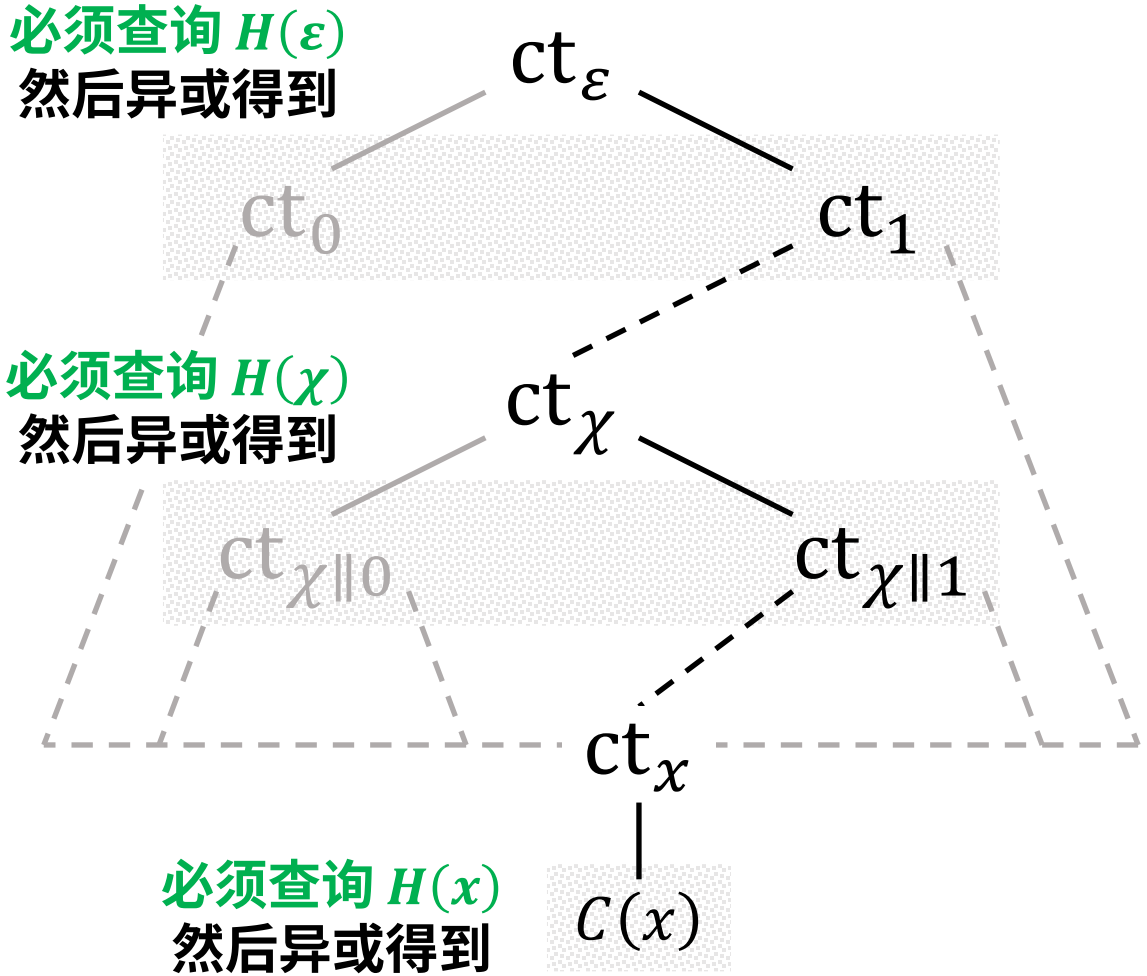
理想化模型或许有用?

在 ROM 里的初步想法

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

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

FE.KeyGen 派生密钥的电路不支持 ROM 查询



如果**不查询** $H(\chi)$
则 $\mathrm{FE.Dec}(\mathrm{sk}_f,\mathrm{ct}_\chi)$ **是纯随机**

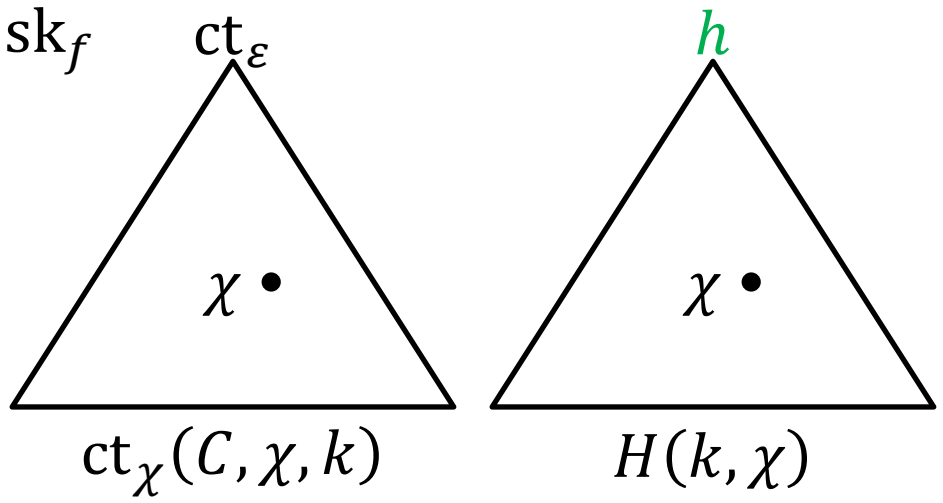
模拟算法
监视 ROM 查询并对其**编程**

PrOM 中的初步尝试

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

利用 k 以及 H 的代码

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



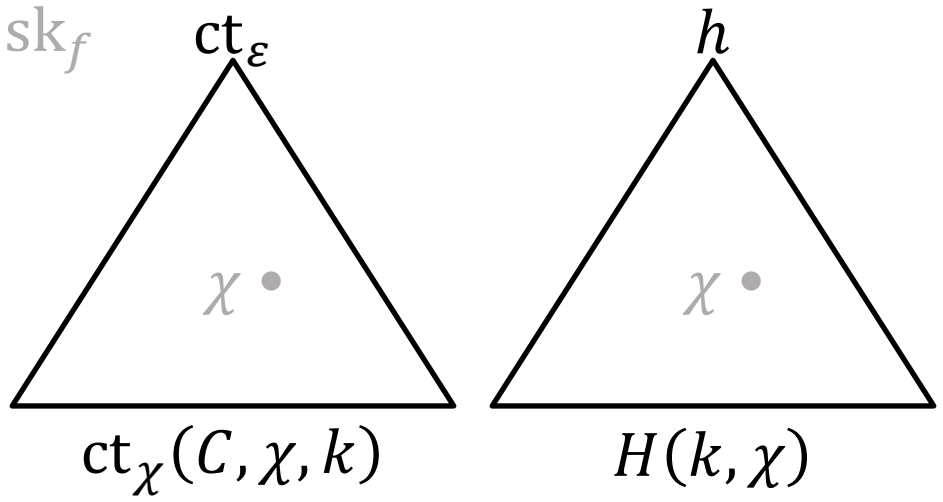
$$\mathrm{Dec}(\mathrm{sk}_f, \mathrm{ct}_\chi) \oplus h(\chi) = (\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1})/C(\chi)$$

利用 h 查询谕言

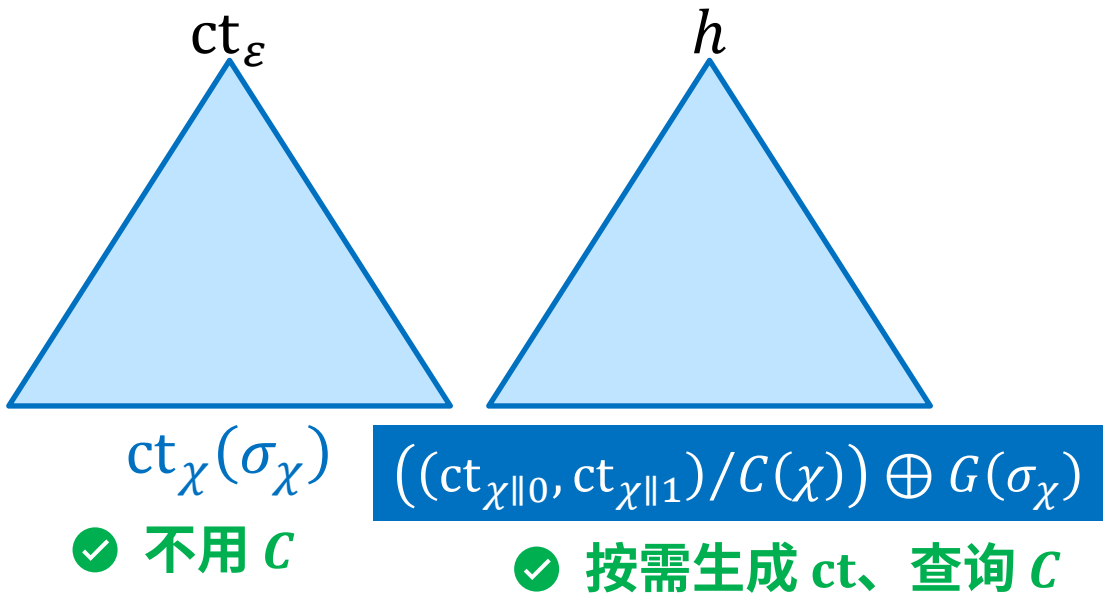
初步尝试：模拟算法

$\mathrm{ct}_\chi \begin{cases} C, \chi, k \\ \boxed{\sigma_\chi} \end{cases}$ 每个 χ 都有独立的种子

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



模拟算法



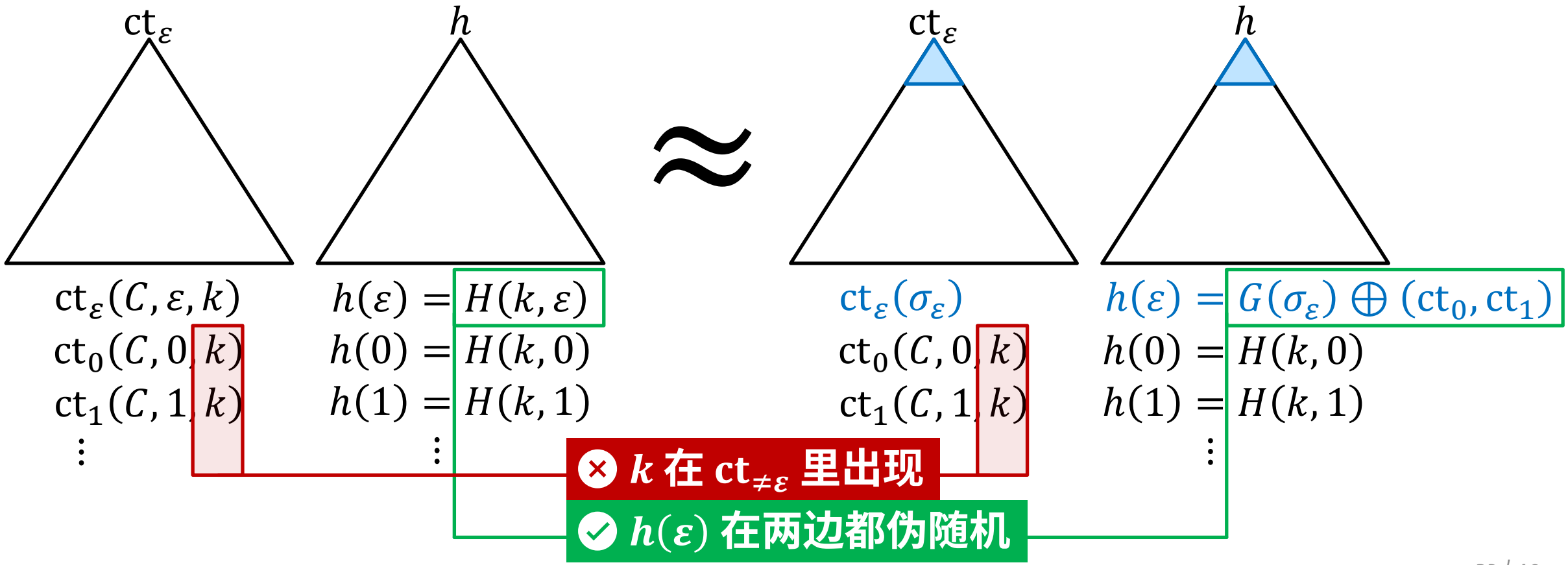
- 不用 C
- 按需生成 ct、查询 C

- 两边的 { FE. Dec 结果, Pr$\mathcal{O}$M 查询回应 } **不可区分**：

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

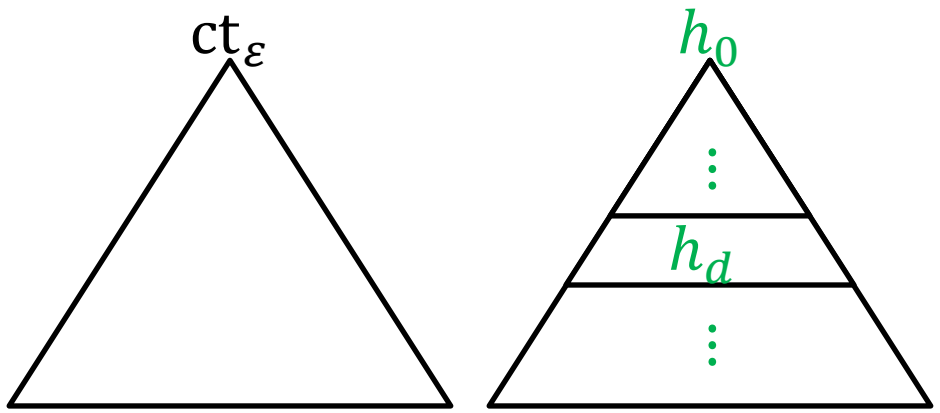
初步尝试：证明

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



再试一次

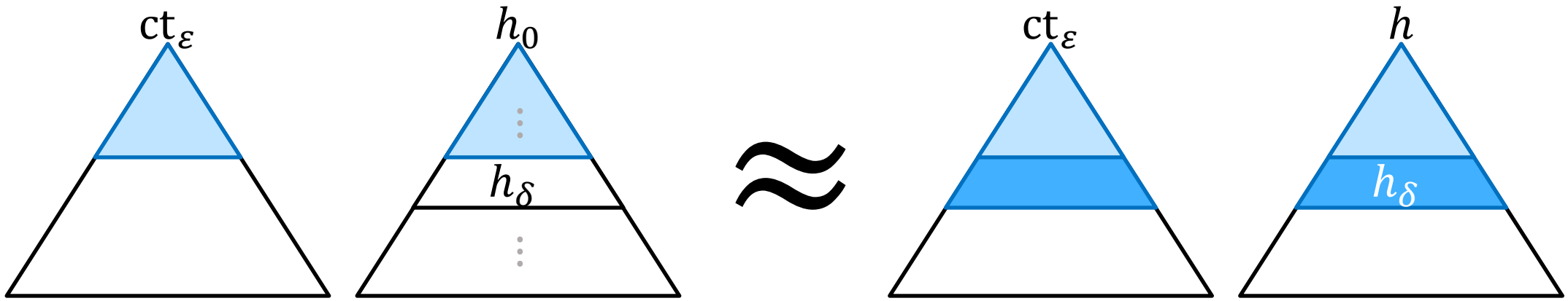
$$\mathrm{ct}_\chi \begin{cases} C, \chi, k_{\geq|\chi|} \\ \sigma_\chi \end{cases} \qquad \mathrm{sk}_f \begin{cases} H(k_{|\chi|}, \chi) \oplus (\cdots) \\ G(\sigma_\chi) \end{cases}$$



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

再试一次：证明

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



$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$
$\mathrm{ct}_{|\chi|=\delta}(C, \dots)$
$\mathrm{ct}_{|\chi|>\delta}(C, \dots)$

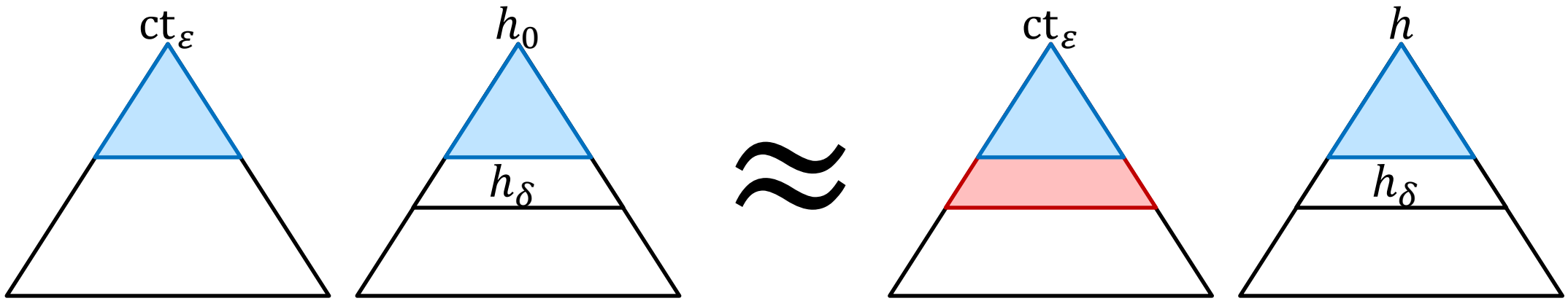
$h_{<\delta}: \$/\cdots$
$h_\delta: H(k_\delta, \chi)$
$h_{d>\delta}: H(k_d, \chi)$

$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$
$\mathrm{ct}_{|\chi|=\delta}(\sigma_\chi)$
$\mathrm{ct}_{|\chi|>\delta}(C, \dots)$

$h_{<\delta}: \$/\cdots$
$h_\delta: \$/\cdots$
$h_{d>\delta}: H(k_d, \chi)$

再试一次：“挪移大法”

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



$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$
$\mathrm{ct}_{|\chi|=\delta}(C, \dots)$
$\mathrm{ct}_{|\chi|>\delta}(C, \dots)$

$h_{<\delta}: \$/\cdots$
$h_\delta: H(k_\delta, \chi)$
$h_{d>\delta}: H(k_d, \chi)$

$\mathrm{ct}_{|\chi|<\delta}(\sigma_\chi)$
$\mathrm{ct}_{|\chi|=\delta} \mapsto H(k_\delta, \chi) \oplus (\cdots)$
$\mathrm{ct}_{|\chi|>\delta}(C, \dots)$

$h_{<\delta}: \$/\cdots$
$h_\delta: H(k_\delta, \chi)$
$h_{d>\delta}: H(k_d, \chi)$

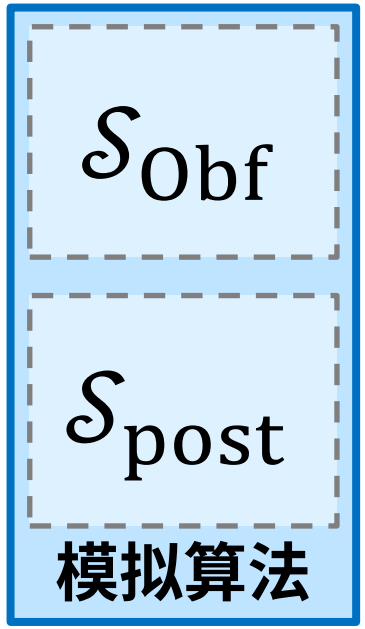
把 $H(k_\delta, \chi) \oplus (\cdots)$ 挪进 sk_f?

没法把解密结果挪进 sk_f

C（此时 $\mathcal{S}_{\mathrm{Obf}}$ 只知道 $1^{|C|}, 1^{|x|}$）

$\mathrm{sk}_f, \mathrm{ct}_\varepsilon, h_0, \ldots, h_D$

查询 $\chi_q \in \mathrm{X}(\mathrm{sk}_f)$ 的 $h_{\delta-1}(\chi_q)$



查询 $h_{\delta-1}(\chi_q) = G(\sigma_{\chi_q}) \oplus (\mathrm{ct}_{\chi_q\|0}, \mathrm{ct}_{\chi_q\|1})$
迫使 $\mathcal{S}_{\mathrm{post}}$ 产生 $\mathrm{ct}_{\chi_q\|0}, \mathrm{ct}_{\chi_q\|1}$

把 $H(k_\delta, \chi) \oplus (\cdots)$ 挪入 $\mathrm{ct}_{|\chi|=\delta}$

1. **无法预测**诸 χ_q，因为它们**适应性**依赖于 sk_f.
2. sk_f 里**没有足够的空间**（同时）存放**所有** $\chi_q \in \mathrm{X}(\mathrm{sk}_f)$ 所对应的 $H(k_\delta, \chi_q \| 0) \oplus (\cdots), H(k_\delta, \chi_q \| 1) \oplus (\cdots)$.

把解密结果挪进 ct_χ 的难点

1. 分离各层所用的散列函数实例
2. 信息挪入适应性安全 FE 的 ct

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

密文长度只有 $|\mathrm{ct}|$

明文长度是 $2|\mathrm{ct}|$

把 $\left(\mathrm{ct}_{\chi\|0}, \mathrm{ct}_{\chi\|1}\right)$ 分块
每次只挪一块

分成 $B = \Theta\left(\sqrt{|\mathrm{ct}|}\right)$ 块
每块长度是 B

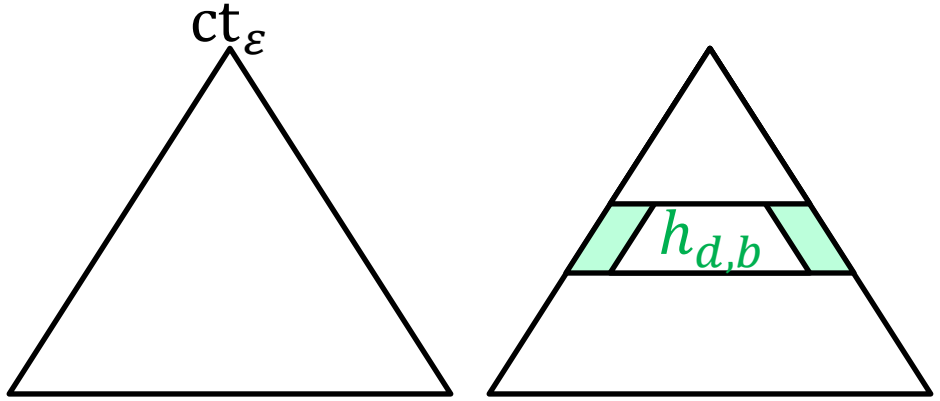
若 $|\text{密文}| \leq |\text{明文}|^{2-\alpha}$ **则可行**

PrOM 中的理想混淆

1. 分离各层所用的散列函数实例
2. 信息挪入适应性安全 FE 的 ct
3. 分块挪动信息

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

拼接前 β 块（模拟）和后 $(B-\beta)$ 块

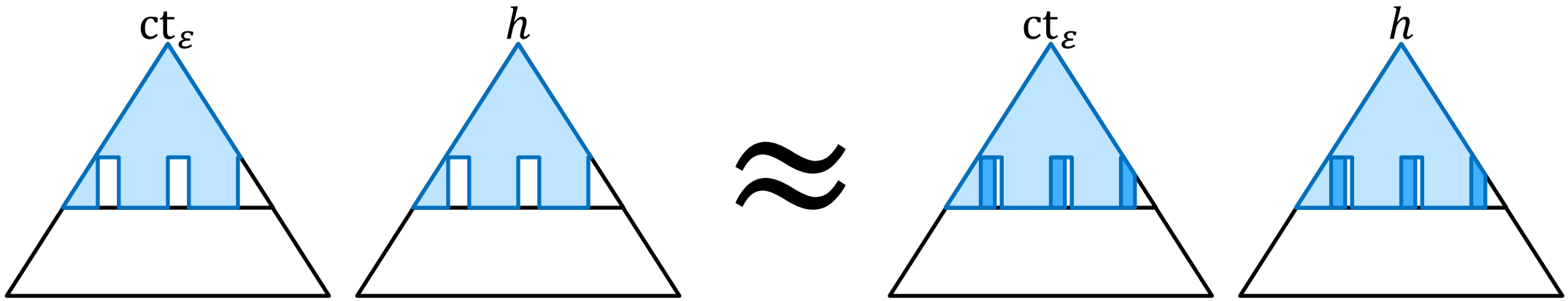


（模拟） $h_{d,b}(\chi) \leftarrow \begin{cases} \$, & |\chi| \neq d; \\ G(\sigma_{\chi,b}) \oplus (\cdots), & |\chi| = d. \end{cases}$

理想混淆：过渡博弈

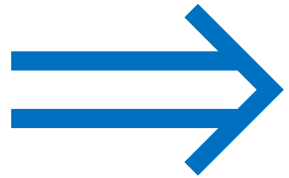
1. 分离各层所用的散列函数实例
2. 信息挪入适应性安全 FE 的 ct
3. 分块挪动信息

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



伪随机谕言模型

（散列函数新模型：理想化 + 代码）



理想混淆

泛函加密

谢谢！

ia.cr/2022/1204 / 哔哩哔哩 BV1sV4y1r7Bc

luoji@cs.washington.edu / luoji.bio