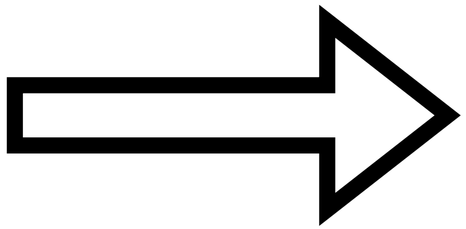
GCWise

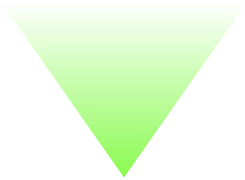
Garbled Circuits with *Sublinear Evaluator*

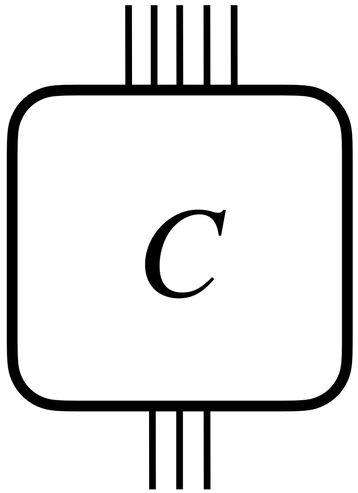
Abida Haque	NC State
David Heath	Georgia Tech ⇒ UIUC
Vladimir Kolesnikov	Georgia Tech
Steve Lu	Stealth Software Technologies, Inc.
Rafail Ostrovsky	UCLA
Akash Shah	UCLA

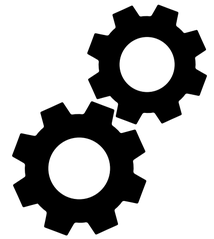
$\hat{C}$

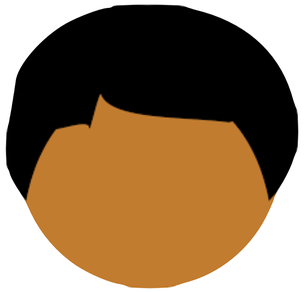


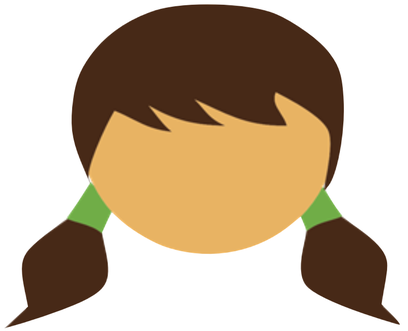
$\hat{C}$





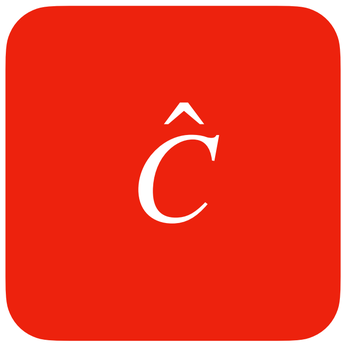


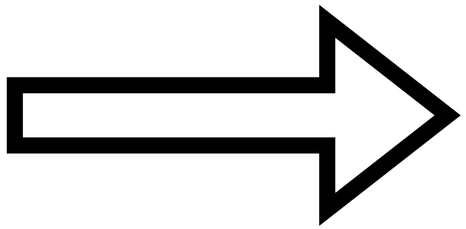


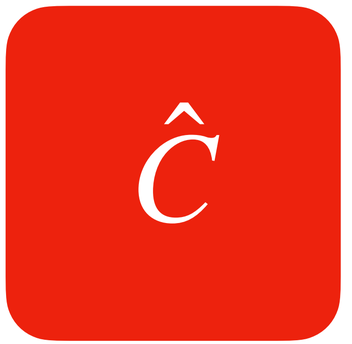


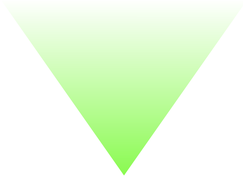
Generator

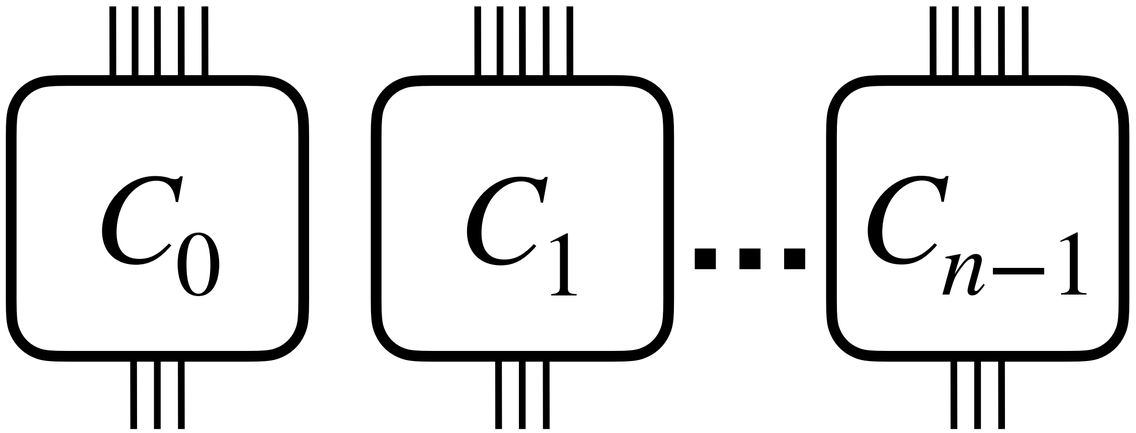
Evaluator

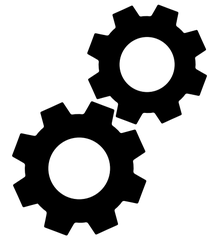


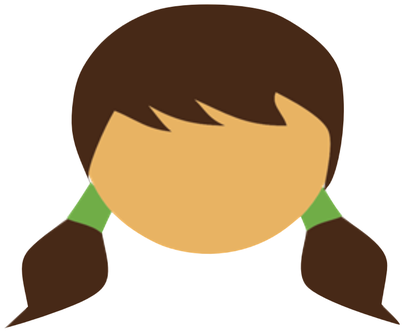








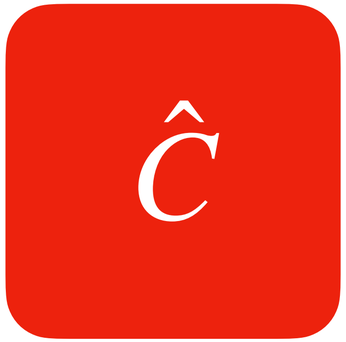


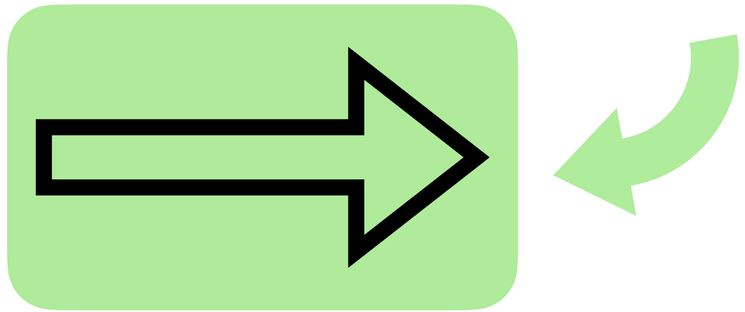


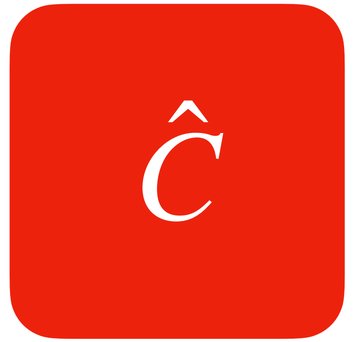
Generator

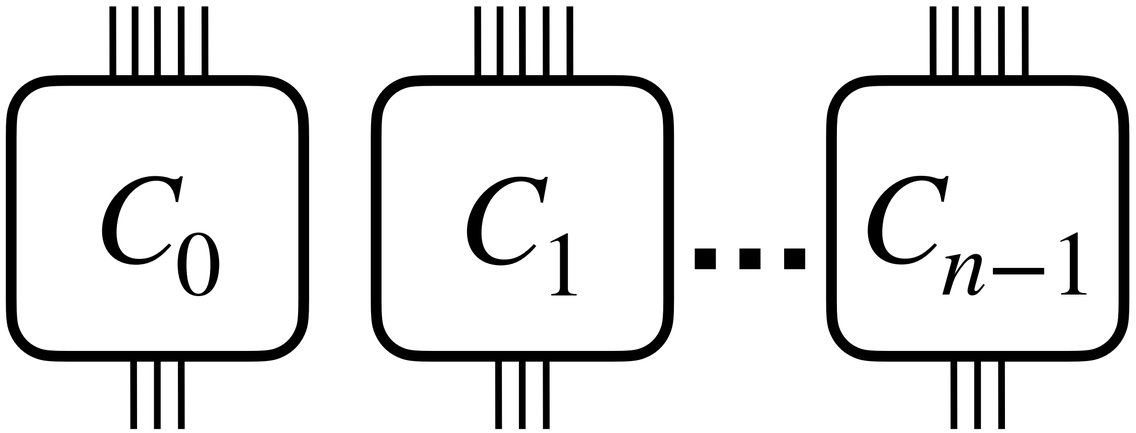
Evaluator

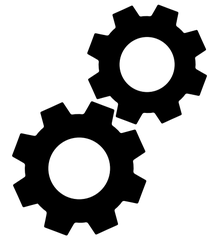
Goal: Sublinear in n

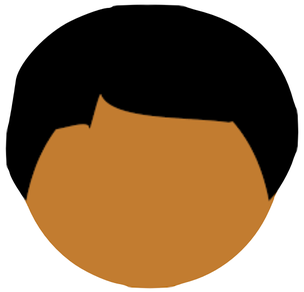








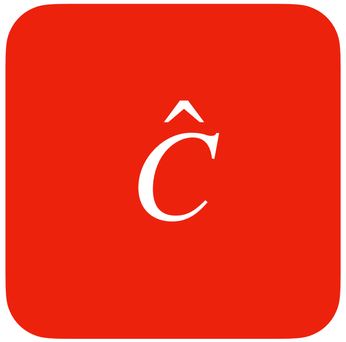


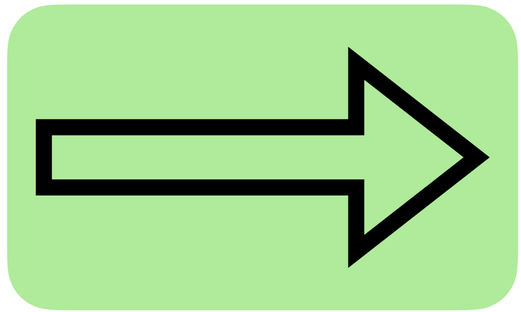


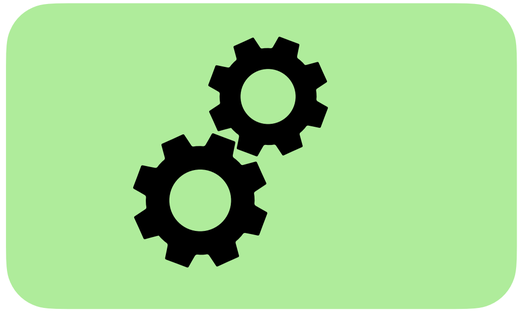
Generator

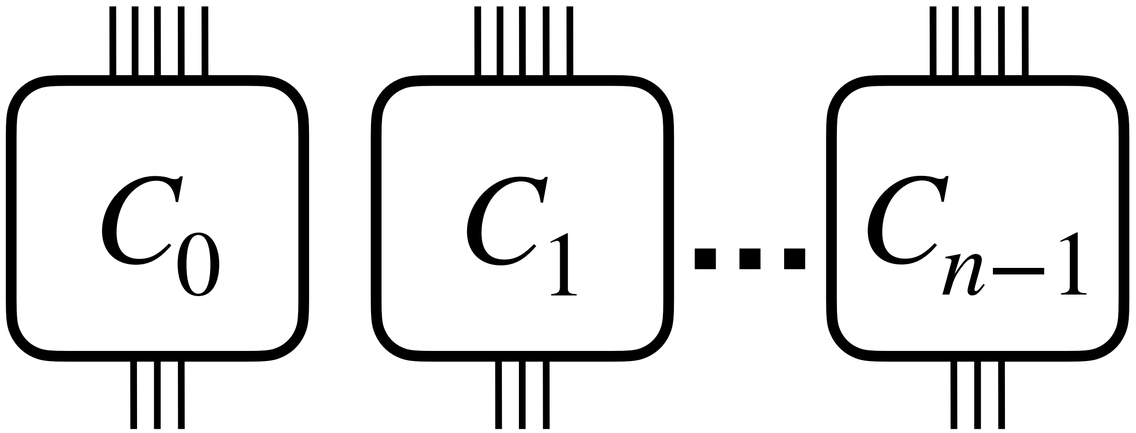
Evaluator

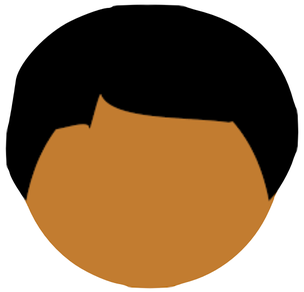
Goal: Sublinear in n











Generator

Evaluator

For functions with conditionals, achieve **sublinear* communication** and **sublinear* computation** for one party

* sublinear in the function description

For functions with conditionals, achieve **compact 2PC**

Compact 2PC

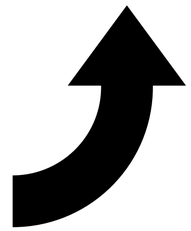
FHE allows compact 2PC, but requires expensive primitives

Stacked Garbling (Heath, Kolesnikov Crypto 2020) *does not* achieve compactness

Compact 2PC

FHE allows compact 2PC, but requires expensive primitives

Stacked Garbling (Heath, Kolesnikov Crypto 2020) *does not* achieve compactness

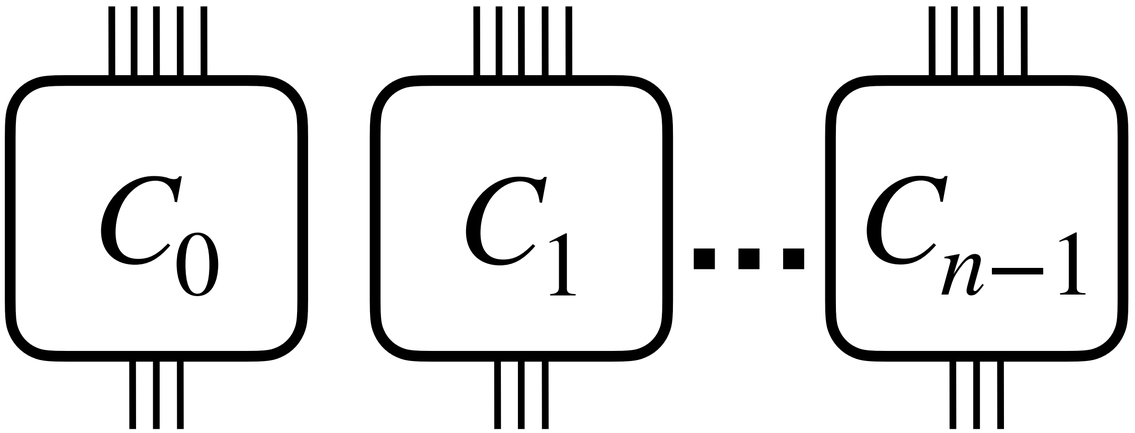
Our starting point 

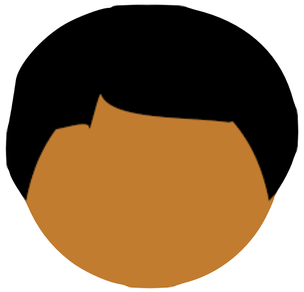
$$f(i,x) = \begin{cases} C_0(x) & \text{if } i = 0 \\ C_1(x) & \text{if } i = 1 \\ \cdots & \\ C_{n-1}(x) & \text{if } i = n-1 \end{cases}$$

$\tilde{O}(\sqrt{n})$ **communication**

$\tilde{O}(\sqrt{n})$ **evaluator computation**

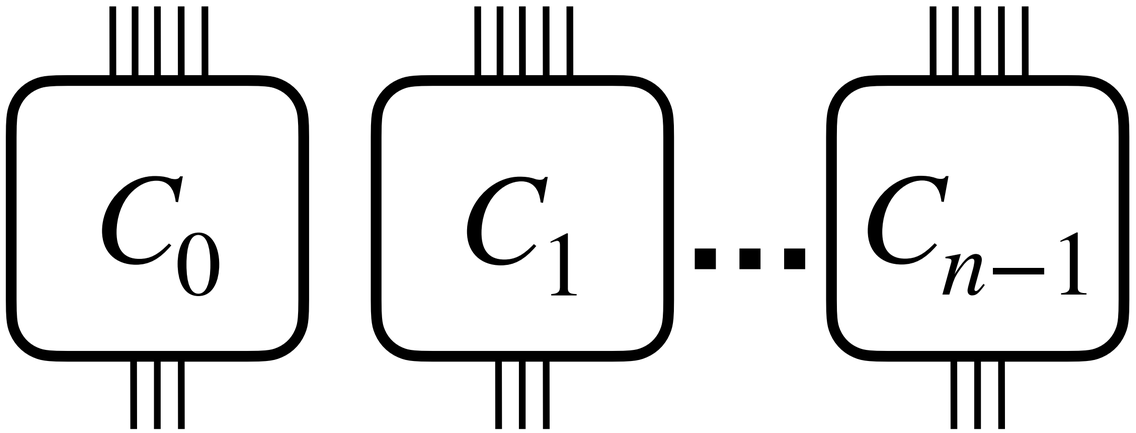
Stacked Garbling





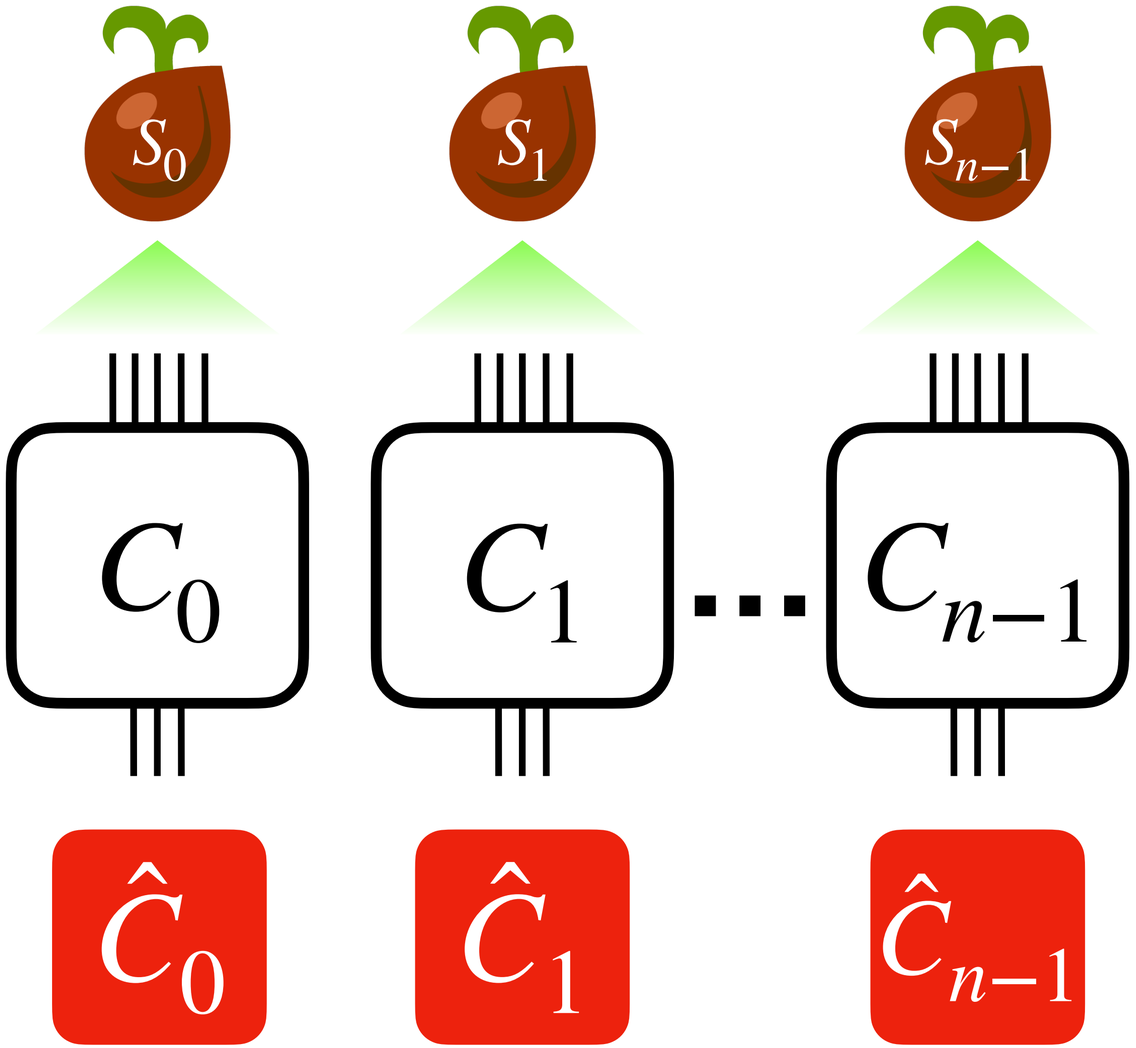
Generator

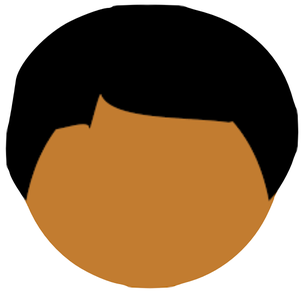
Stacked Garbling



Generator

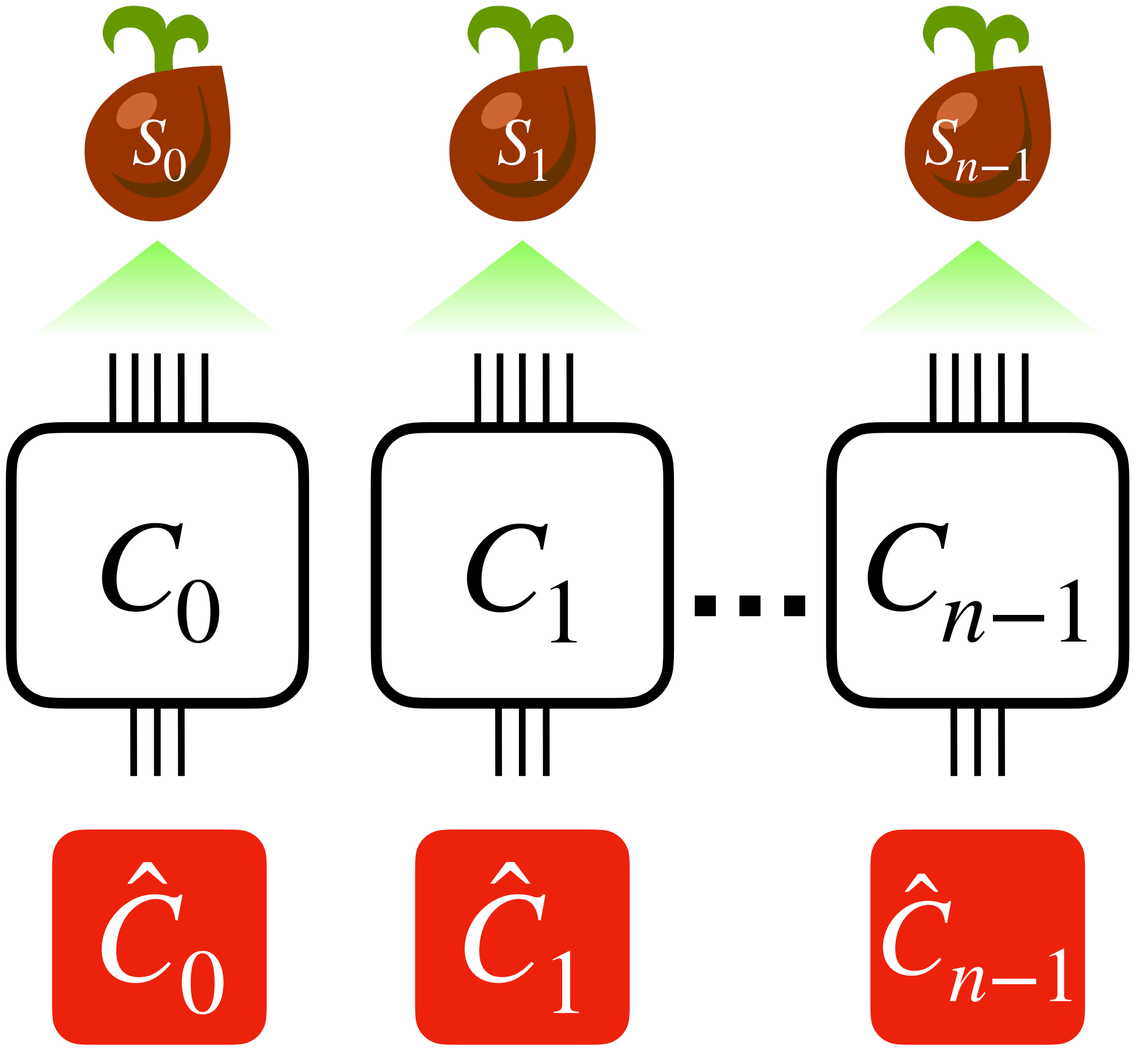
Stacked Garbling

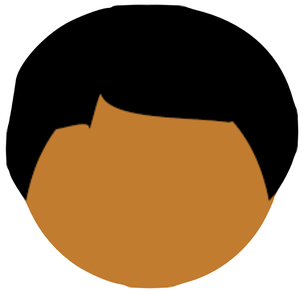




Generator

Stacked Garbling

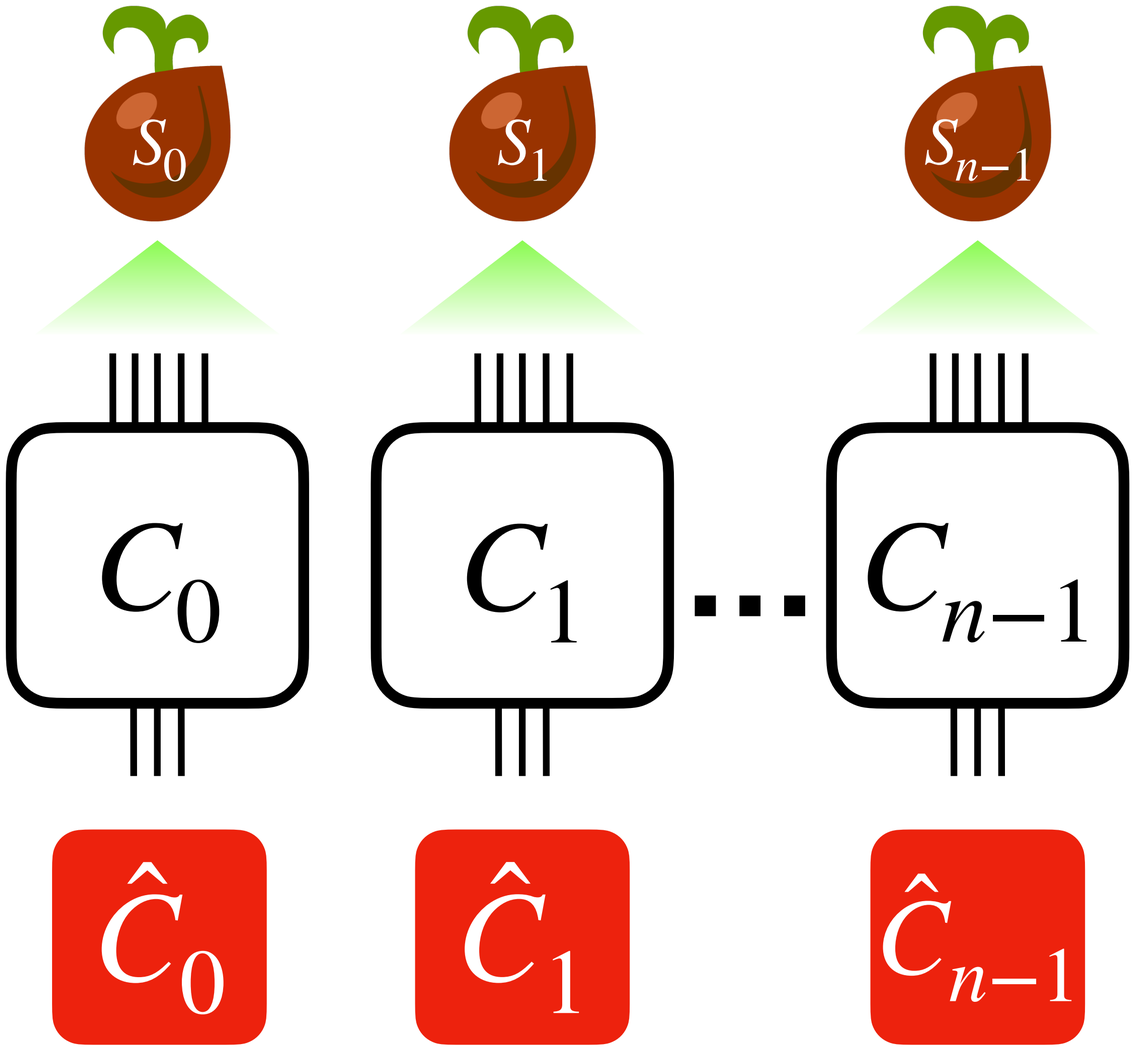




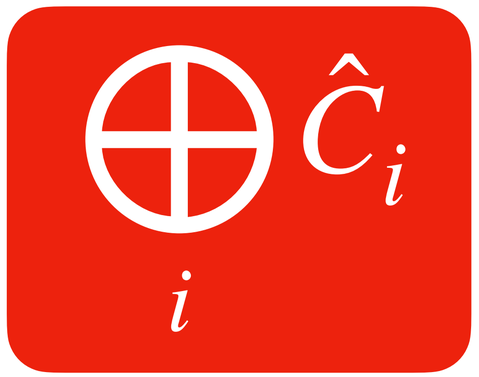
Generator

$$\bigoplus_i \hat{C}_i$$

Stacked Garbling



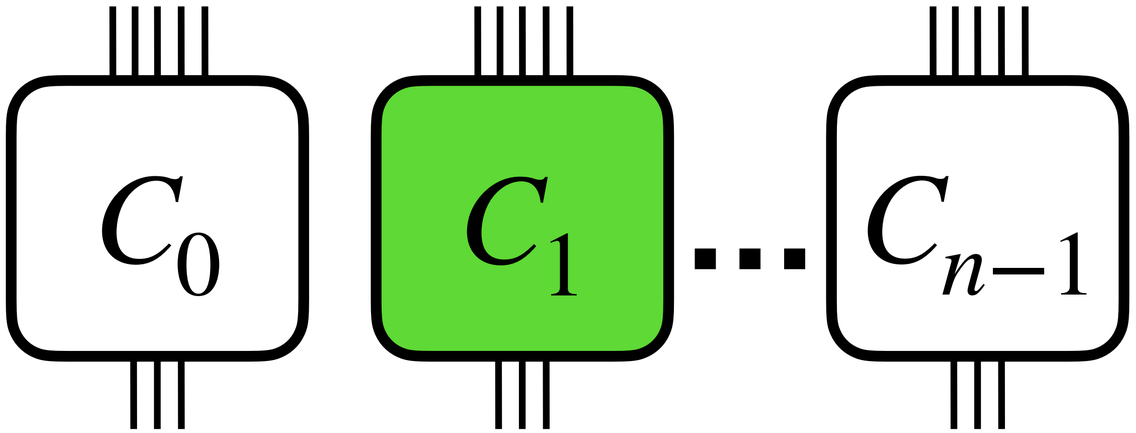
Size independent of n



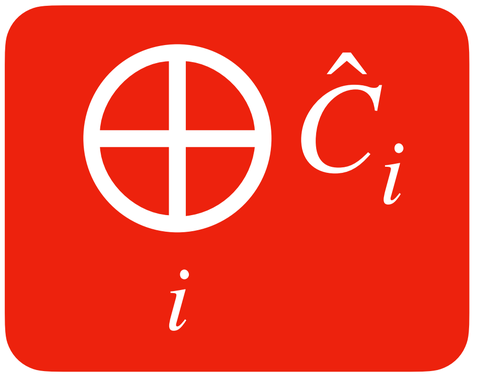
Generator

Stacked Garbling

… where Evaluator knows the active branch

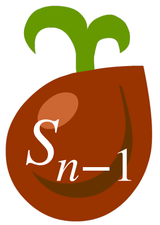


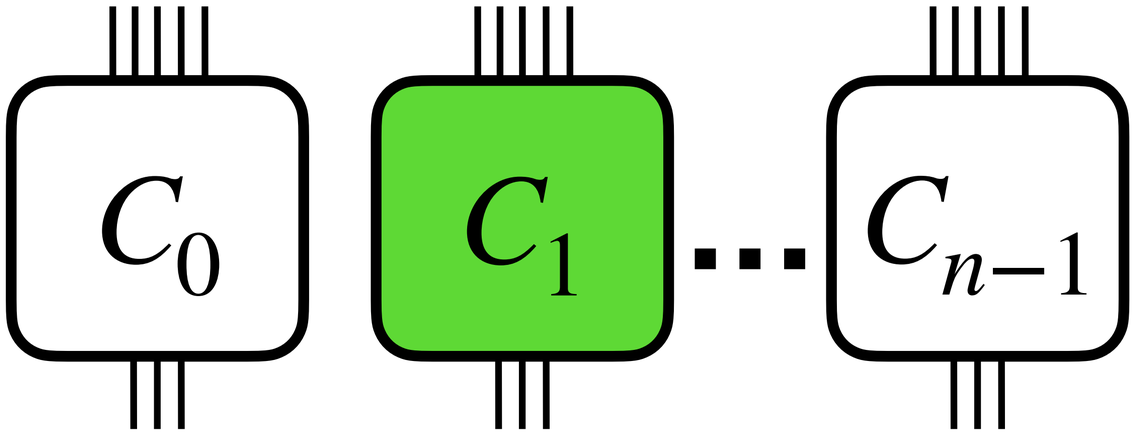
Evaluator



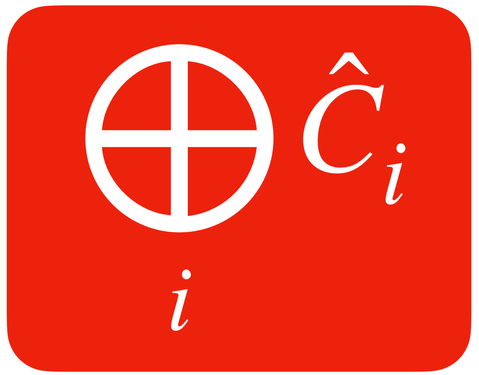
Stacked Garbling

… where Evaluator knows the active branch



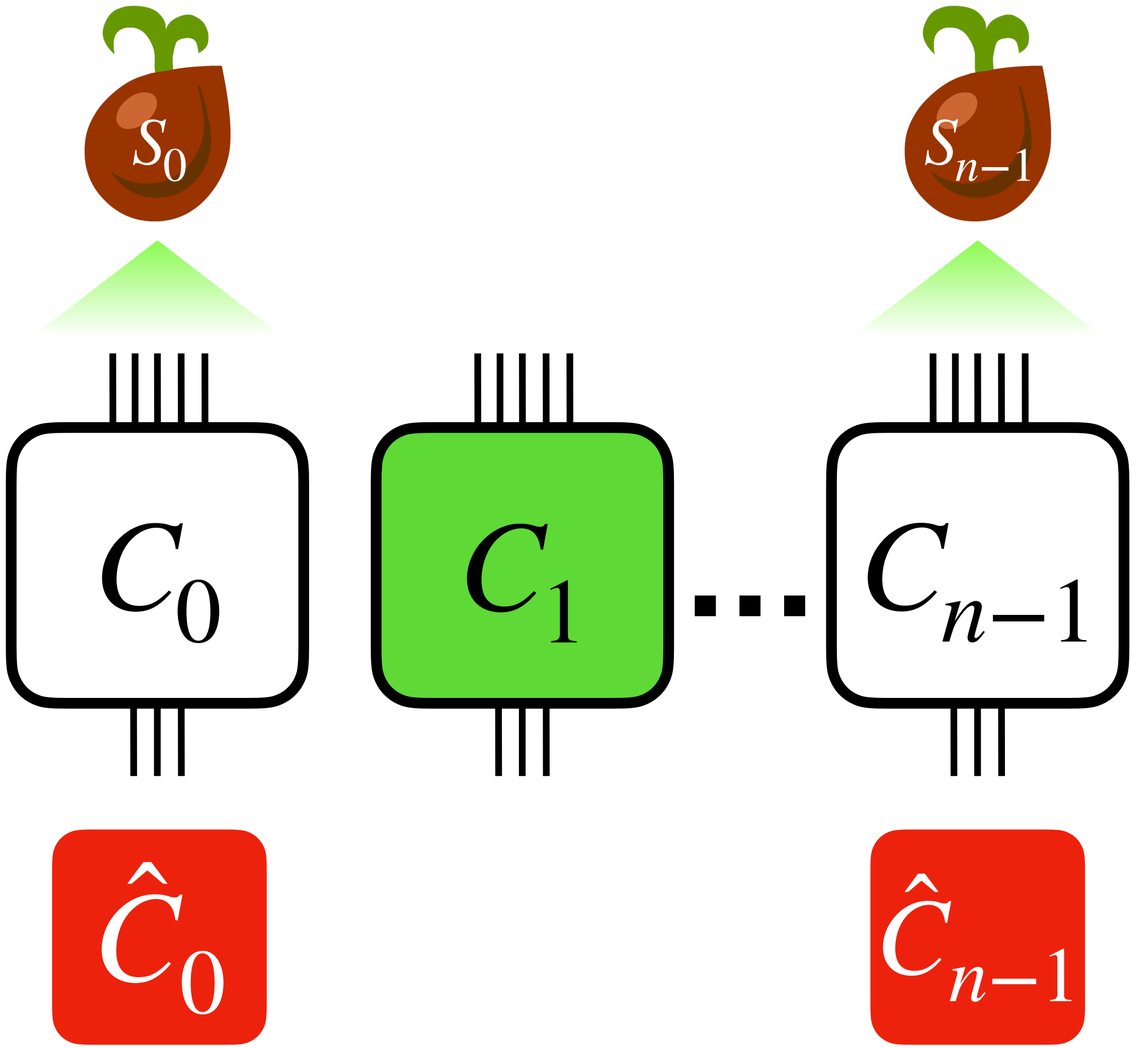


Evaluator



Stacked Garbling

… where Evaluator knows the active branch

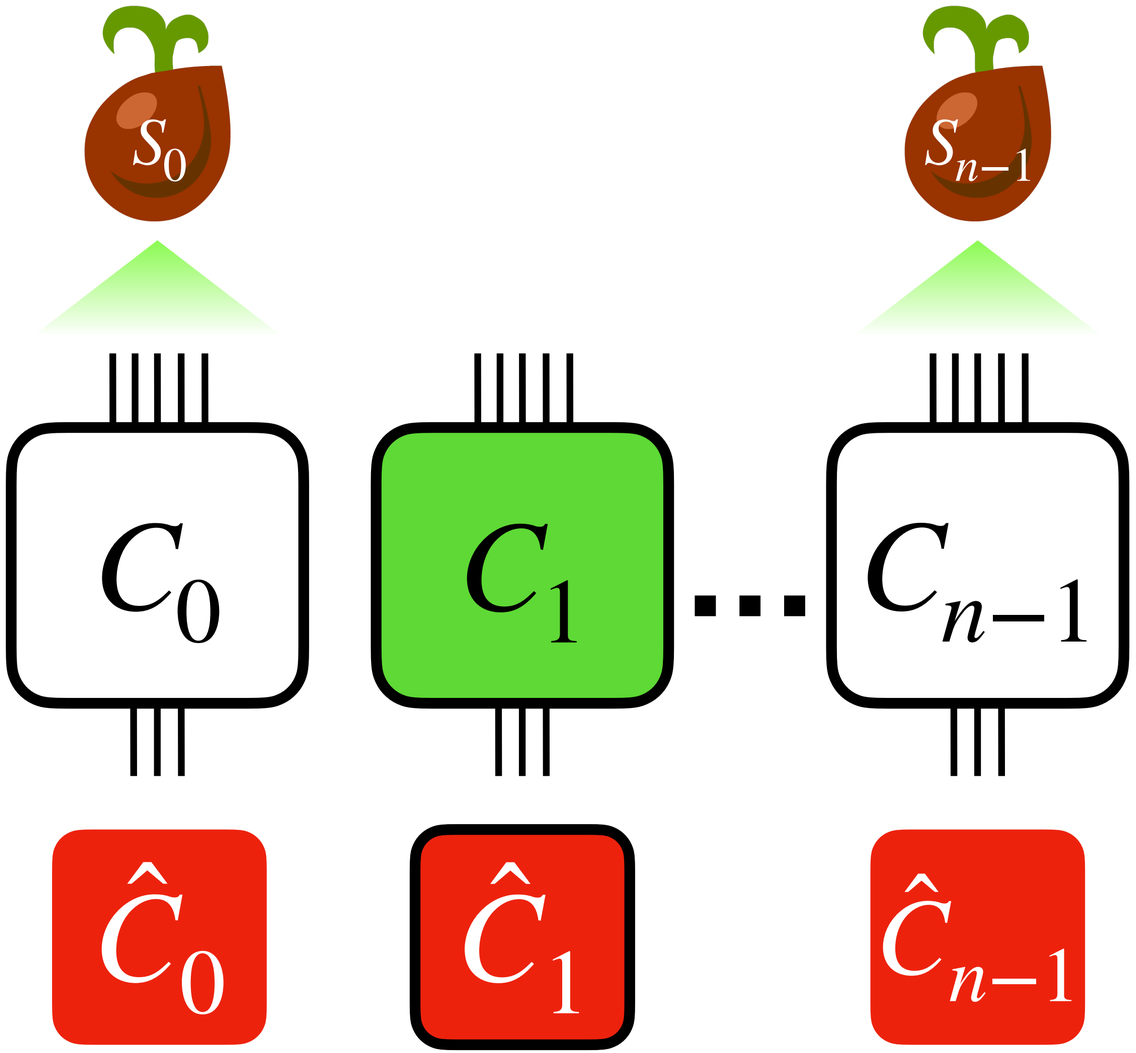


Evaluator

$$\bigoplus_i \hat{C}_i$$

Stacked Garbling

… where Evaluator knows the active branch



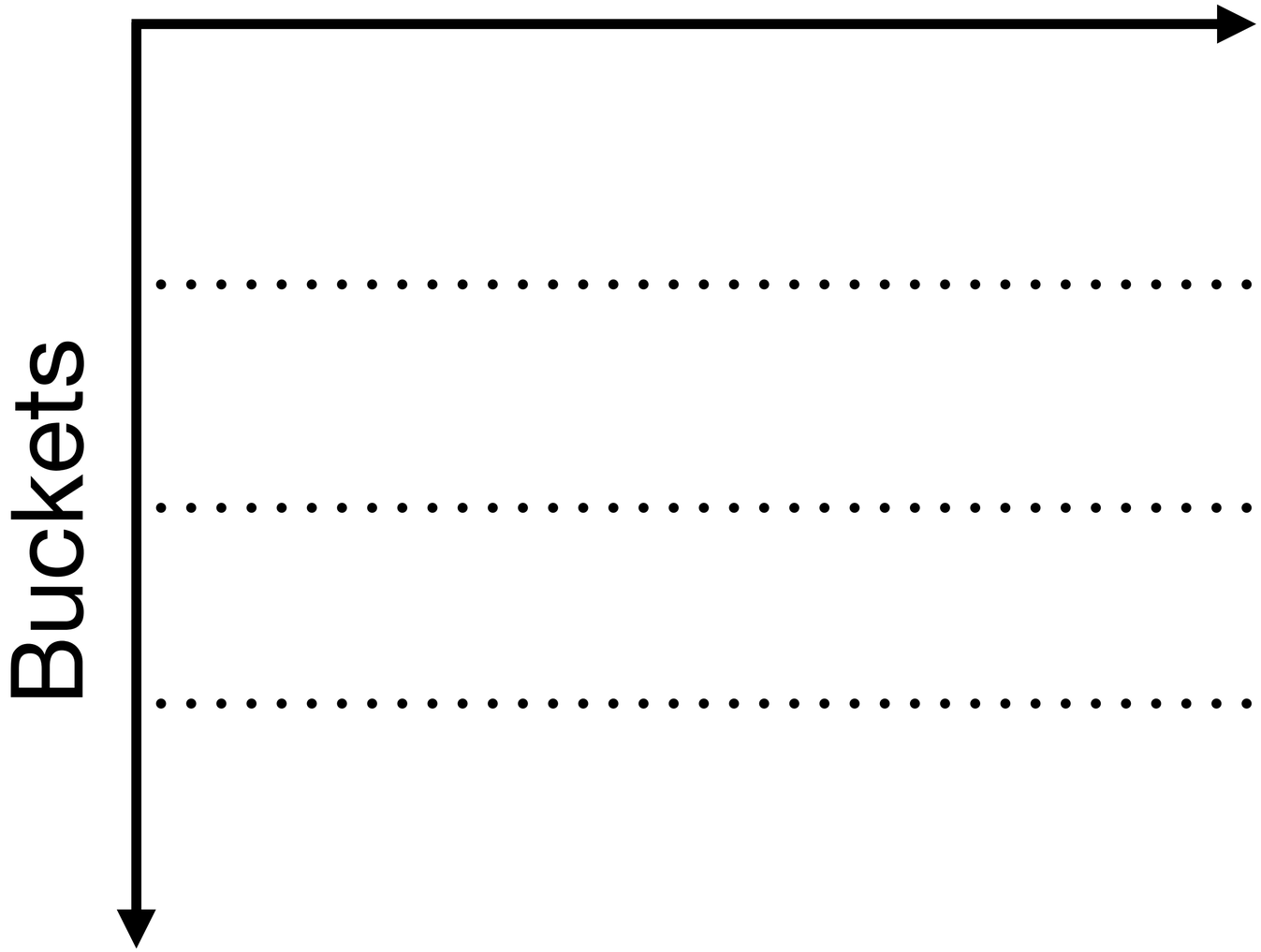
Evaluator

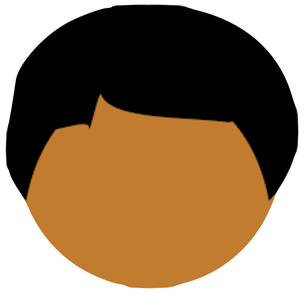
$$\bigoplus_i \hat{C}_i$$

Stacked Garbling achieves sublinear communication…

But not sublinear Evaluator computation

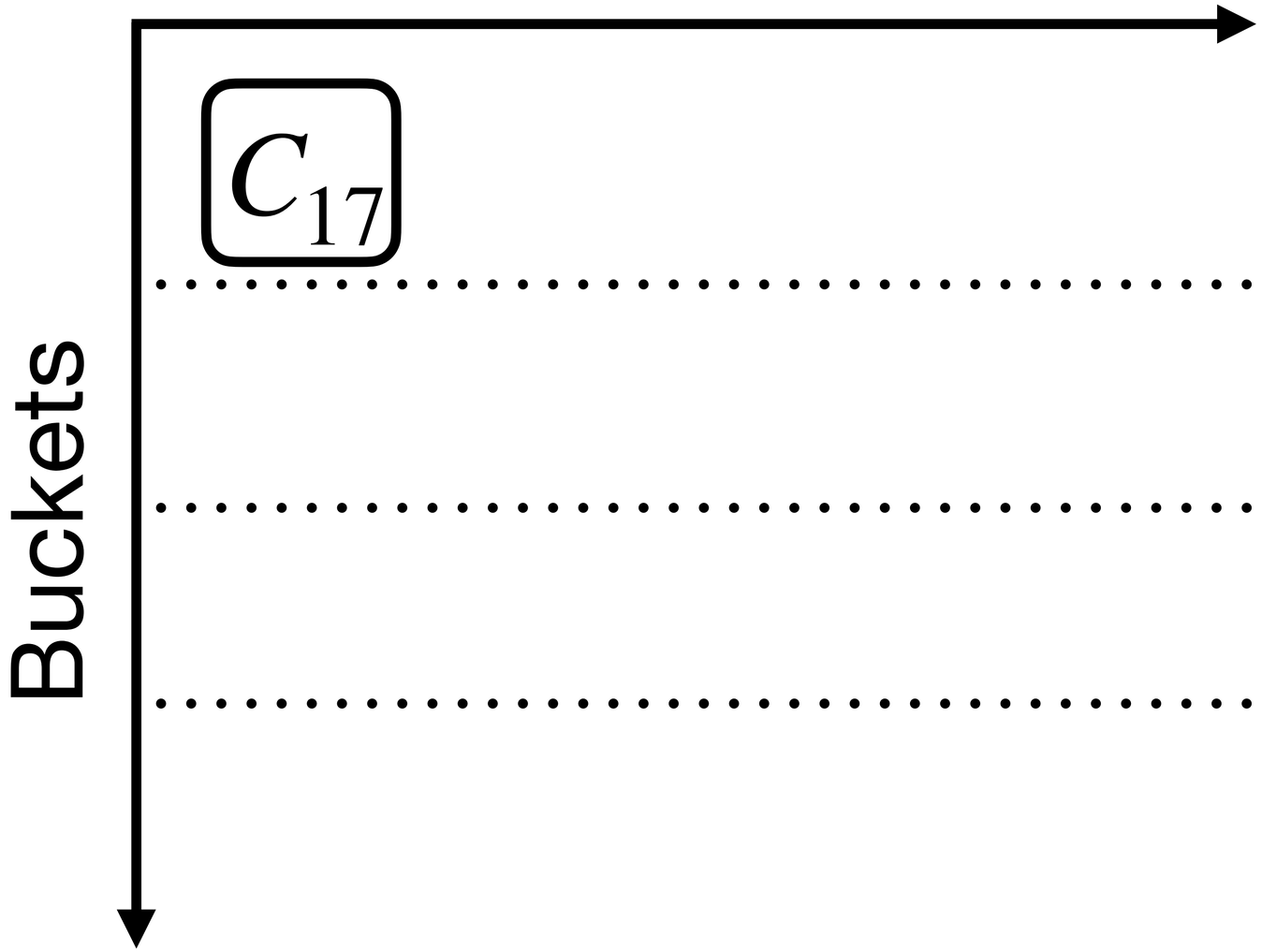
$\tilde{O}(\sqrt{n})$ Bucket Content

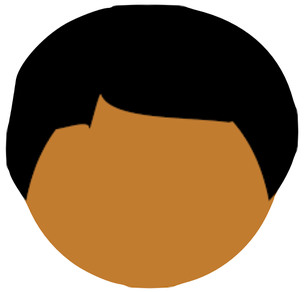




Generator

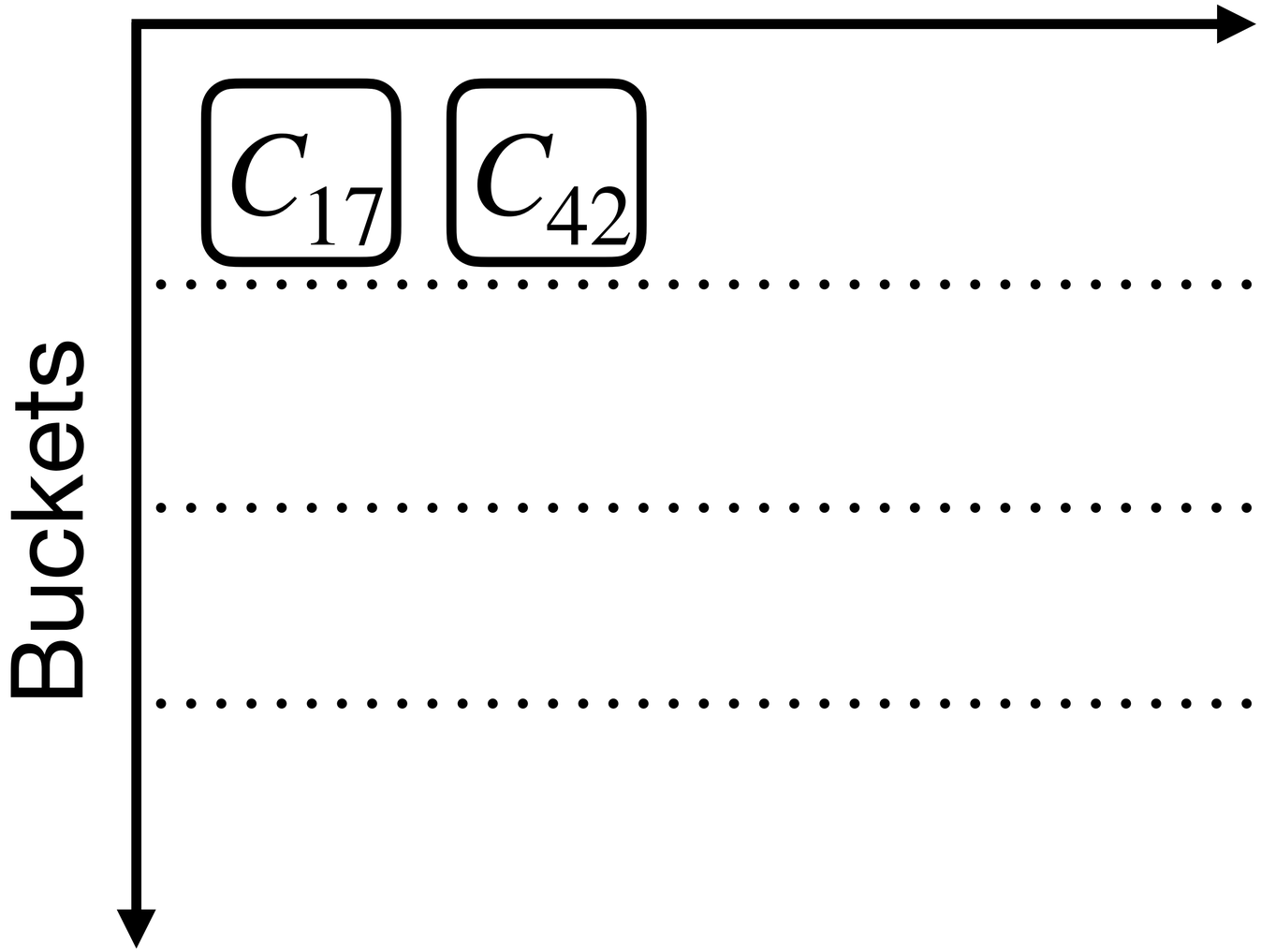
$\tilde{O}(\sqrt{n})$ Bucket Content

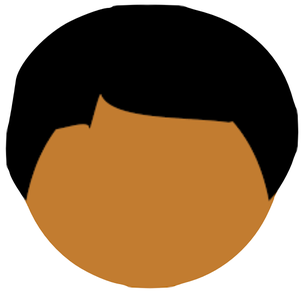




Generator

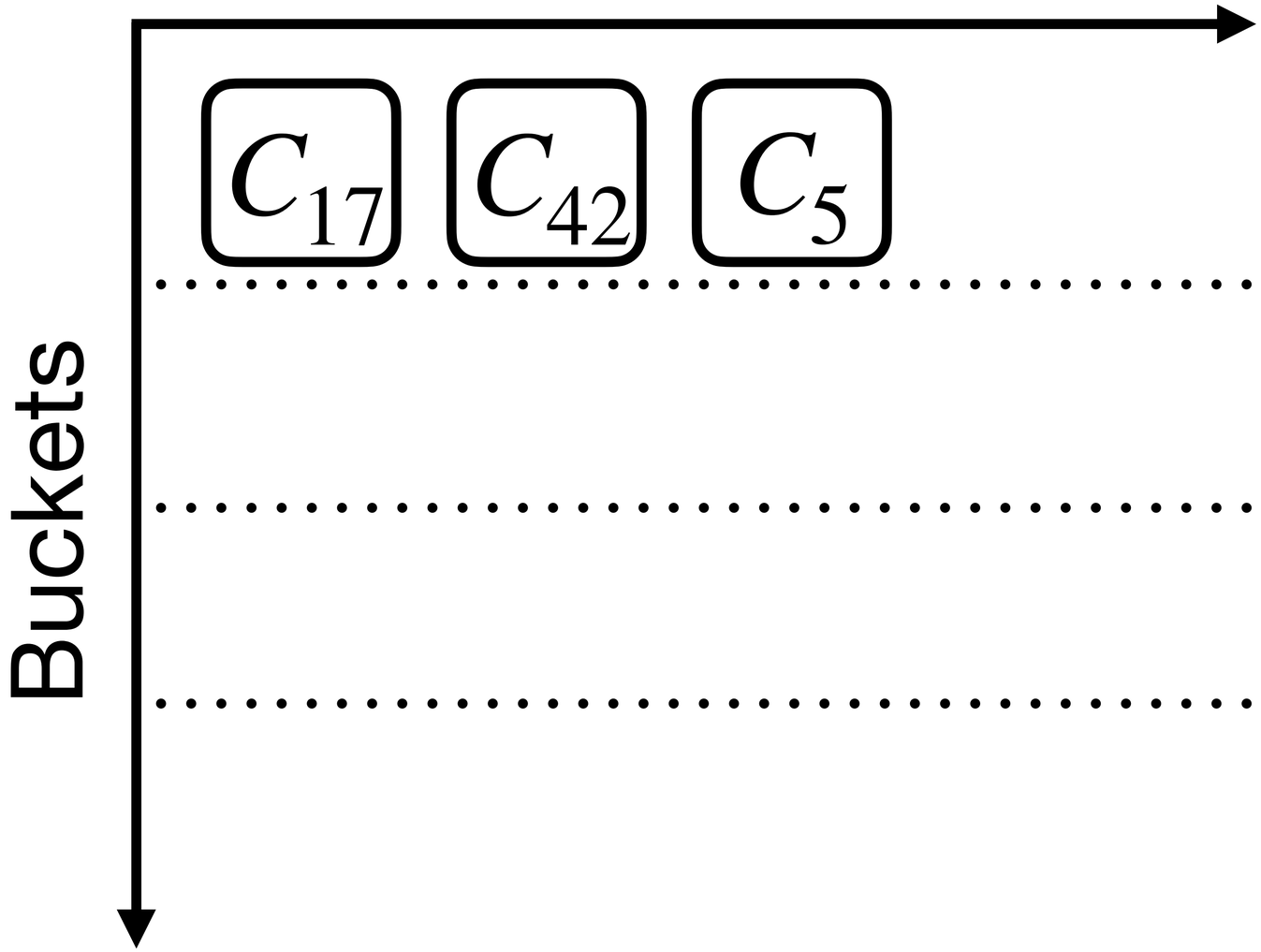
$\tilde{O}(\sqrt{n})$ Bucket Content

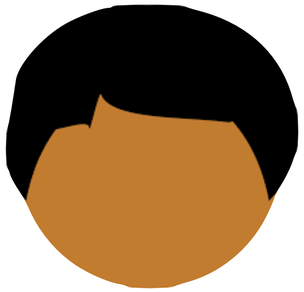




Generator

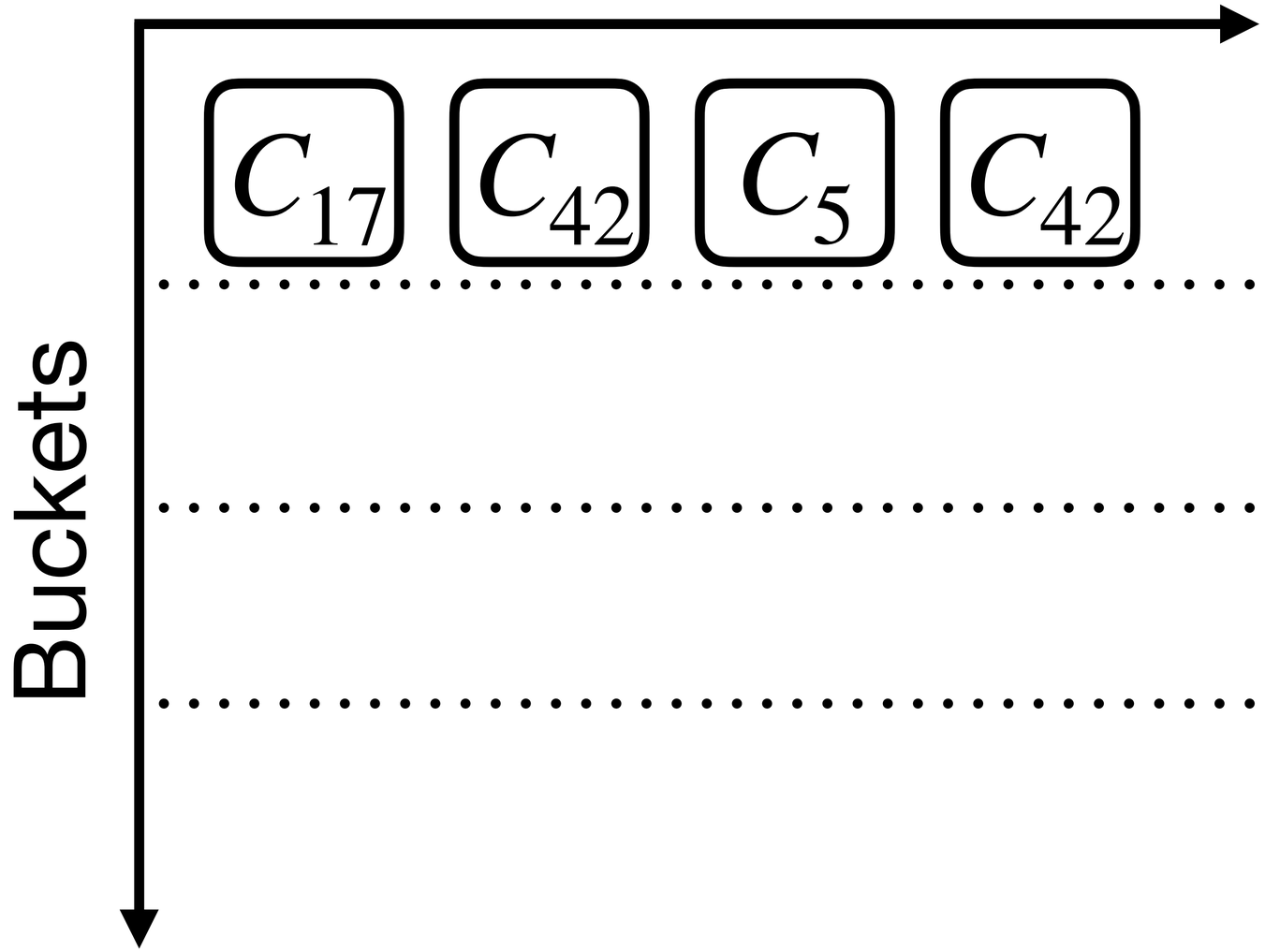
$\tilde{O}(\sqrt{n})$ Bucket Content

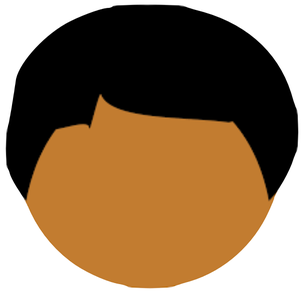




Generator

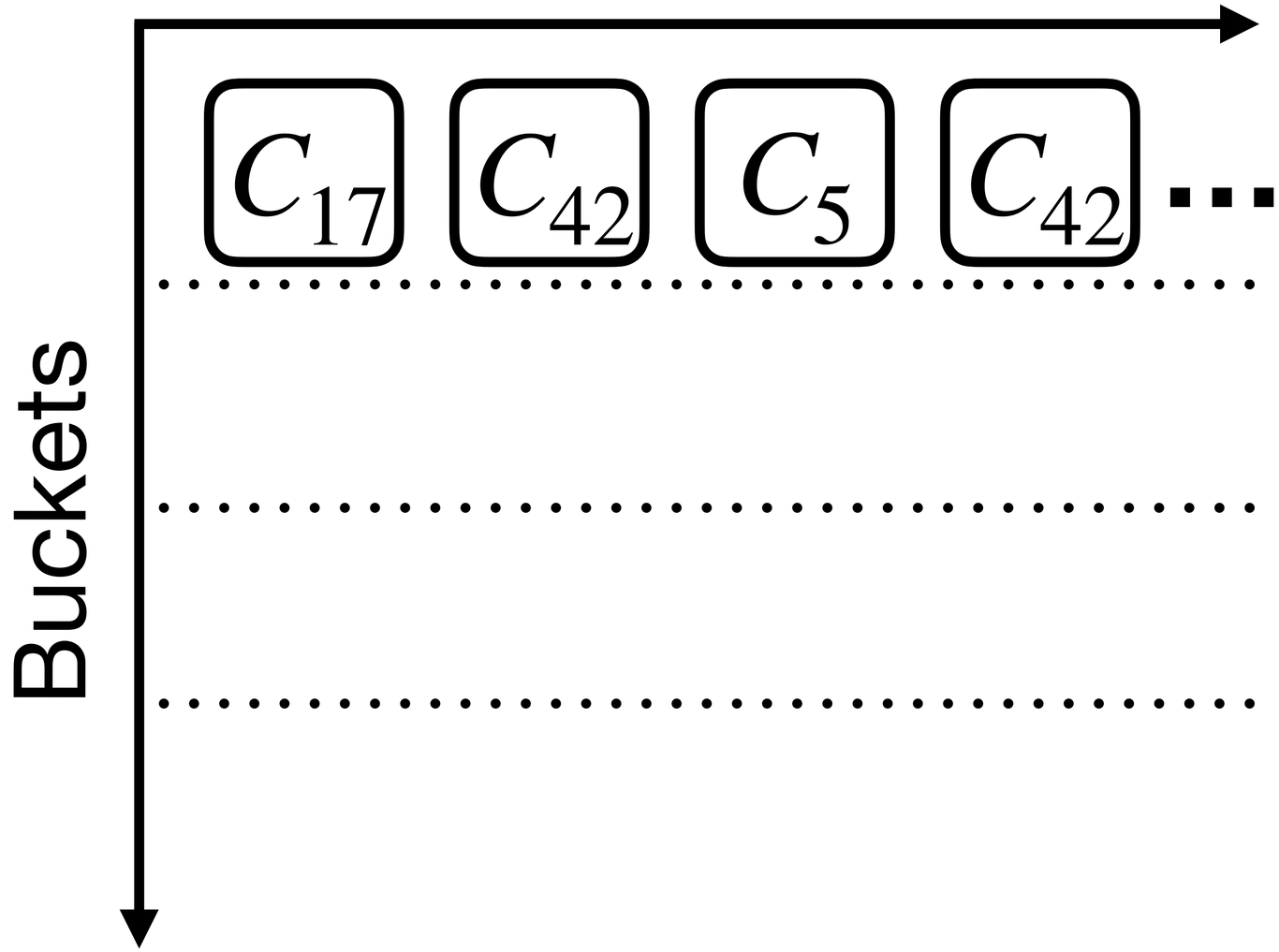
$\tilde{O}(\sqrt{n})$ Bucket Content

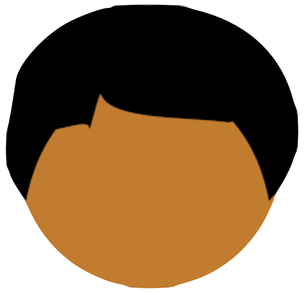




Generator

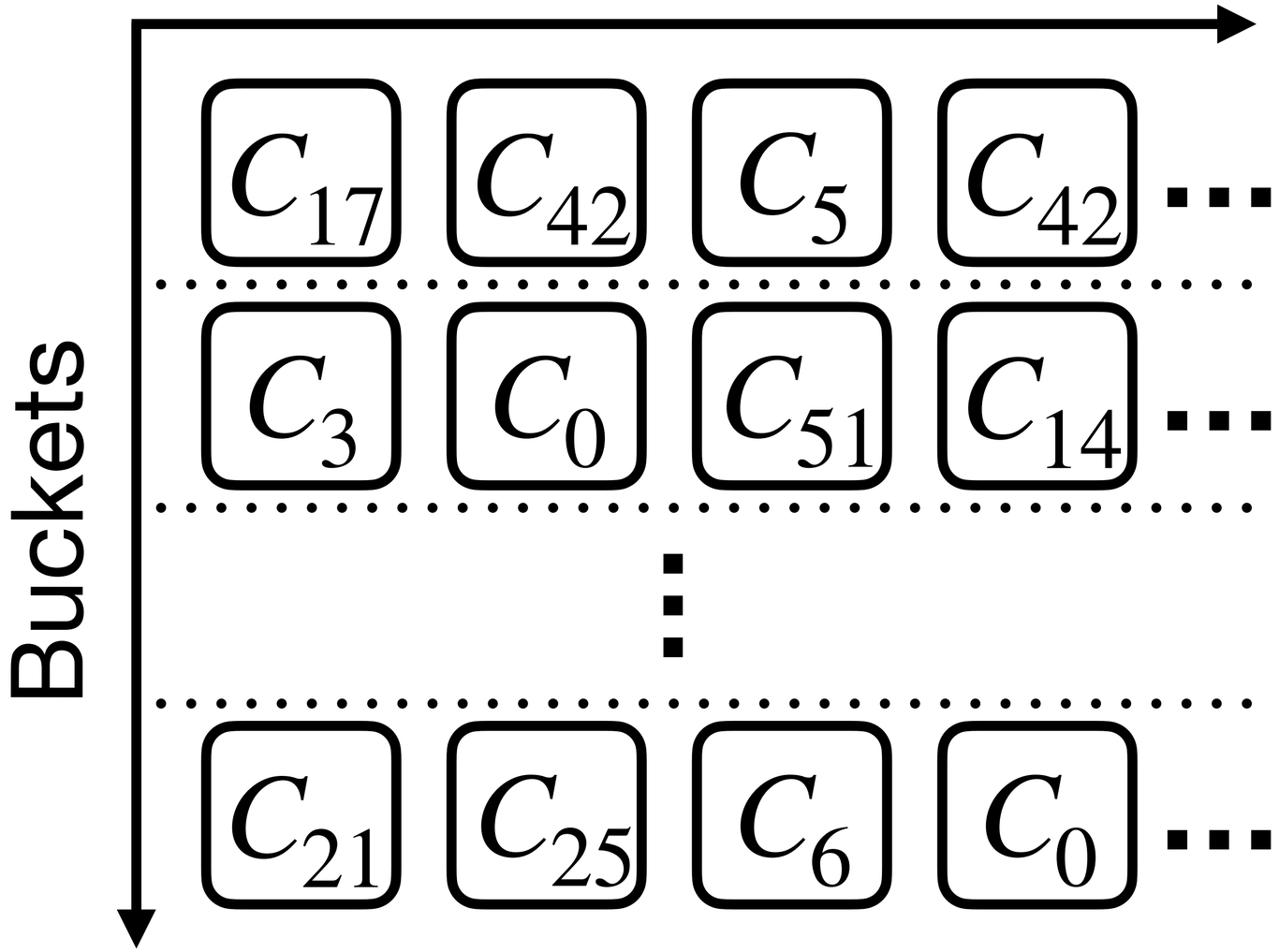
$\tilde{O}(\sqrt{n})$ Bucket Content

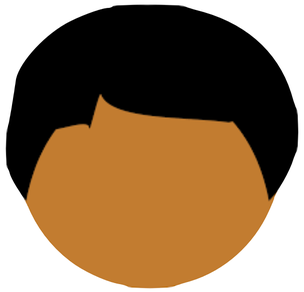




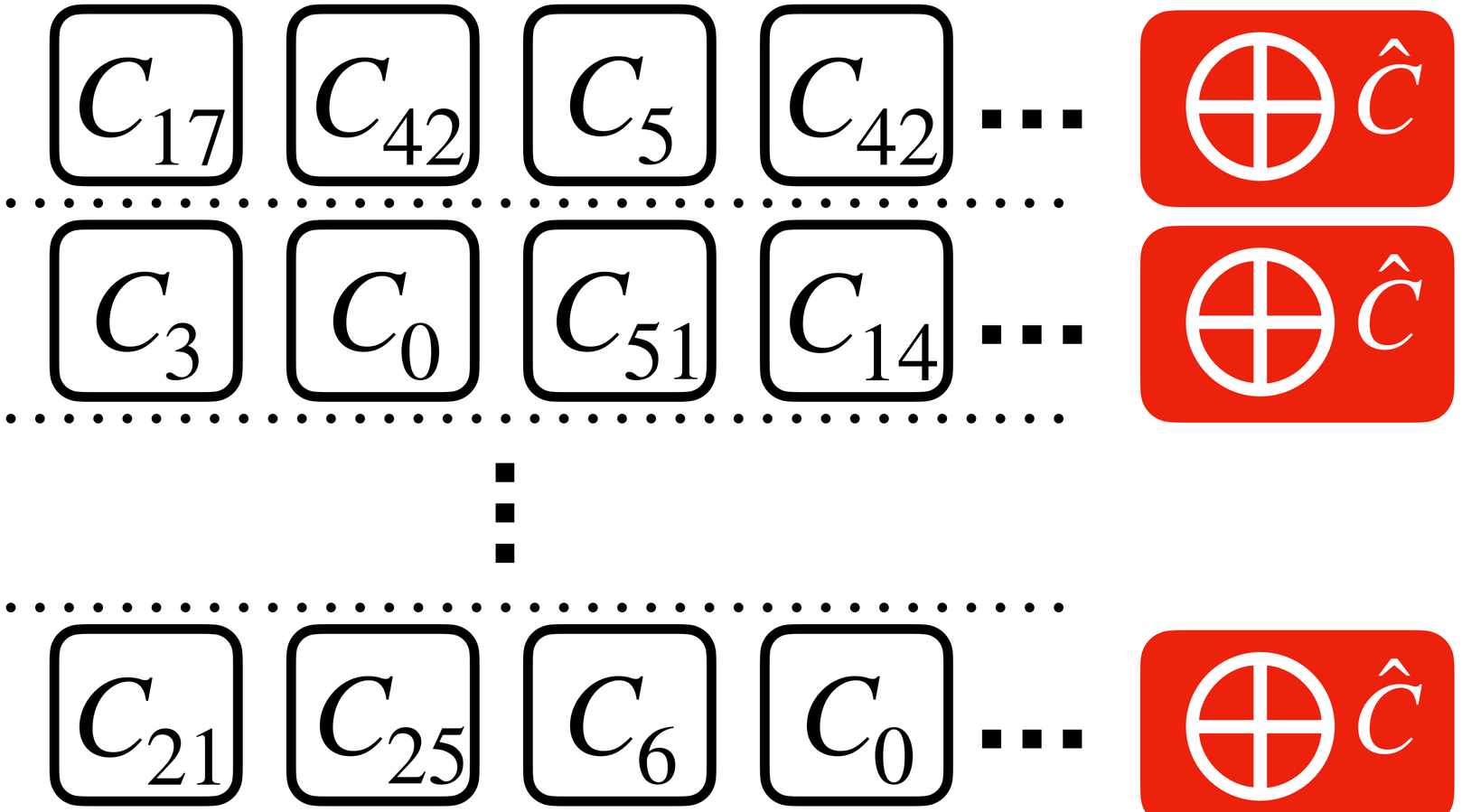
Generator

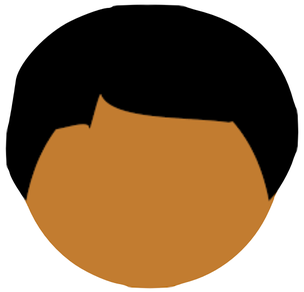
$\tilde{O}(\sqrt{n})$ Bucket Content



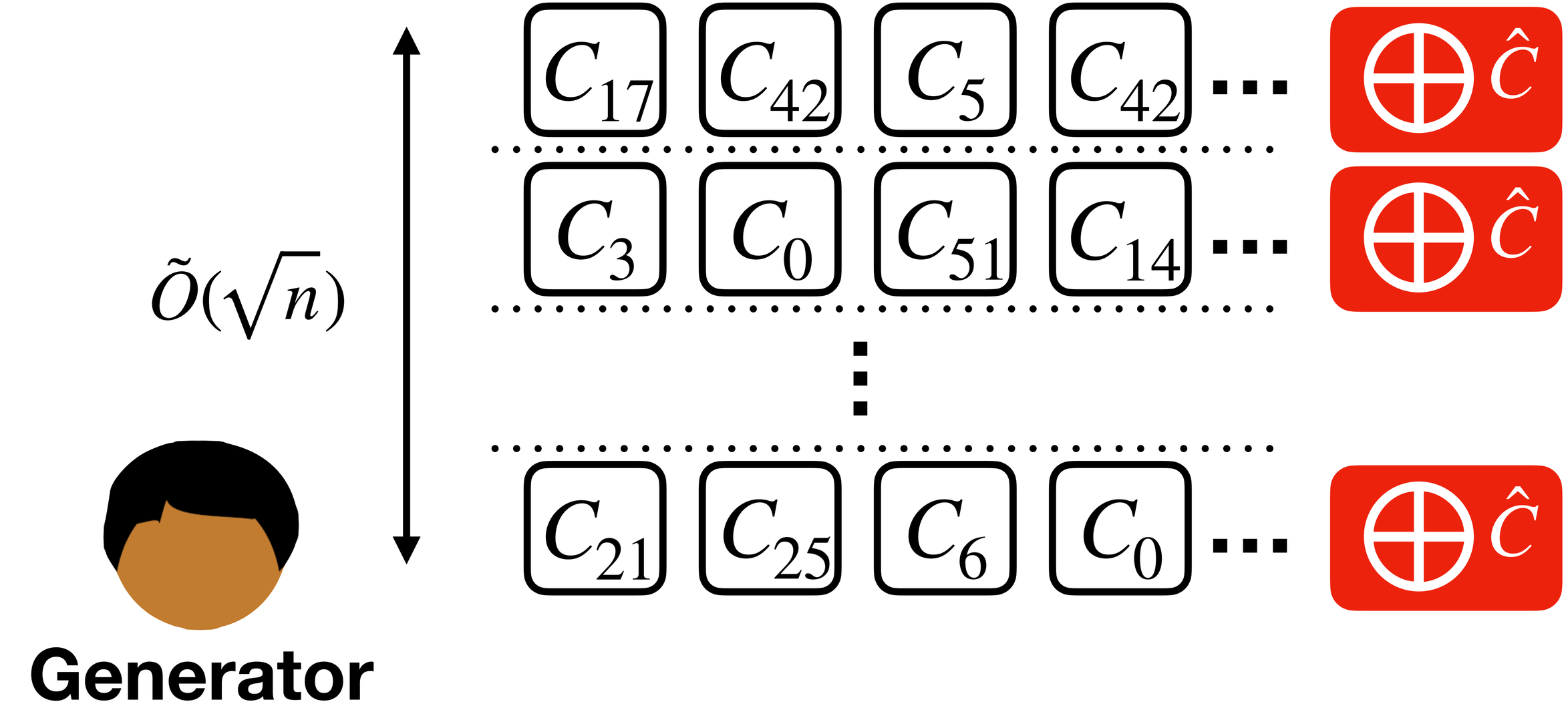


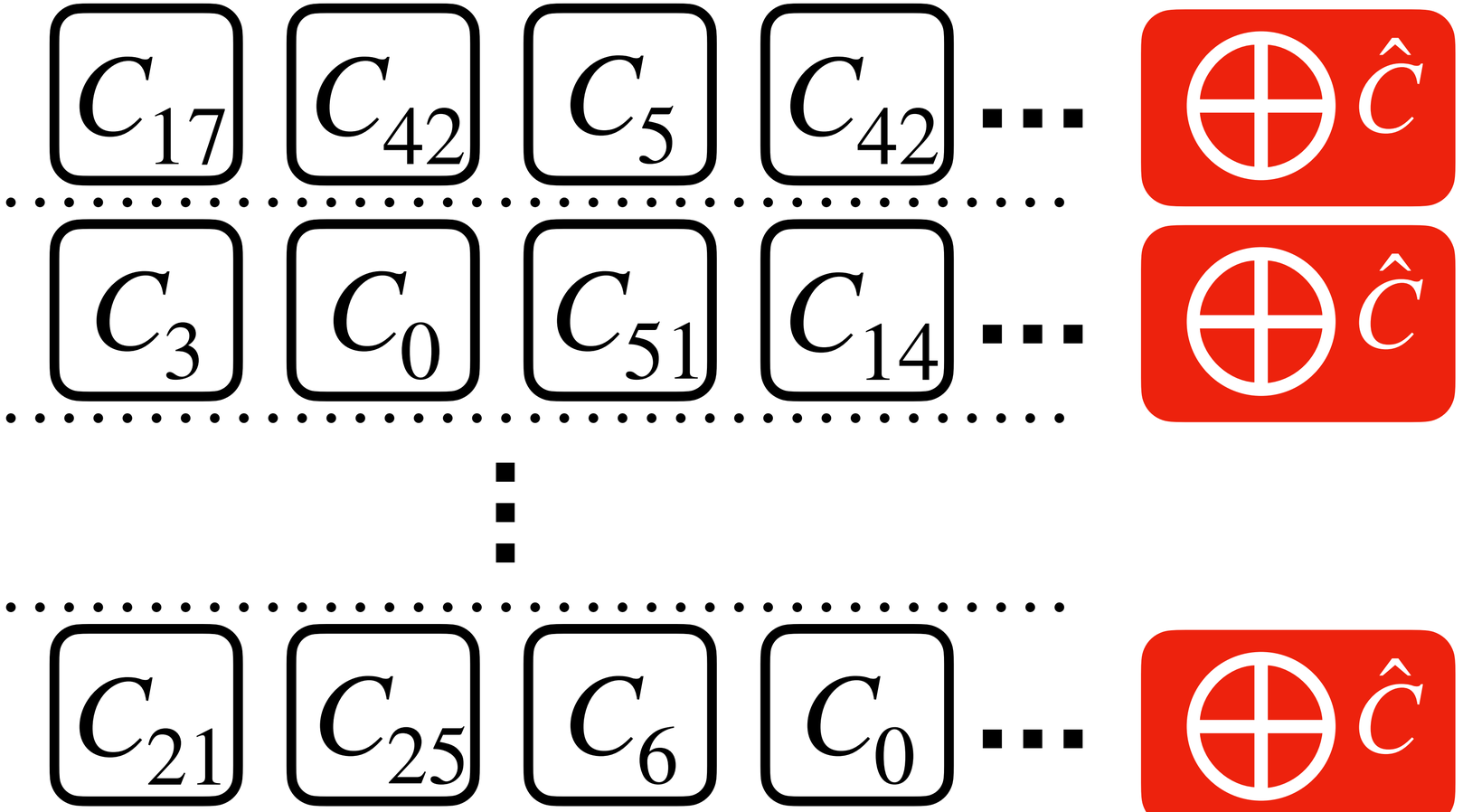
Generator





Generator


C_{17} C_{42} C_5 C_{42} ⋯ $\hat{C}$
C_3 C_0 C_{51} C_{14} ⋯ $\hat{C}$
C_{21} C_{25} C_6 C_0 ⋯ $\hat{C}$
$\tilde{O}(\sqrt{n})$
Generator



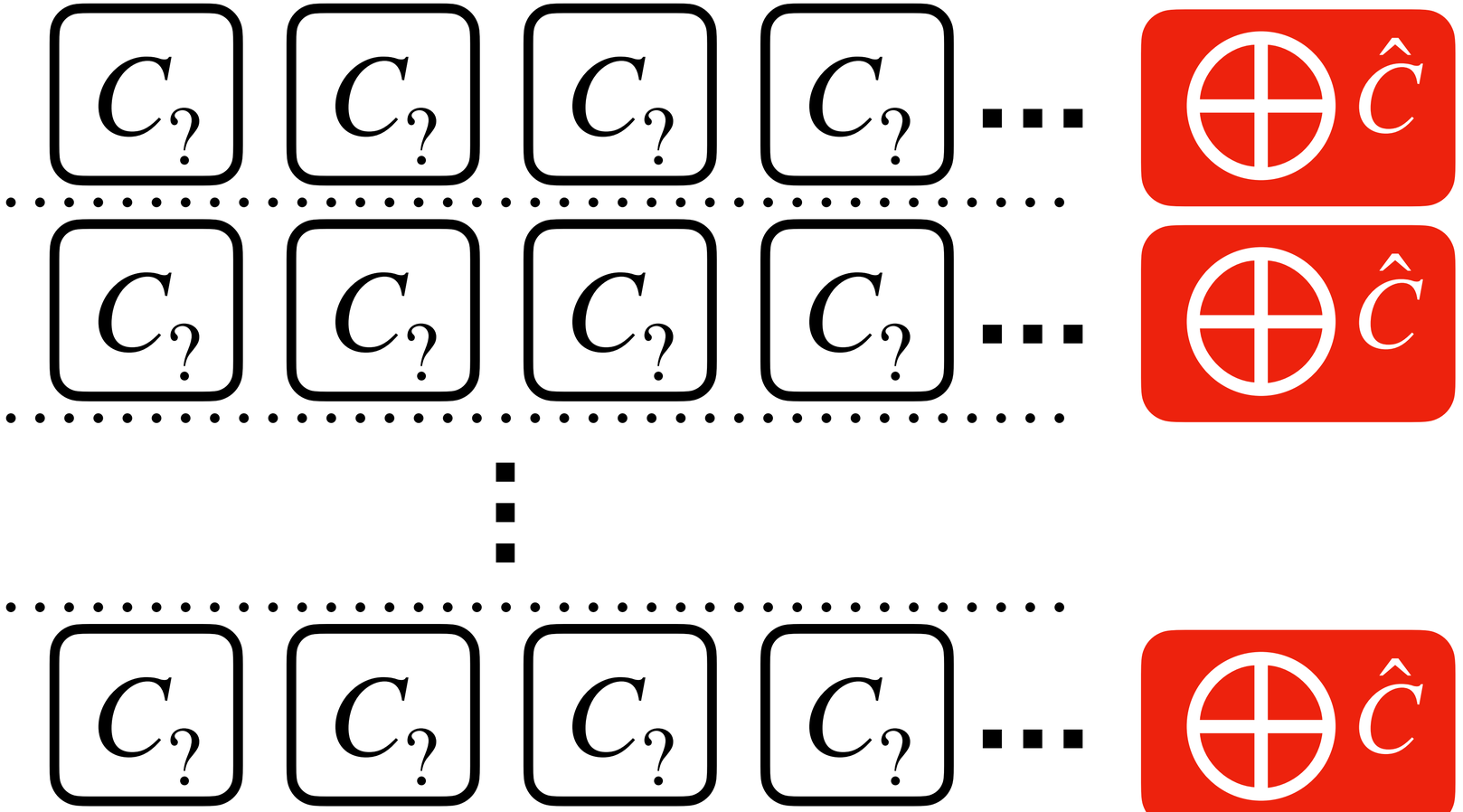
Evaluator

Universal Circuit

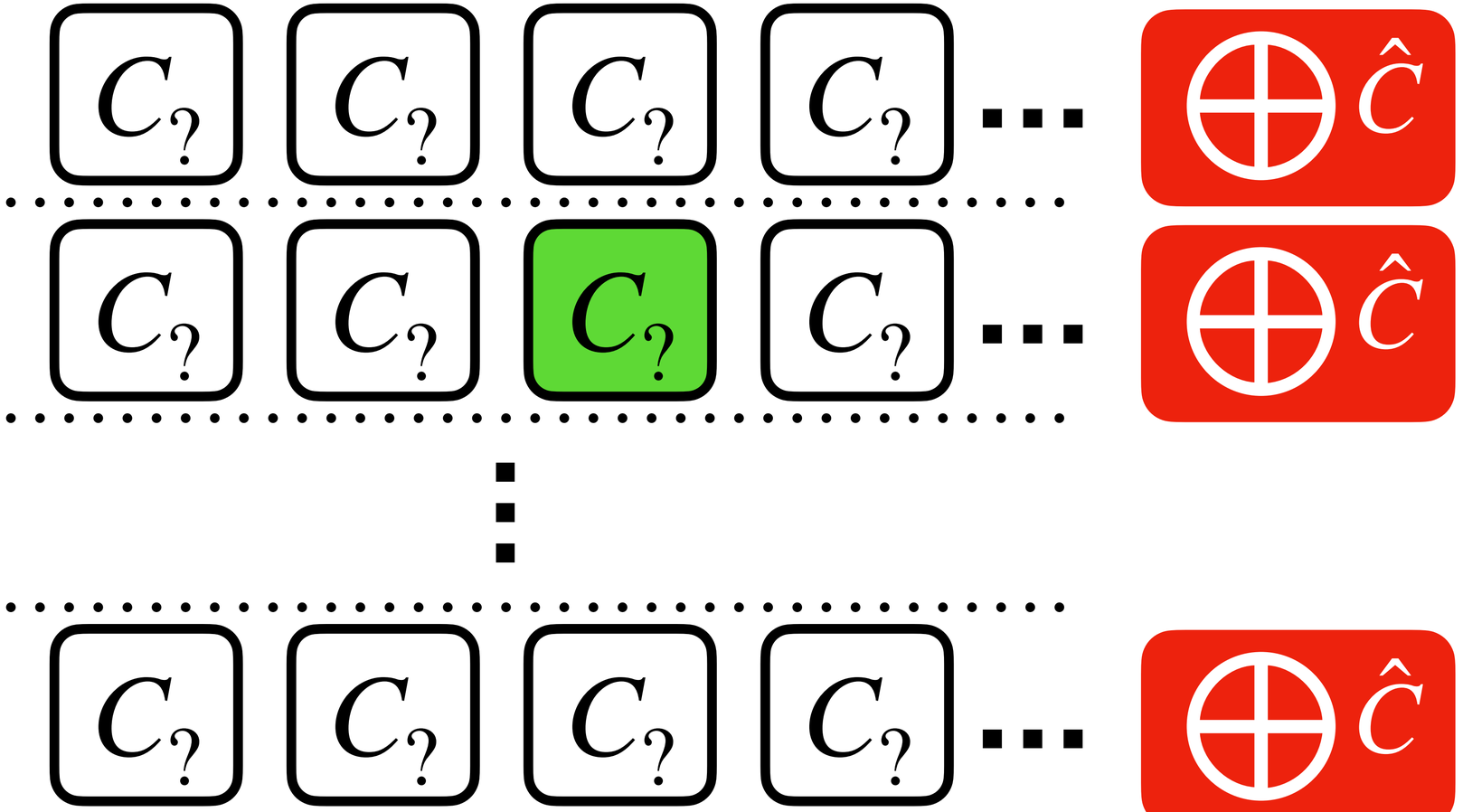
Universal Circuit

$O(\log |C|)$ overhead

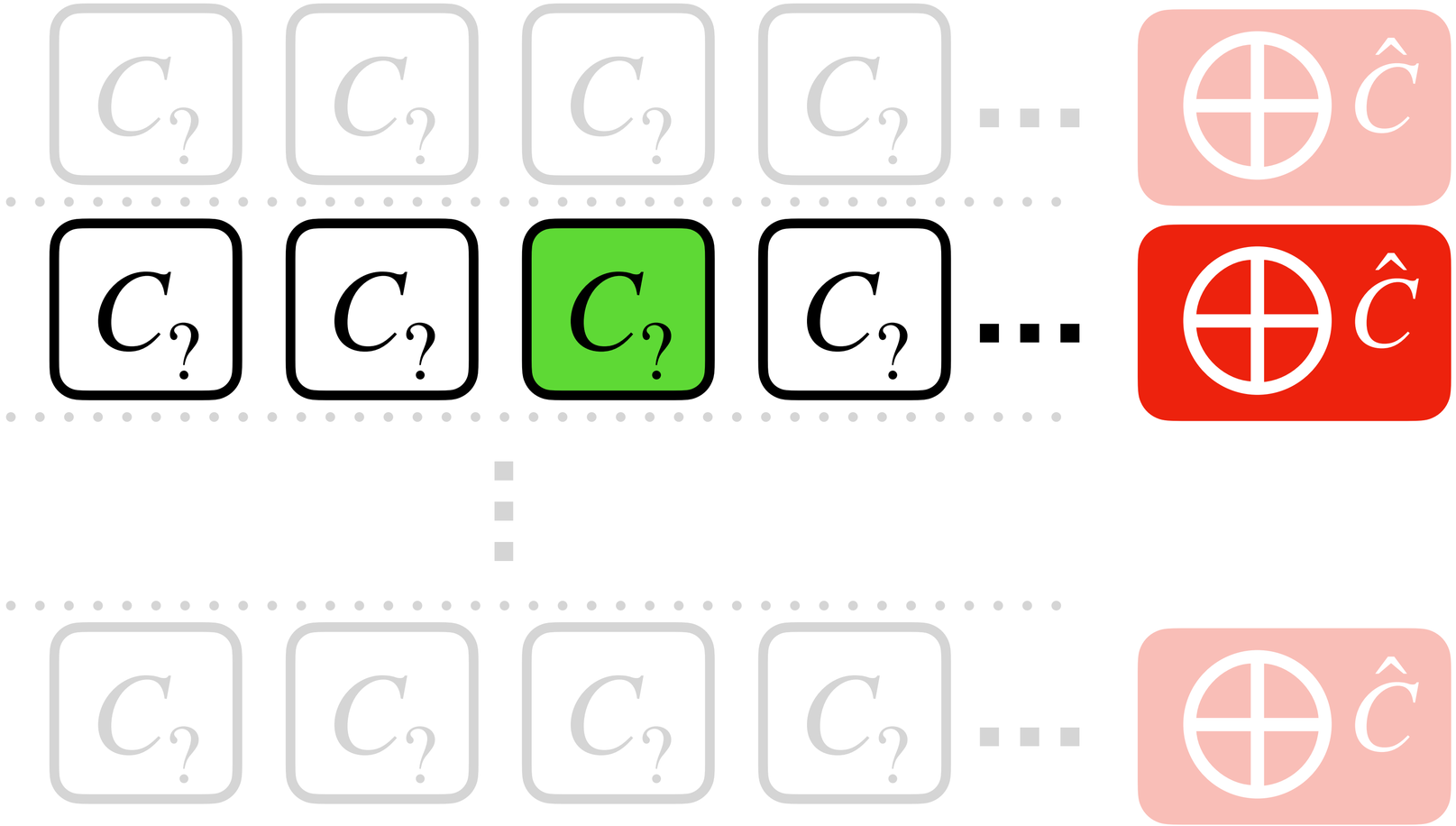
$O(1)$ overhead in special cases



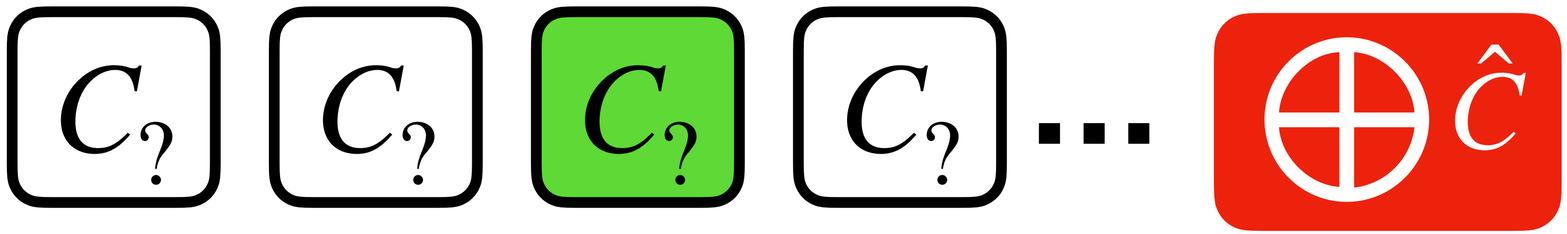
Evaluator



Evaluator

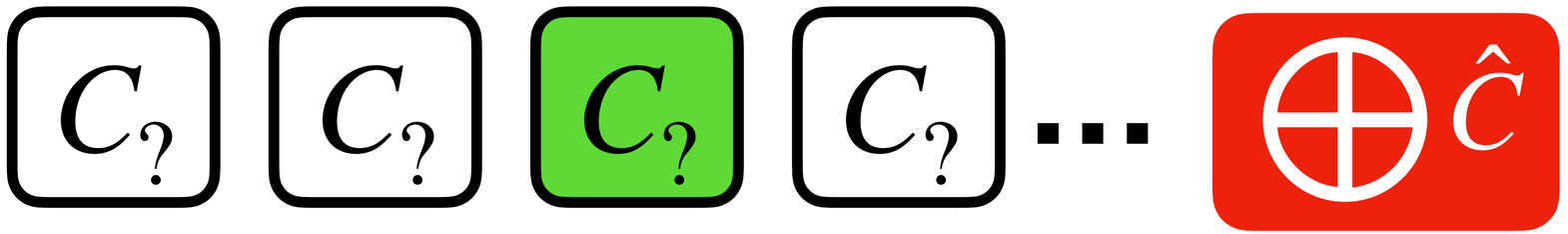


Evaluator

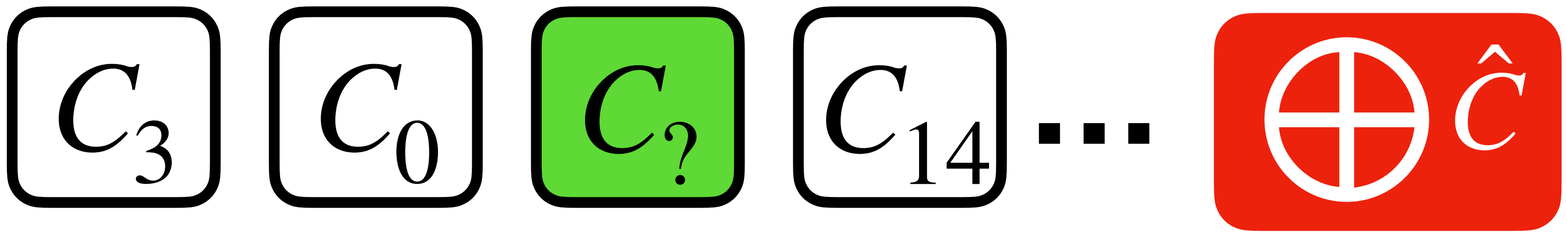


Evaluator

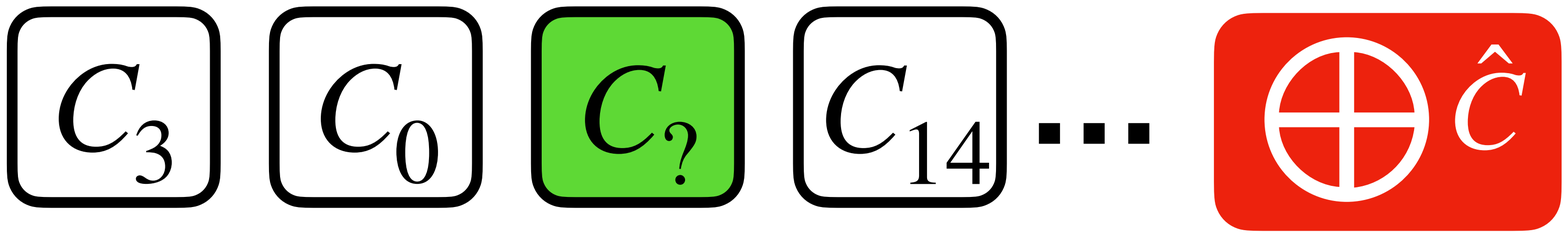
$\tilde{O}(\sqrt{n})$



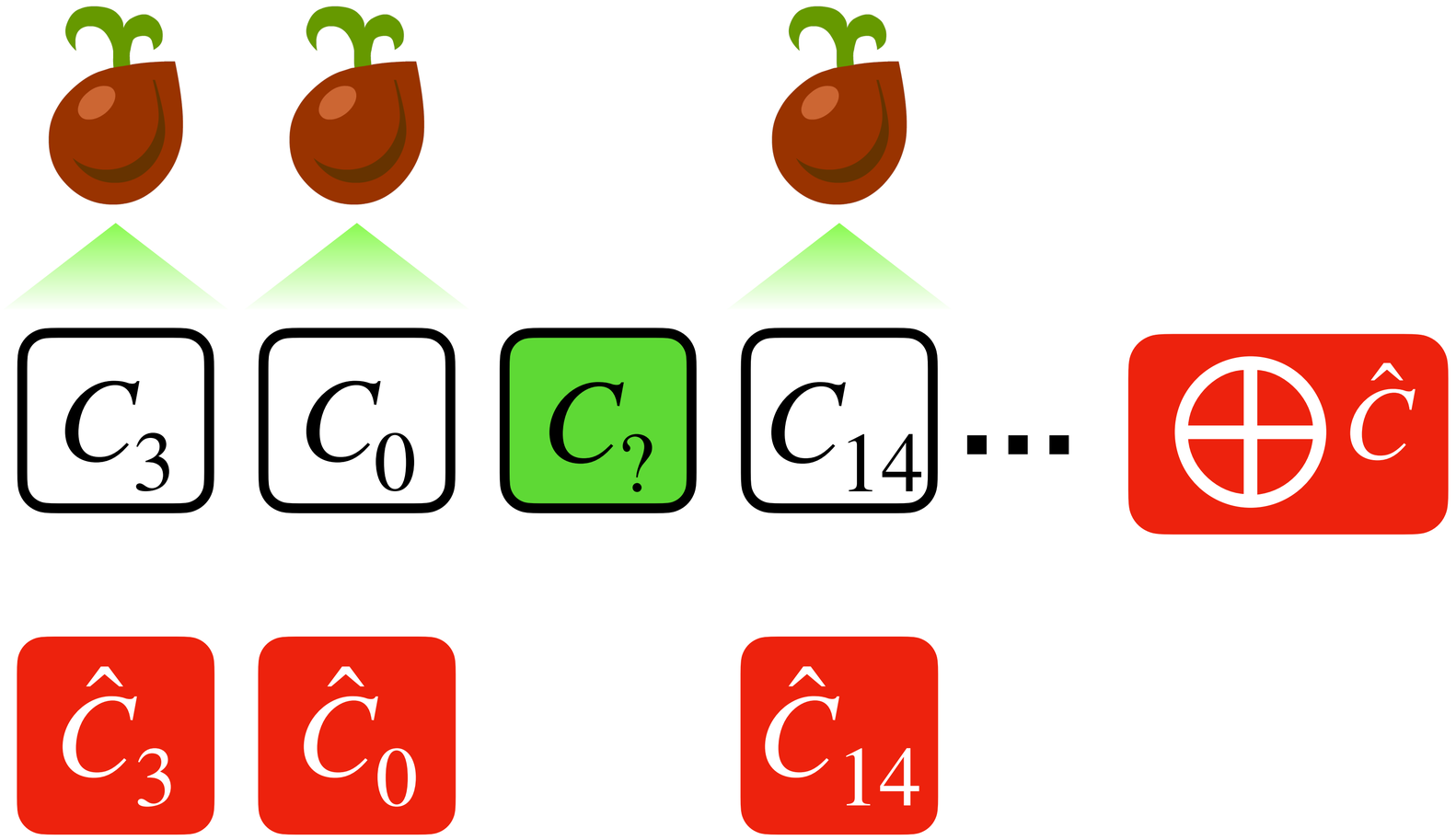
Evaluator



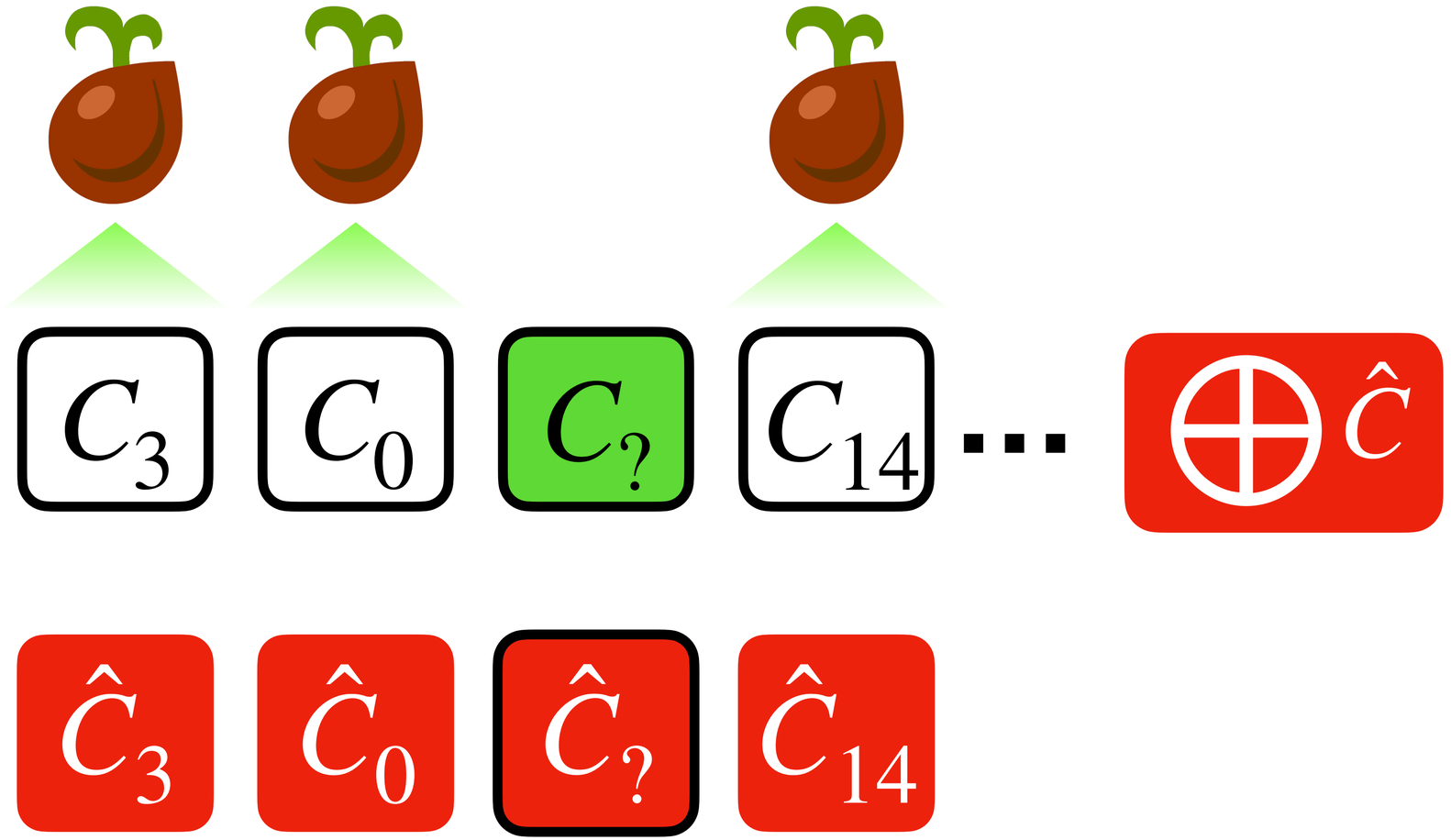
Evaluator



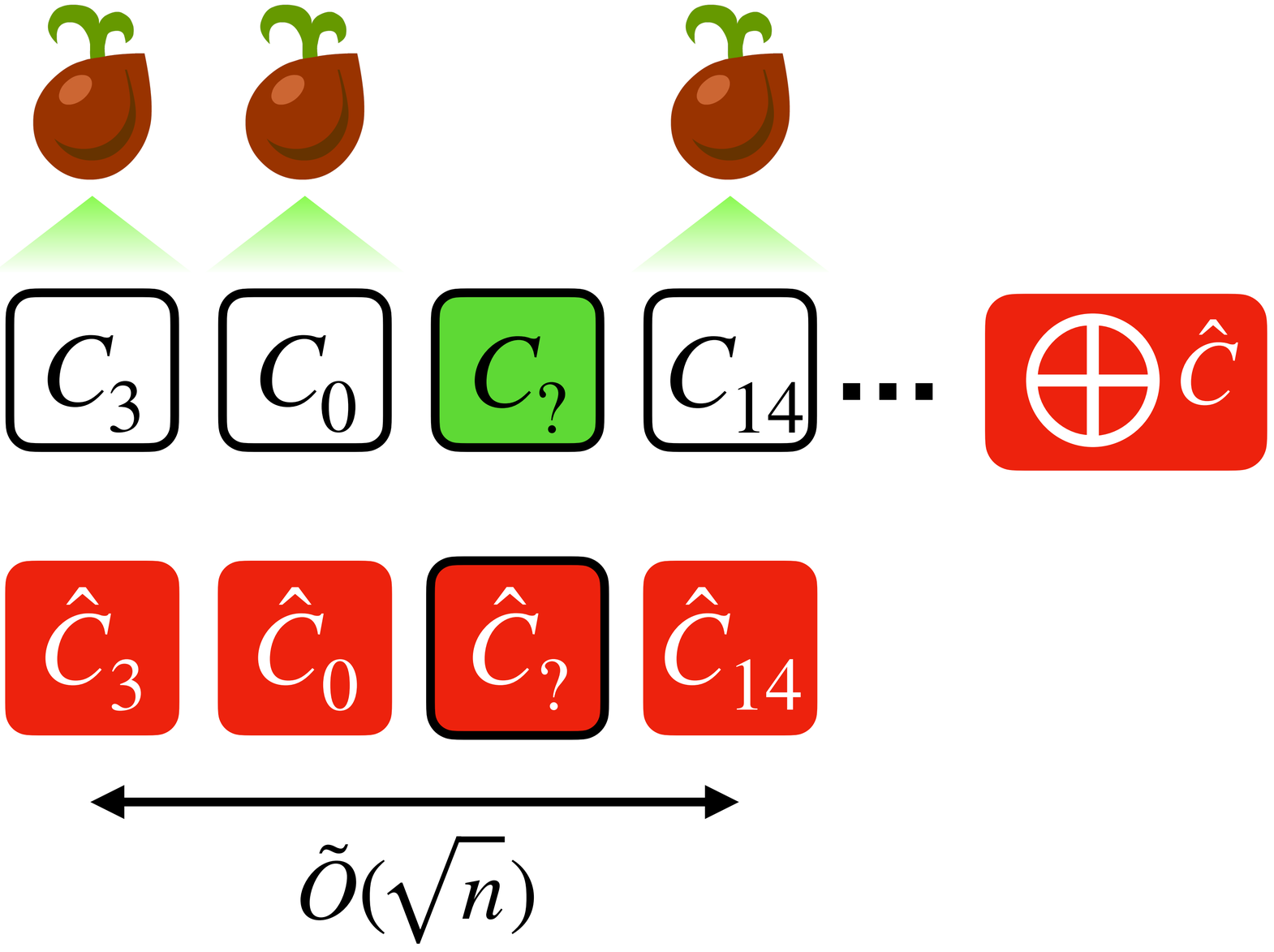
Evaluator



Evaluator



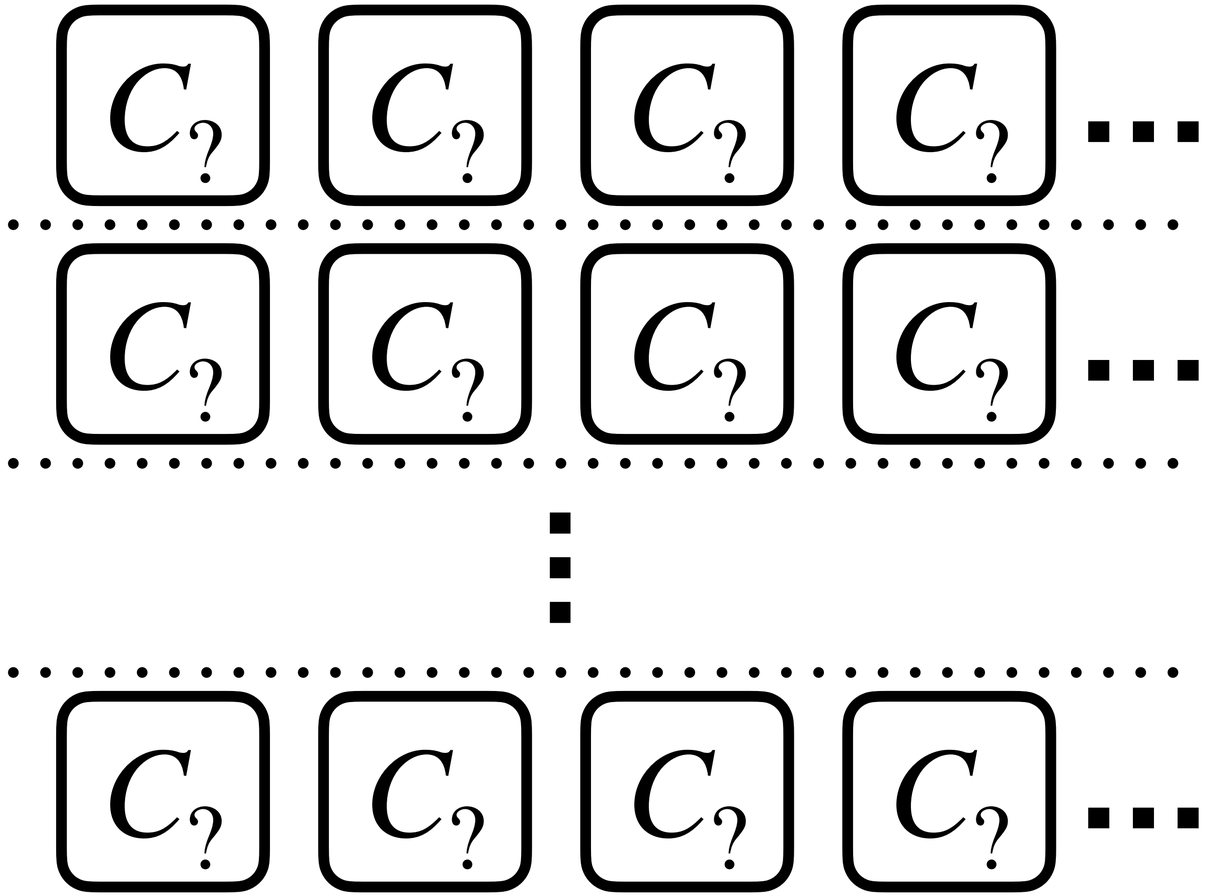
Evaluator



Evaluator

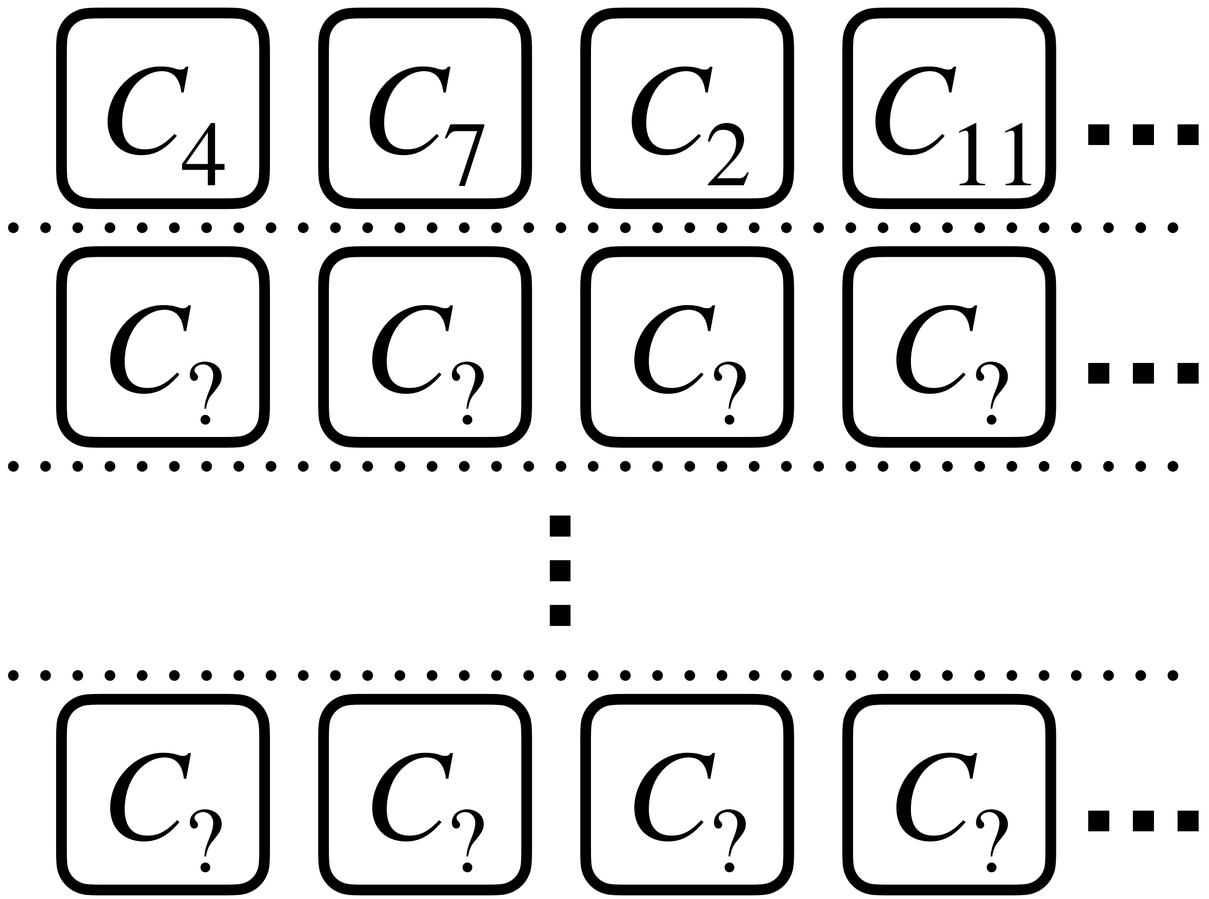
Surprising technical challenge:

Compactly identify and reveal to the Evaluator the active branch/siblings



Key Insight:

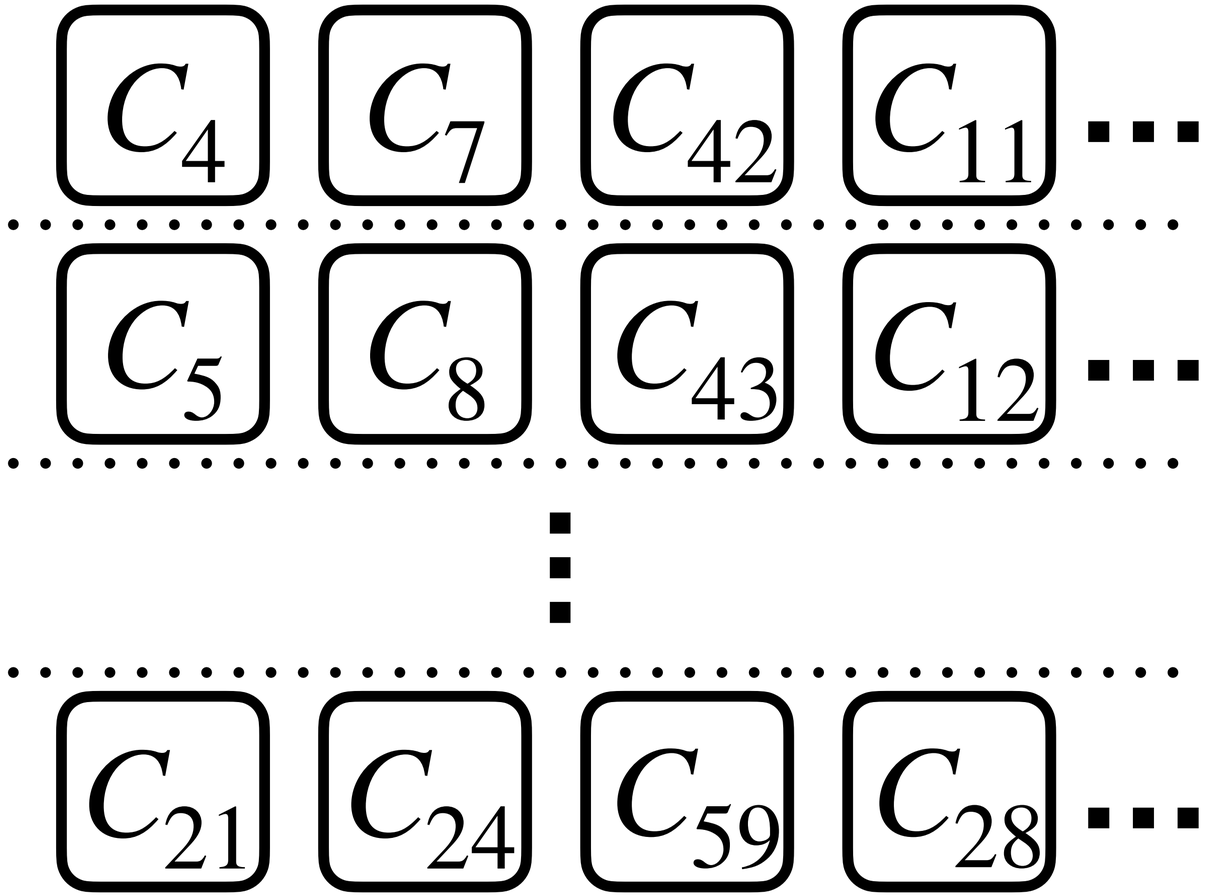
Pseudorandomly select the content of *one* bucket



Key Insight:

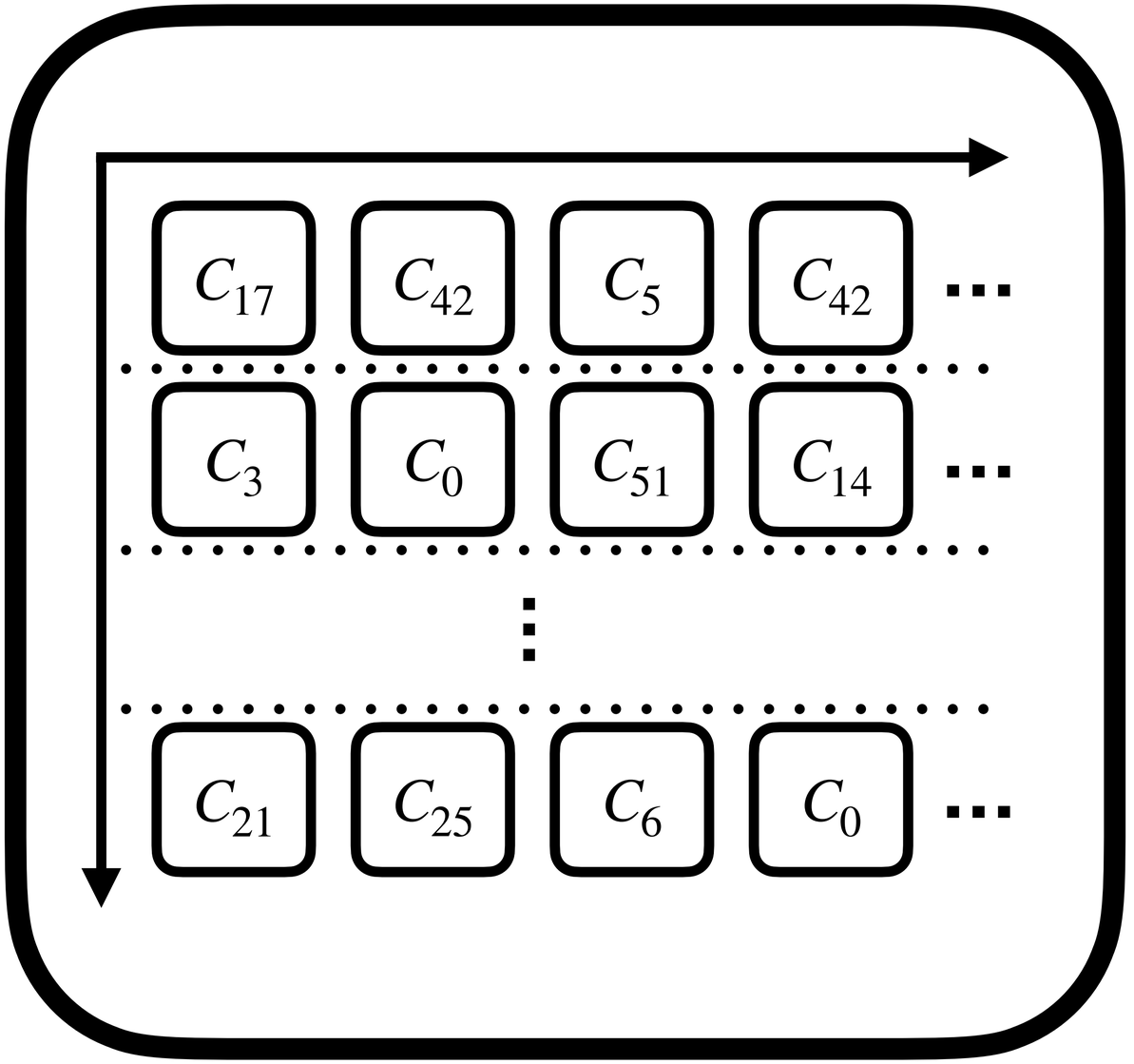
Pseudorandomly select the content of *one* bucket

Choose the content of the other buckets *based off this first bucket*

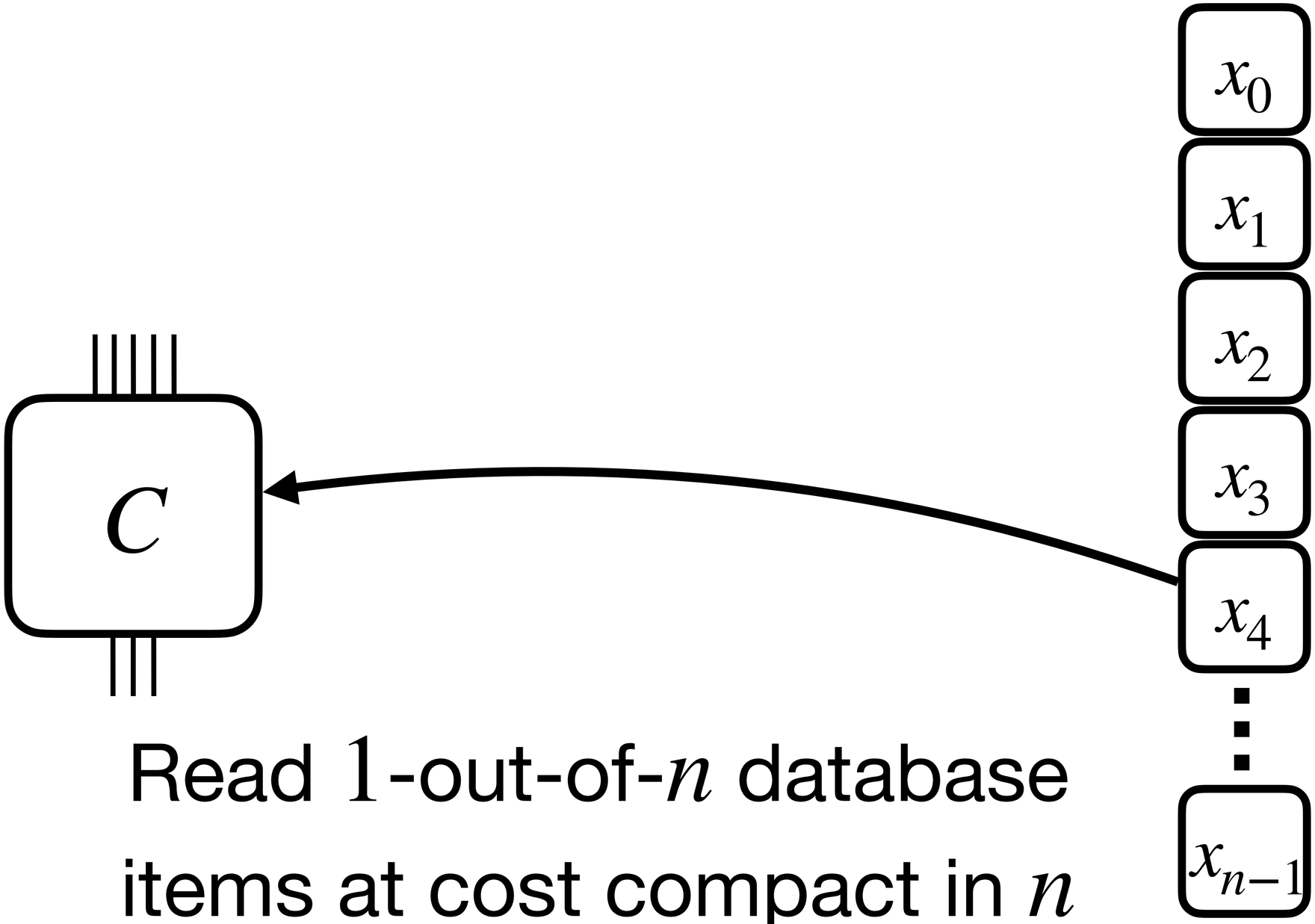


GCWise

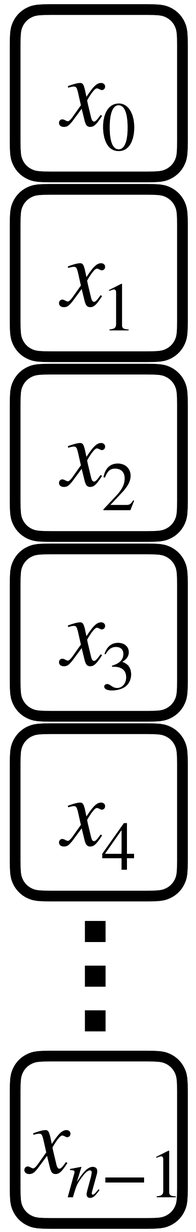
- Organize branches into $\tilde{O}(\sqrt{n})$ buckets
- For each bucket, stack the branches
- Evaluator considers only the single active bucket



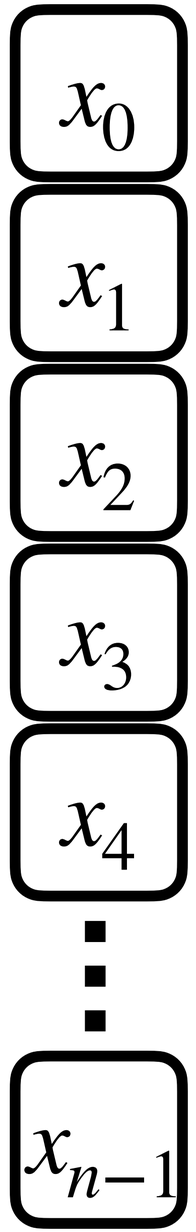
Garbled PIR

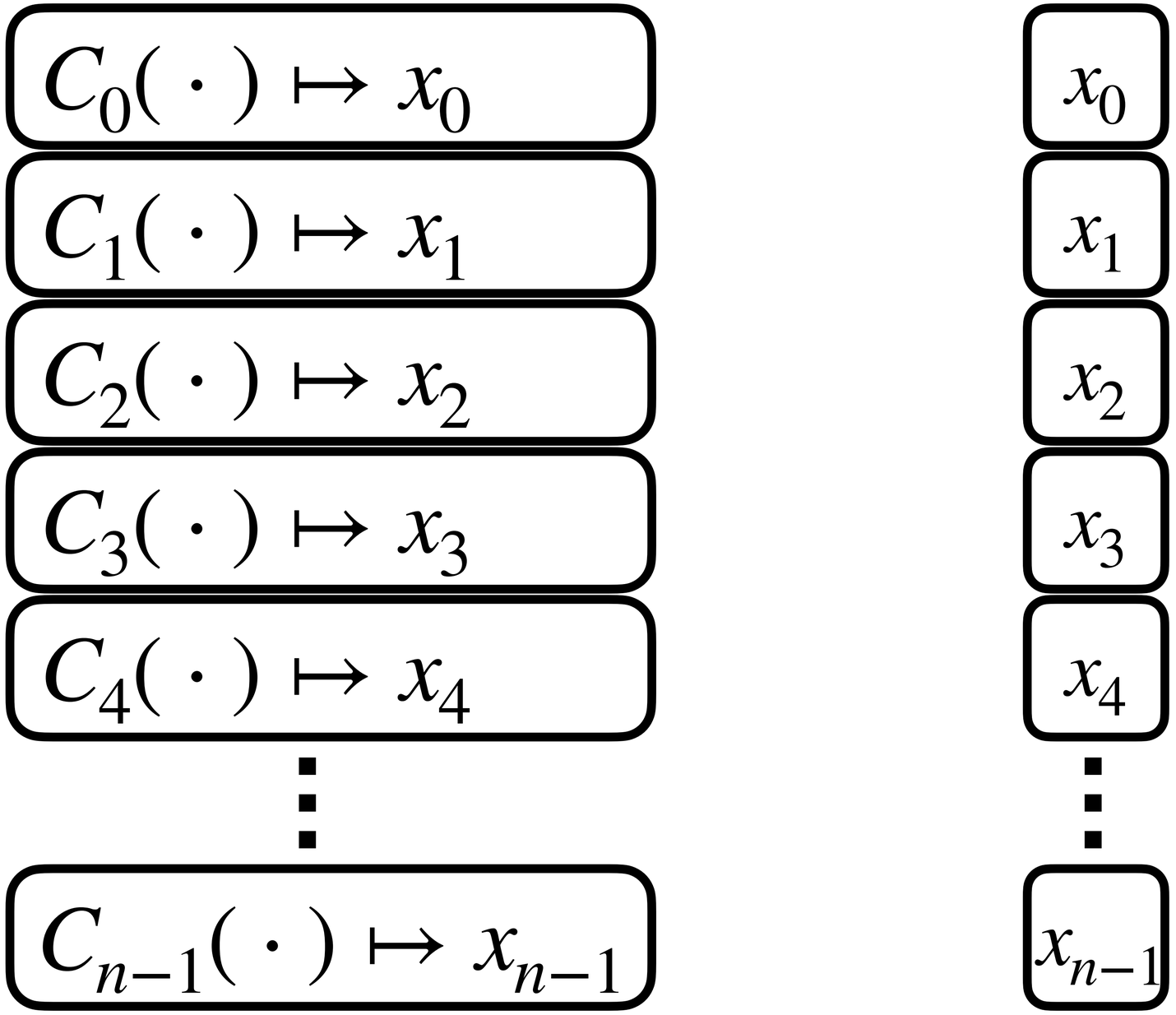


Read 1-out-of-n database items at cost compact in n


x_0
x_1
x_2
x_3
x_4
x_{n-1}

$$C_0(\,\cdot\,) \mapsto x_0$$




$C_0(\cdot) \mapsto x_0$
$C_1(\cdot) \mapsto x_1$
$C_2(\cdot) \mapsto x_2$
$C_3(\cdot) \mapsto x_3$
$C_4(\cdot) \mapsto x_4$
$C_{n-1}(\cdot) \mapsto x_{n-1}$
x_0
x_1
x_2
x_3
x_4
x_{n-1}

GCWise

- Compact 2PC for functions with conditionals

- Garbled PIR