Constrained Pseudorandom Functions From Homomorphic Secret Sharing

Geoffroy Couteau[1], Pierre Meyer[1,2], Alain Passelègue[3,4], and

Mahshid Riahinia[4]

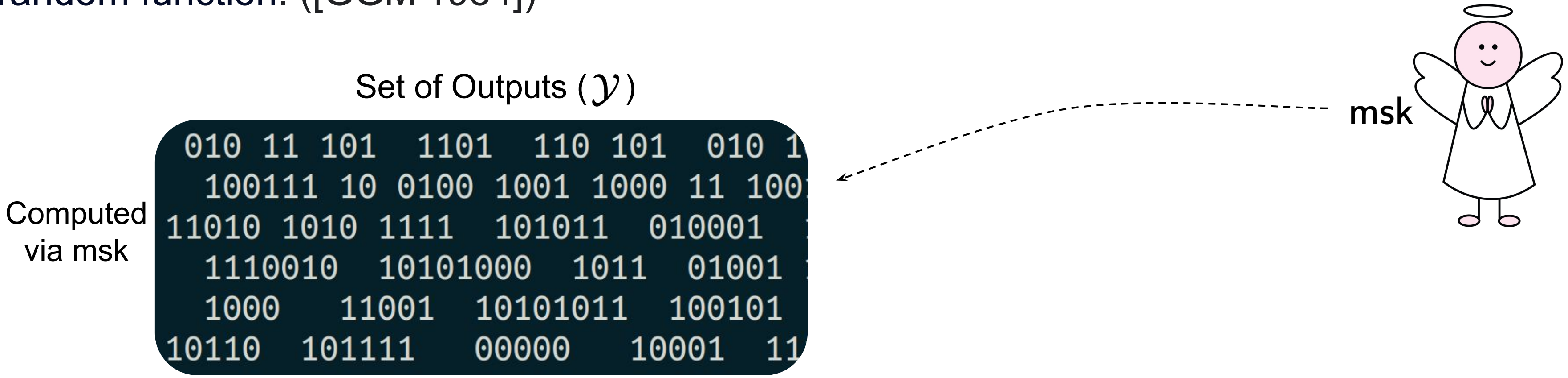
[1] Université Paris Cité, CNRS, IRIF, Paris, France.
[2] Reichman University, Herzliya, Israel.
[3] Inria, France.
[4] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France.

EUROCRYPT, 2023

Constrained Pseudorandom Function

Pseudorandom function: $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Definition. A deterministic keyed function that is computationally indistinguishable from a truly random function. ([GGM'1984])



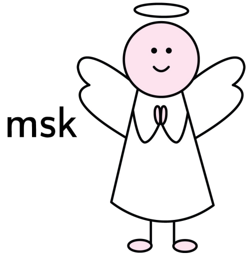
Constrained Pseudorandom Function

Pseudorandom function: $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Definition. A deterministic keyed function that is computationally indistinguishable from a truly random function. ([GGM'1984])

Set of Outputs ($\mathcal{Y}$)

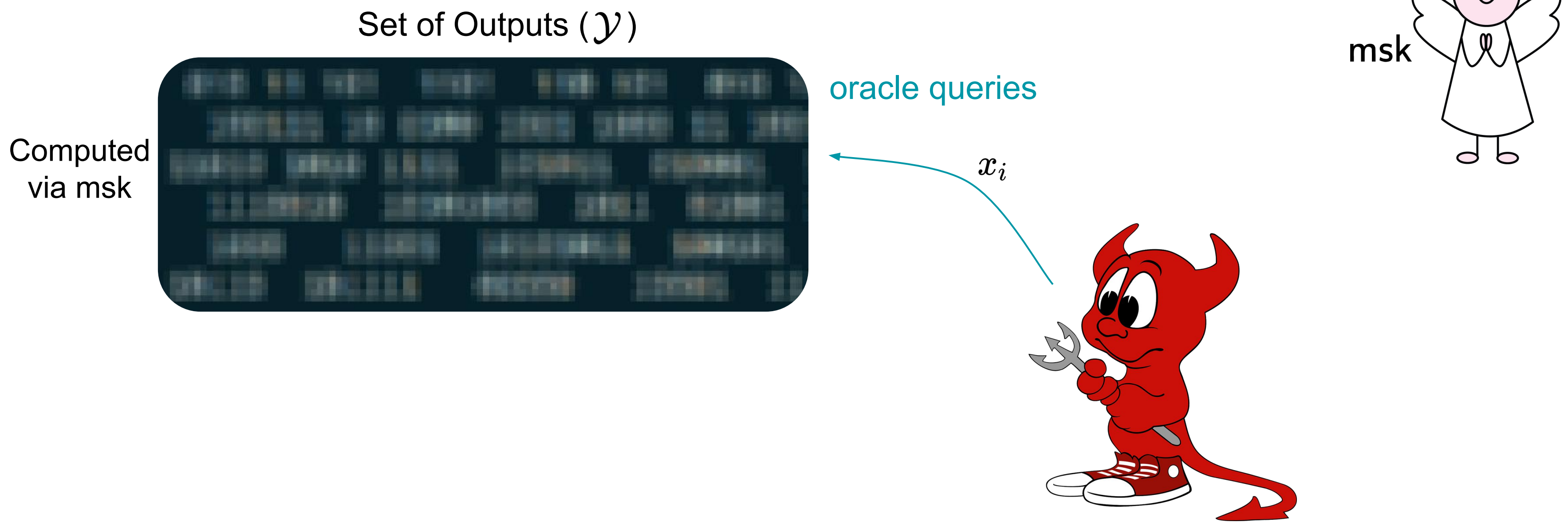
Computed via msk



Constrained Pseudorandom Function

Pseudorandom function: $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

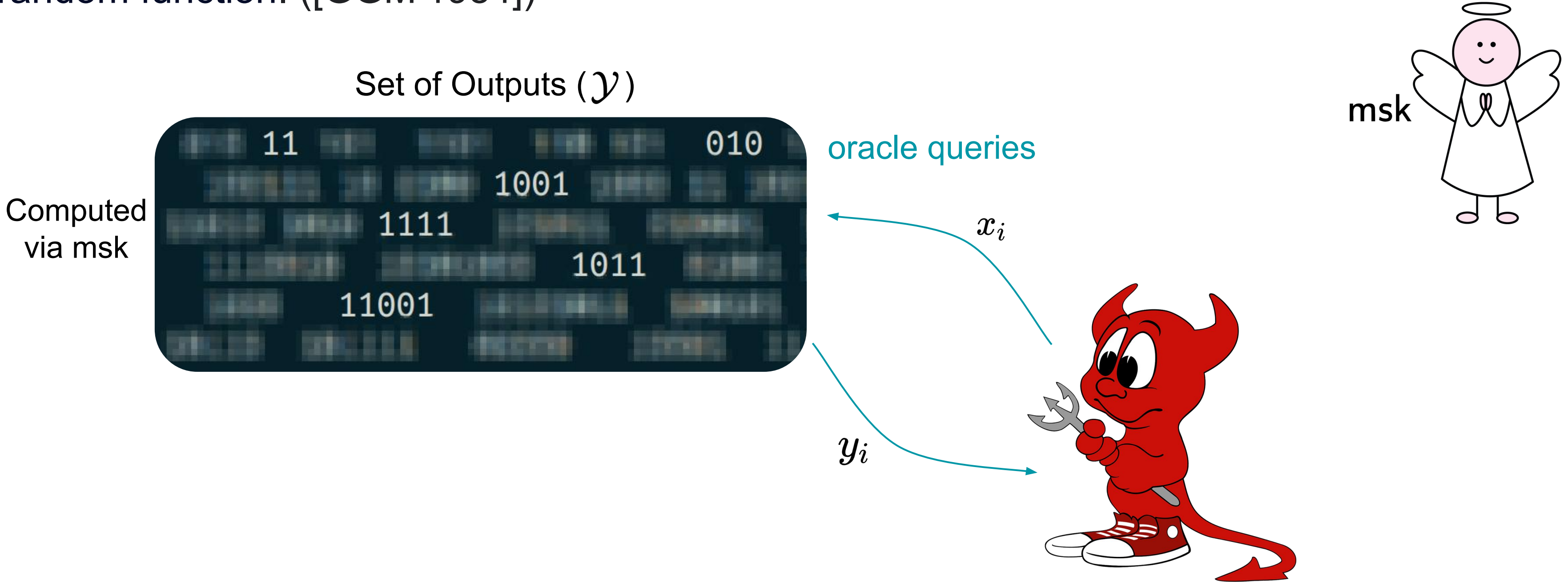
Definition. A deterministic keyed function that is computationally indistinguishable from a truly random function. ([GGM'1984])



Constrained Pseudorandom Function

Pseudorandom function: $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

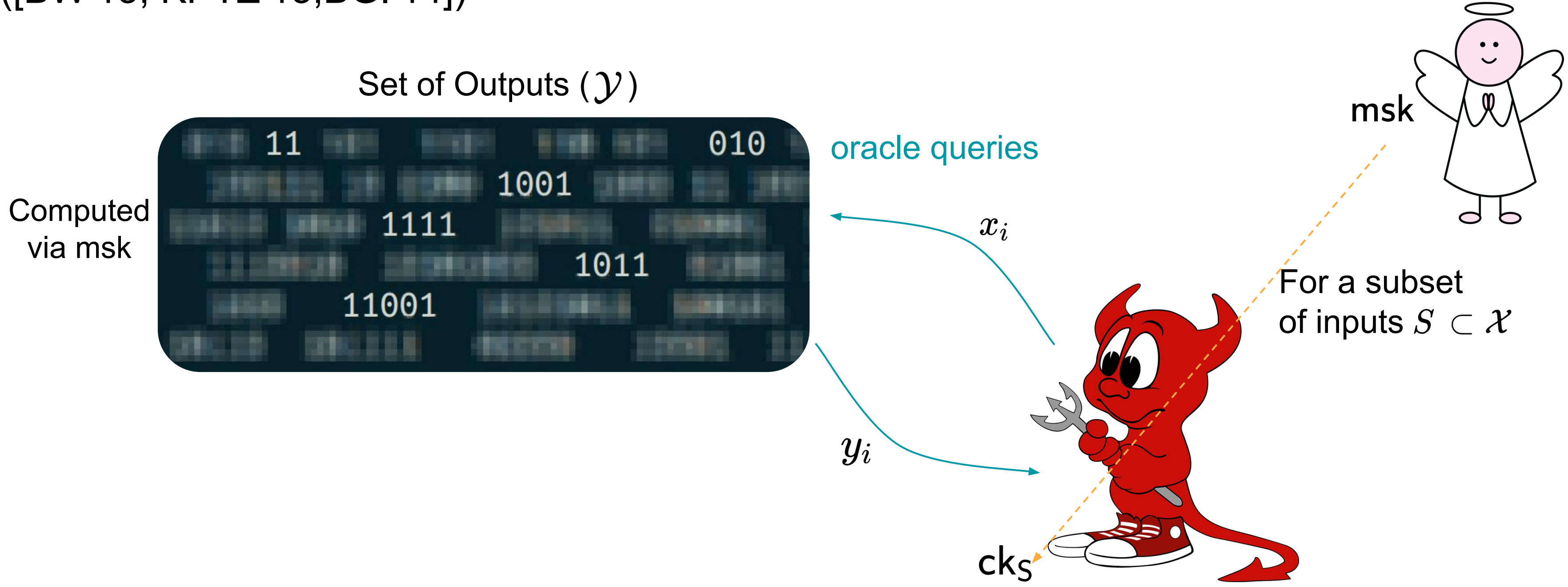
Definition. A deterministic keyed function that is computationally indistinguishable from a truly random function. ([GGM'1984])



Constrained Pseudorandom Function

Constrained *Pseudorandom function:* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

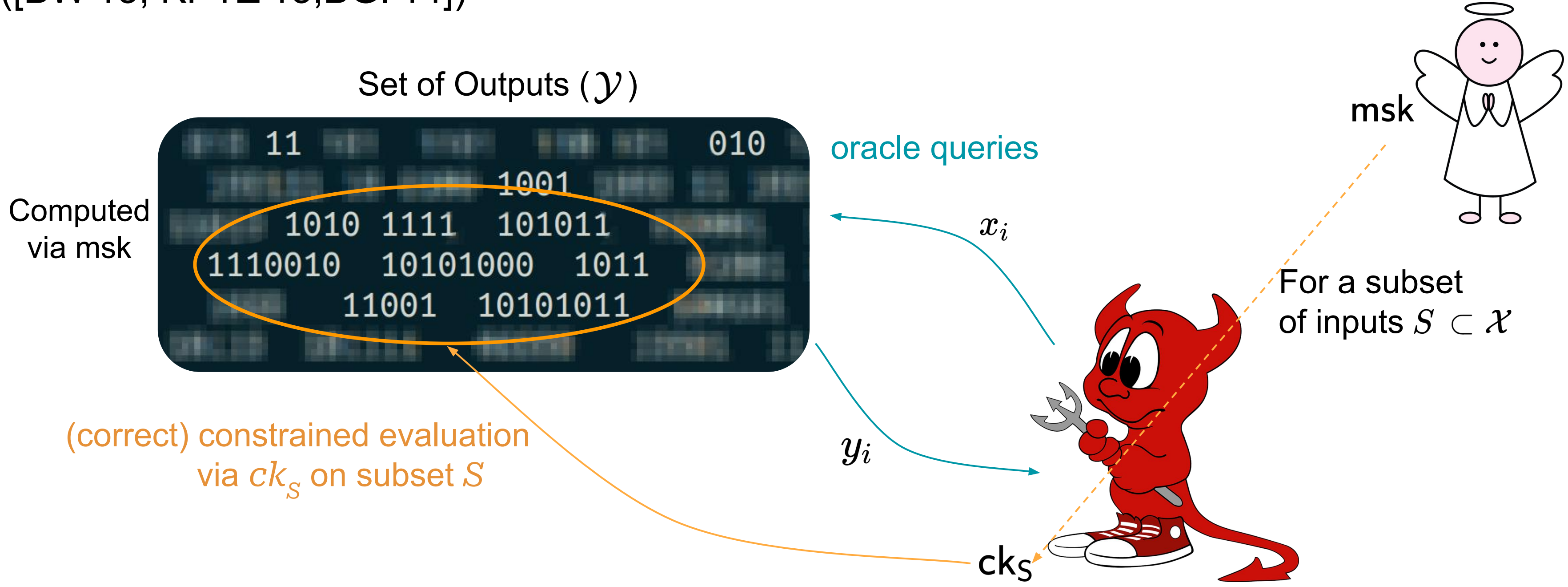
Definition. A pseudorandom function with constrained access to the evaluation. ([BW'13, KPTZ'13,BGI'14])



Constrained Pseudorandom Function

Constrained *Pseudorandom function:* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

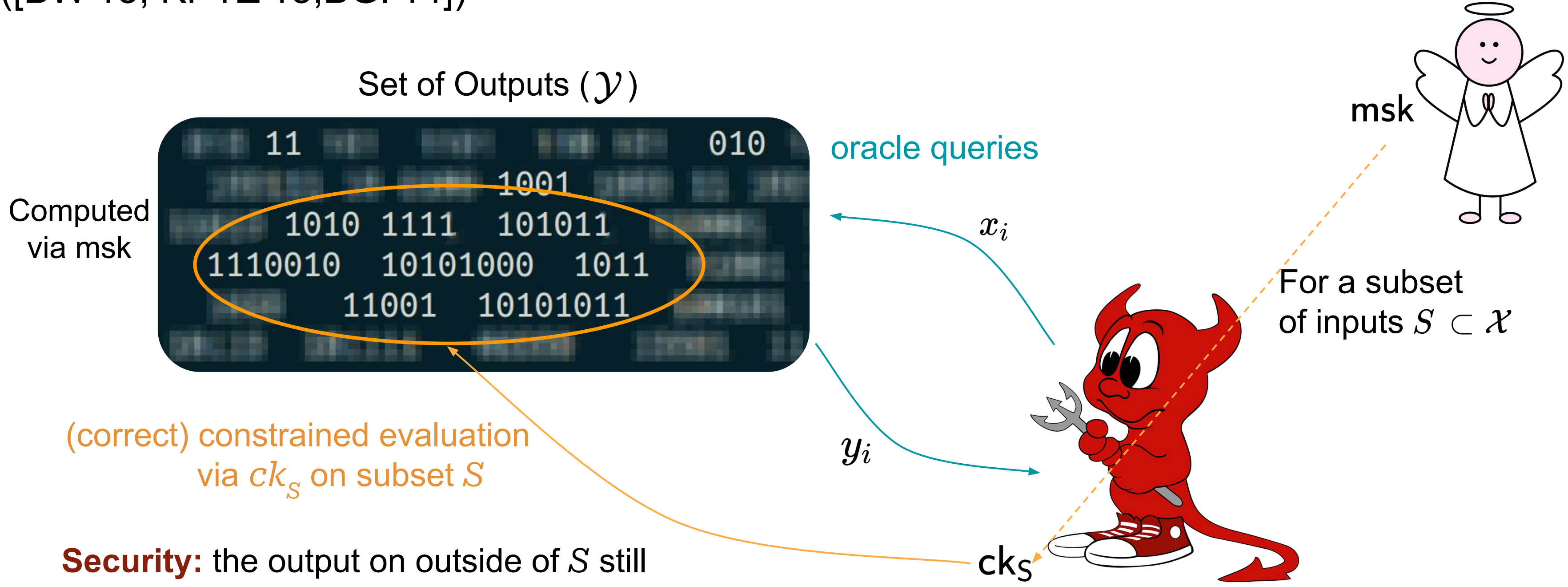
Definition. A pseudorandom function with constrained access to the evaluation. ([BW'13, KPTZ'13,BGI'14])


Set of Outputs ($\mathcal{Y}$)
Computed via msk
11
010
1001
1010 1111 101011
1110010 10101000 1011
11001 10101011
oracle queries
x_i
y_i
msk
For a subset of inputs $S \subset \mathcal{X}$
(correct) constrained evaluation via ck_S on subset S
ck_S

Constrained Pseudorandom Function

Constrained *Pseudorandom function:* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Definition. A pseudorandom function with constrained access to the evaluation. ([BW'13, KPTZ'13,BGI'14])

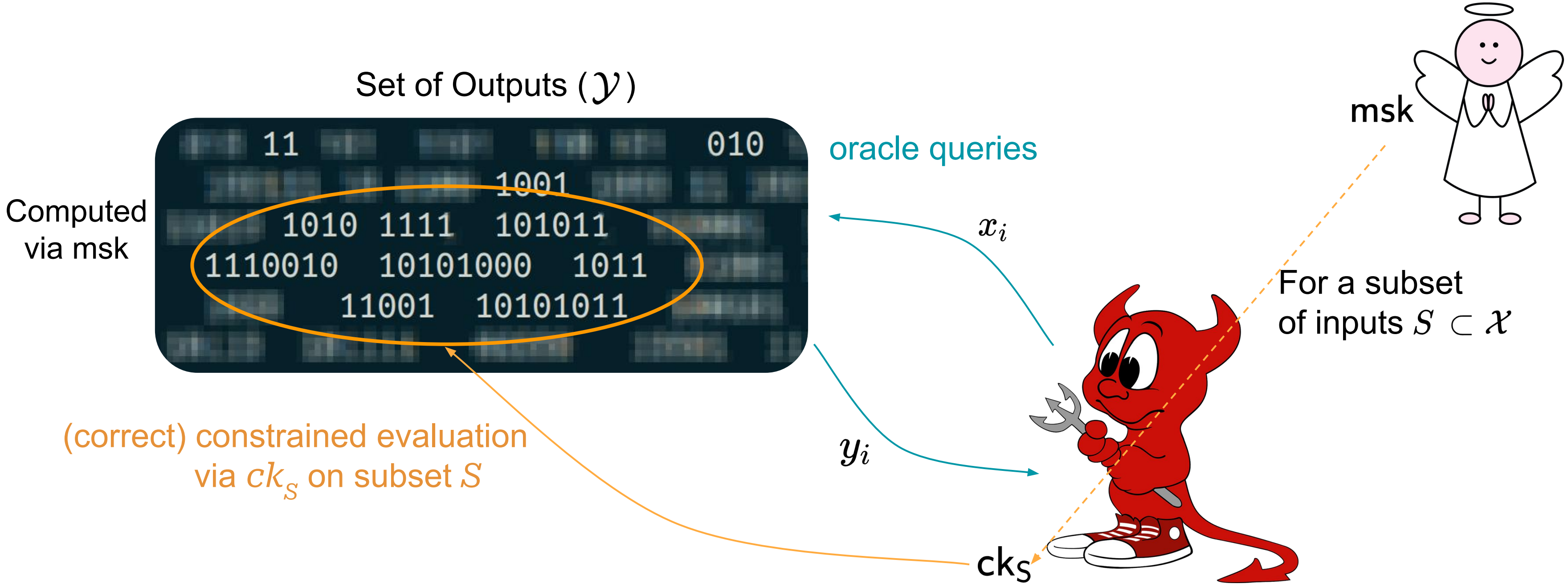


Set of Outputs ($\mathcal{Y}$)
Computed via msk
11
010
1001
1010 1111 101011
1110010 10101000 1011
11001 10101011
oracle queries
x_i
y_i
msk
For a subset of inputs $S \subset \mathcal{X}$
(correct) constrained evaluation via ck_S on subset S
ck_S

Security: the output on outside of S still looks random to the adversary.

Constrained Pseudorandom Function

Subset S is defined via a predicate $C : \mathcal{X} \to \{0,1\}$; $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

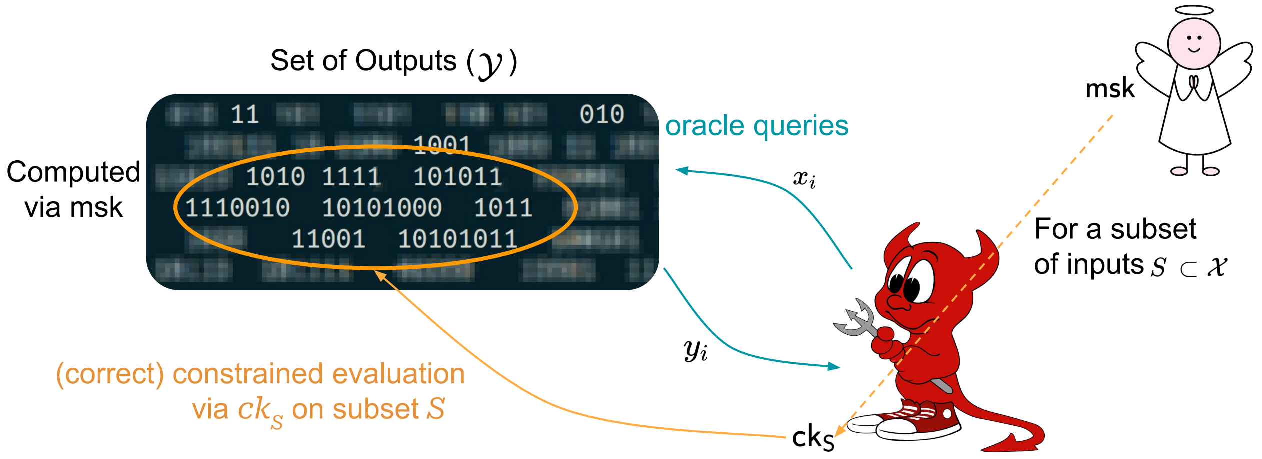


Set of Outputs ($\mathcal{Y}$)
Computed via msk
11
010
1001
1010 1111 101011
1110010 10101000 1011
11001 10101011
oracle queries
x_i
y_i
msk
For a subset of inputs $S \subset \mathcal{X}$
(correct) constrained evaluation via ck_S on subset S
ck_S

Our contributions

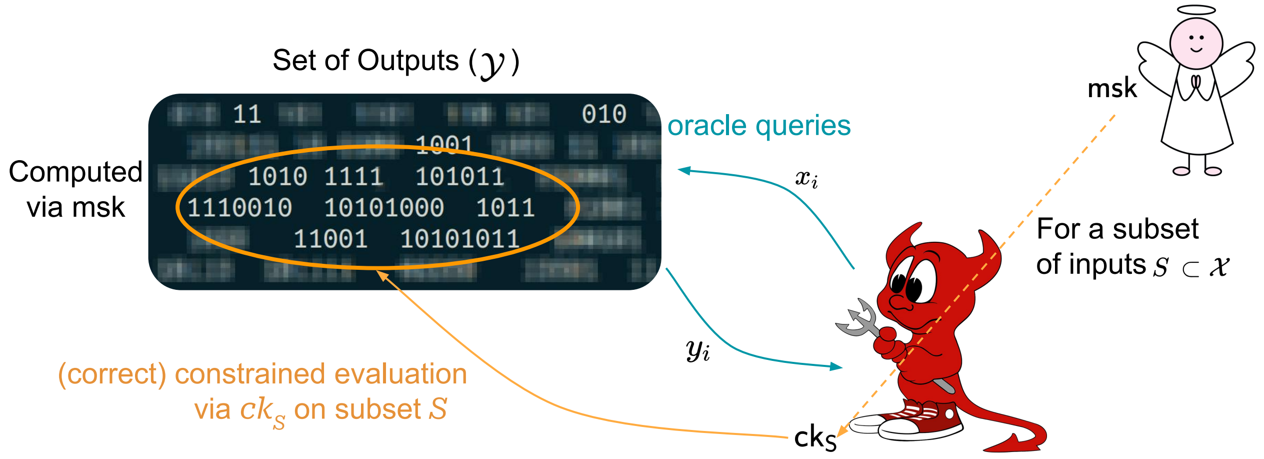
1-key (selectively-secure) constrained PRF

for inner-product and NC^1 predicates.


Set of Outputs ($\mathcal{Y}$)
11
010
1001
1010 1111 101011
1110010 10101000 1011
11001 10101011
Computed via msk
oracle queries
x_i
y_i
msk
For a subset of inputs $S \subset \mathcal{X}$
(correct) constrained evaluation via ck_S on subset S
ck_S

Our contributions

1-key (selectively-secure) constrained PRF for inner-product and NC^1 predicates.


Set of Outputs ($\mathcal{Y}$)
11
010
1001
1010 1111 101011
1110010 10101000 1011
11001 10101011
Computed via msk
oracle queries
x_i
y_i
msk
For a subset of inputs $S \subset \mathcal{X}$
ck_S
(correct) constrained evaluation via ck_S on subset S

+ MPC Applications

Homomorphic Secret Sharing

Definition. Protocol for performing distributed evaluation on a secret. ([BGI'16])

Program $P \in \mathcal{P}$

$\boxed{\text{Goal: Evaluate } P(s).}$

s

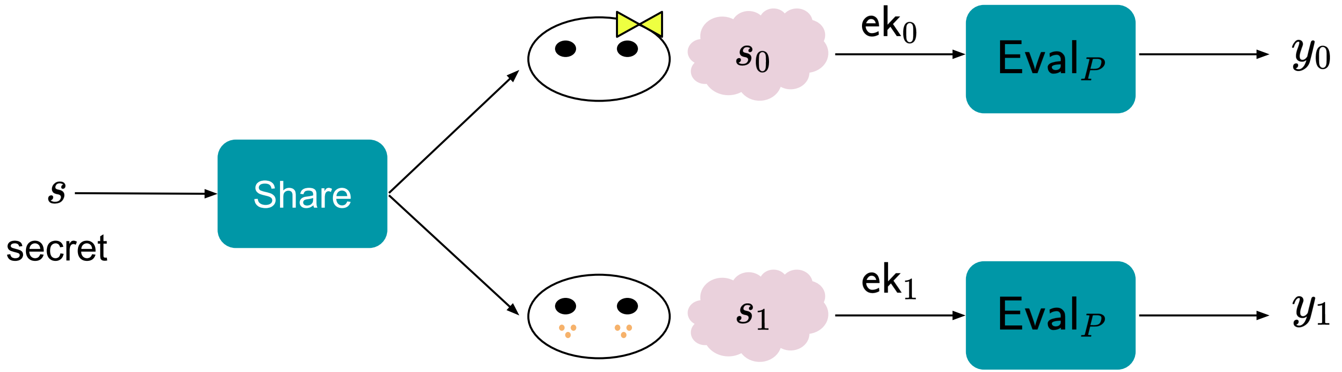
secret

Homomorphic Secret Sharing

Definition. Protocol for performing distributed evaluation on a secret. ([BGI'16])

Program $P \in \mathcal{P}$

$\boxed{\text{Goal: Evaluate } P(s).}$

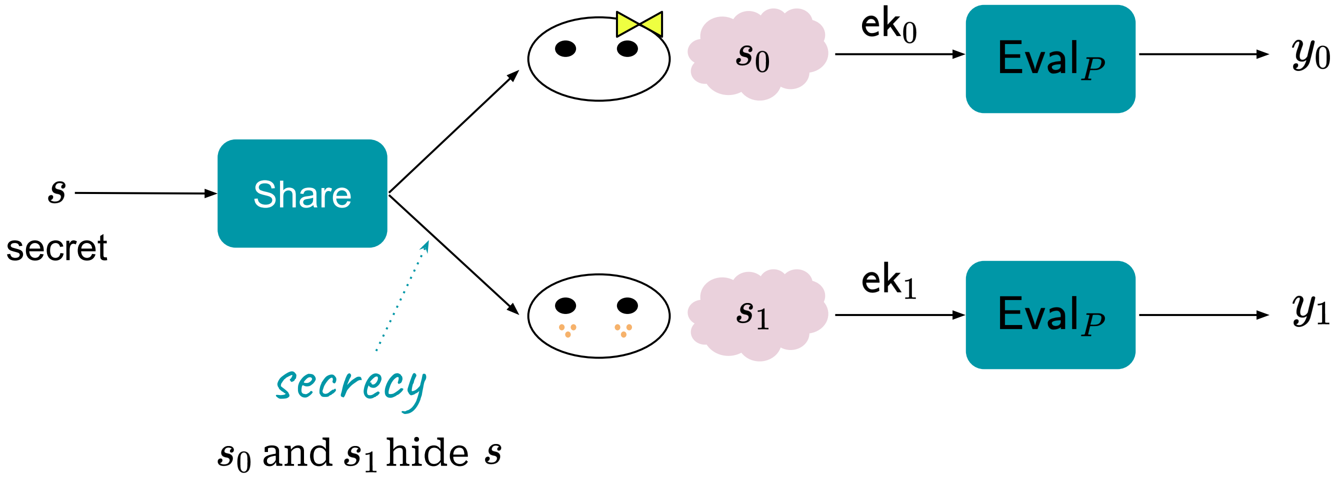


Homomorphic Secret Sharing

Definition. Protocol for performing distributed evaluation on a secret. ([BGI'16])

Program $P \in \mathcal{P}$

$\boxed{\text{Goal: Evaluate } P(s).}$

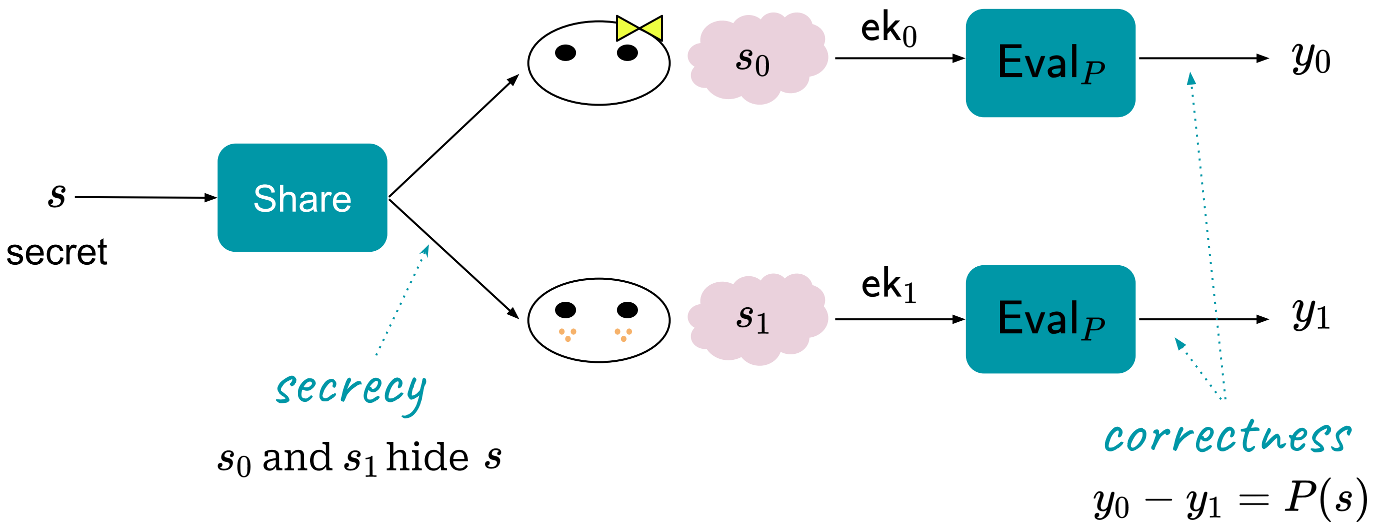


Homomorphic Secret Sharing

Definition. Protocol for performing distributed evaluation on a secret. ([BGI'16])

Program $P \in \mathcal{P}$

$\boxed{\text{Goal: Evaluate } P(s).}$



Our contributions

1-key constrained PRF for inner-product and NC^1 predicates from homomorphic secret sharing.

- Extending homomorphic secret sharing properties.
- (most of) Existing HSS schemes satisfy these properties.

  $\rightsquigarrow$ new constructions of constrained PRF.

- Revisiting Applications of HSS to Secure Computation.
  - Secure computation with silent preprocessing, and
  - Secure computation with sublinear communication.

Our contributions

1-key constrained PRF for inner-product and NC^1 predicates from homomorphic secret sharing.

- Extending homomorphic secret sharing properties.
- (most of) Existing HSS schemes satisfy these properties.

  $\rightsquigarrow$ new constructions of constrained PRF.

- Revisiting Applications of HSS to Secure Computation.
  - Secure computation with silent preprocessing, and
  - Secure computation with sublinear communication.

*(1) **D**ecisional **C**omposite **R**esiduosity*

*(2) **LWE** with superpolynomial modulus*

*(3) Hardness of the **Joye-Libert** encryption scheme*

*(4) **DDH & DXDH** over class groups*

*(5) **H**ard **M**embership **S**ubgroup over class groups*

Constrained PRF from Homomorphic Secret Sharing

(general strategy)

Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Constrained PRF from Homomorphic Secret Sharing

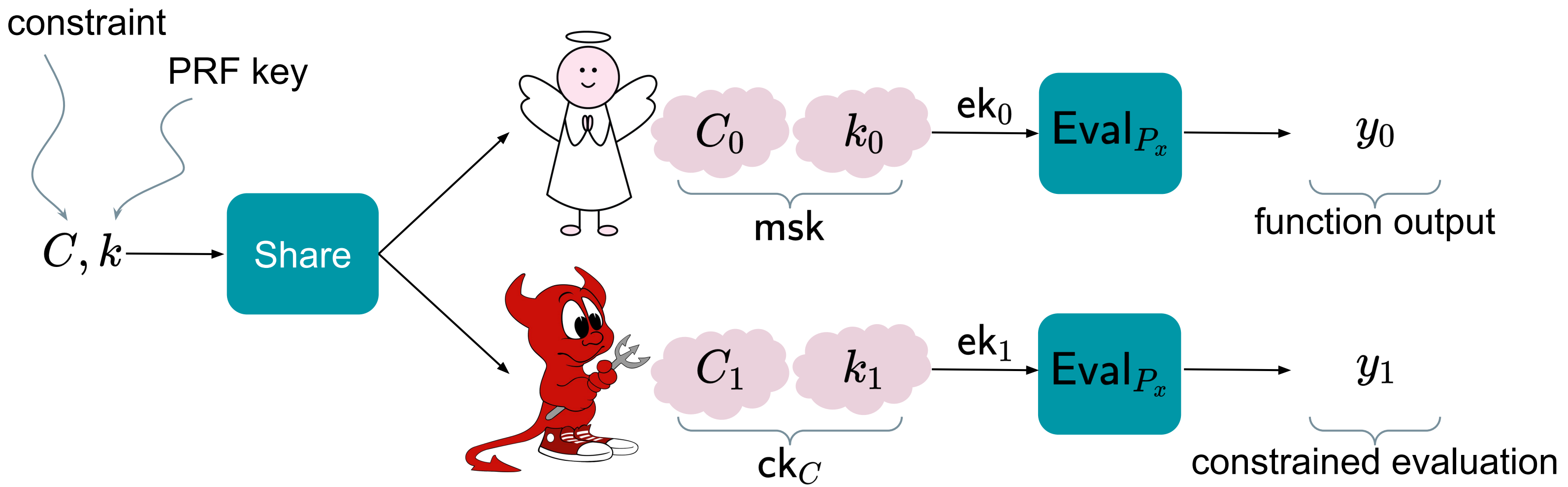
For a constraint C: 𝓧 -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.

Constrained PRF from Homomorphic Secret Sharing

For a constraint $C: \mathcal{X} \to \{0,1\}$: $S_C = \{x \in \mathcal{X} : C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

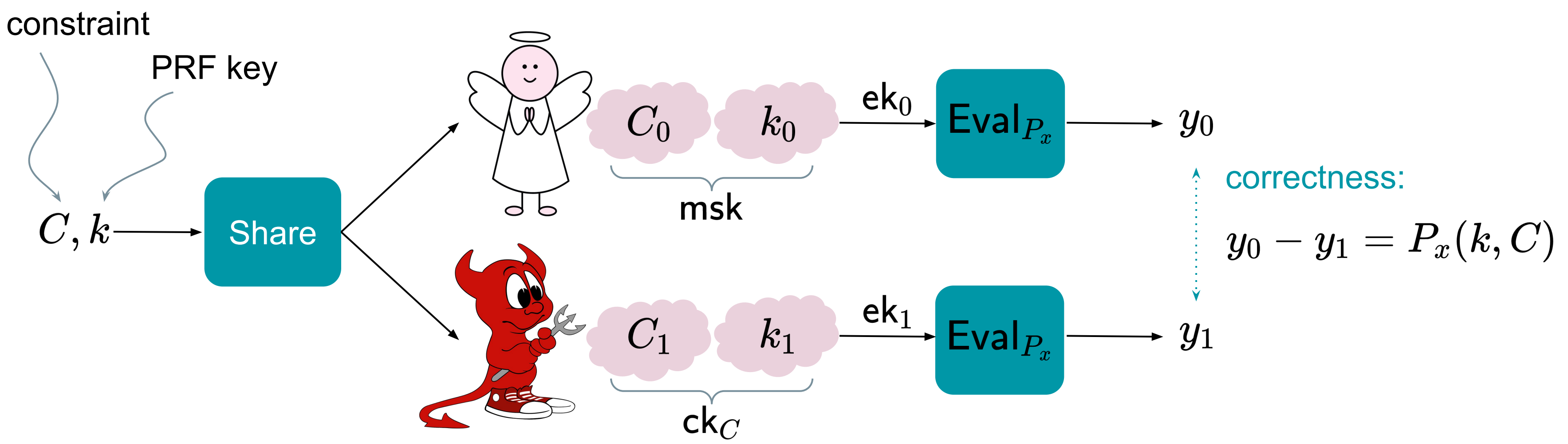
Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.



Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.

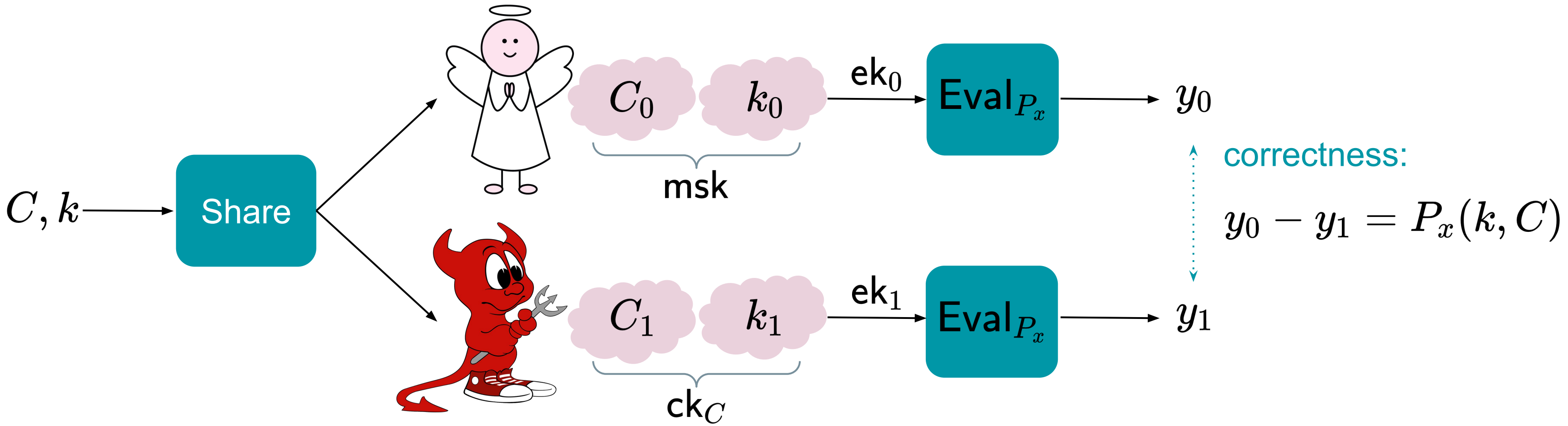


Constrained PRF from Homomorphic Secret Sharing

For a constraint $C: \mathcal{X} \to \{0,1\}$: $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

$\rightarrow x \in S_C \Rightarrow P_x(k, C) = 0$

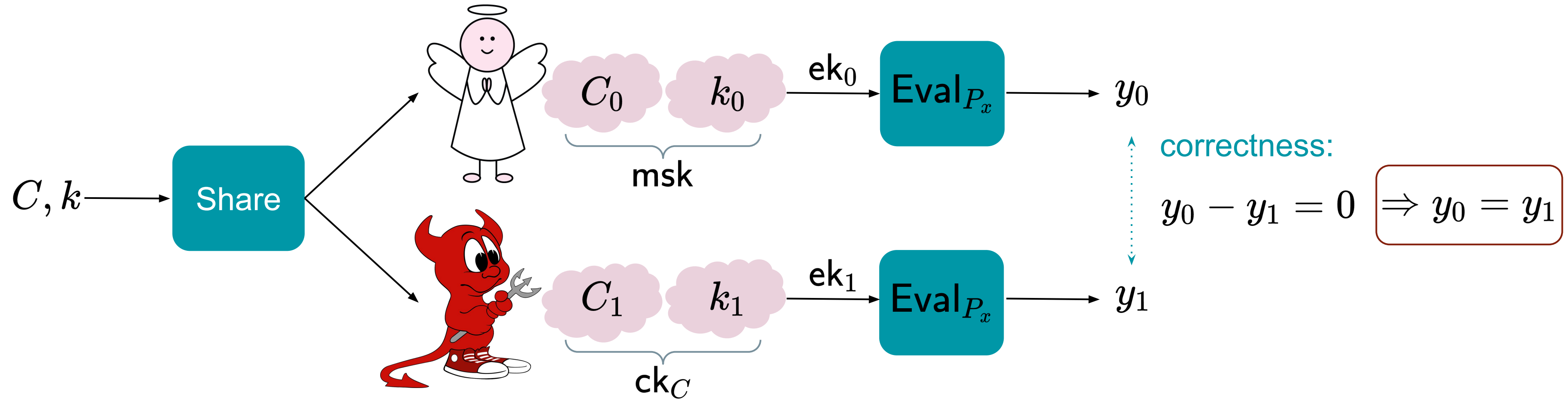


Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

$\rightarrow x \in S_C \Rightarrow P_x(k, C) = 0$



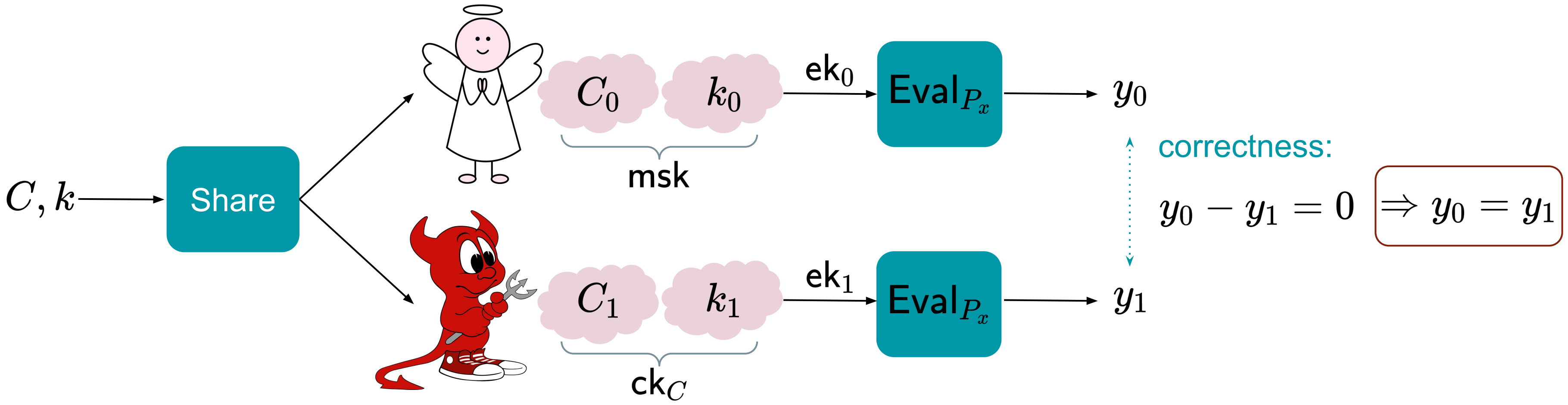
Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

idea

➡ $x \in S_C \Rightarrow P_x(k, C) = 0$ ⤳ Equal outputs



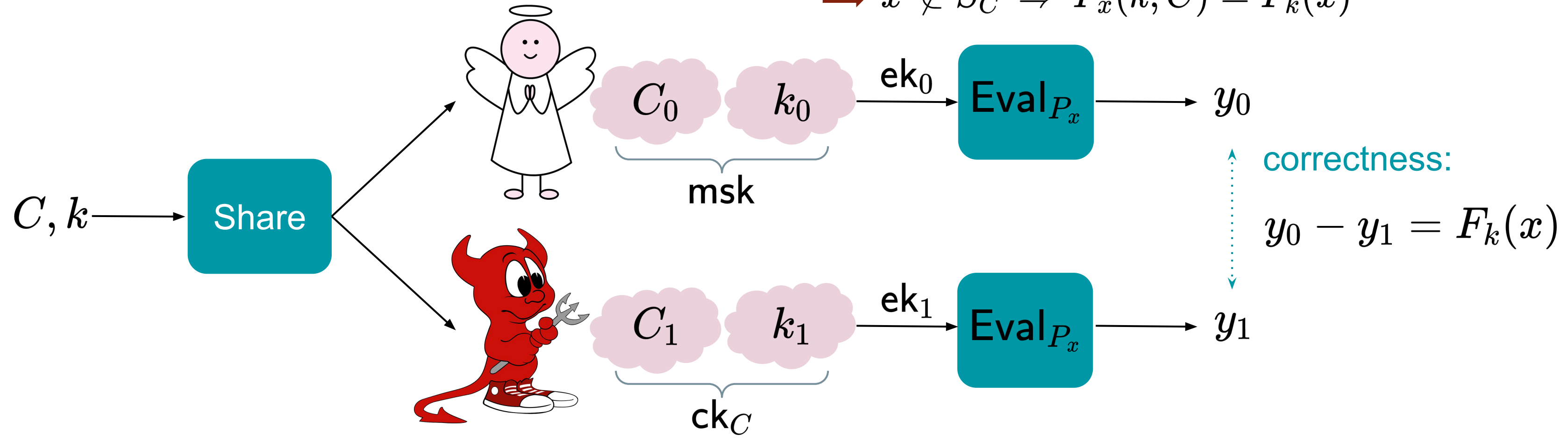
Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

$x \in S_C \Rightarrow P_x(k, C) = 0$ ⟶ Equal outputs

➡ $x \notin S_C \Rightarrow P_x(k, C) = F_k(x)$



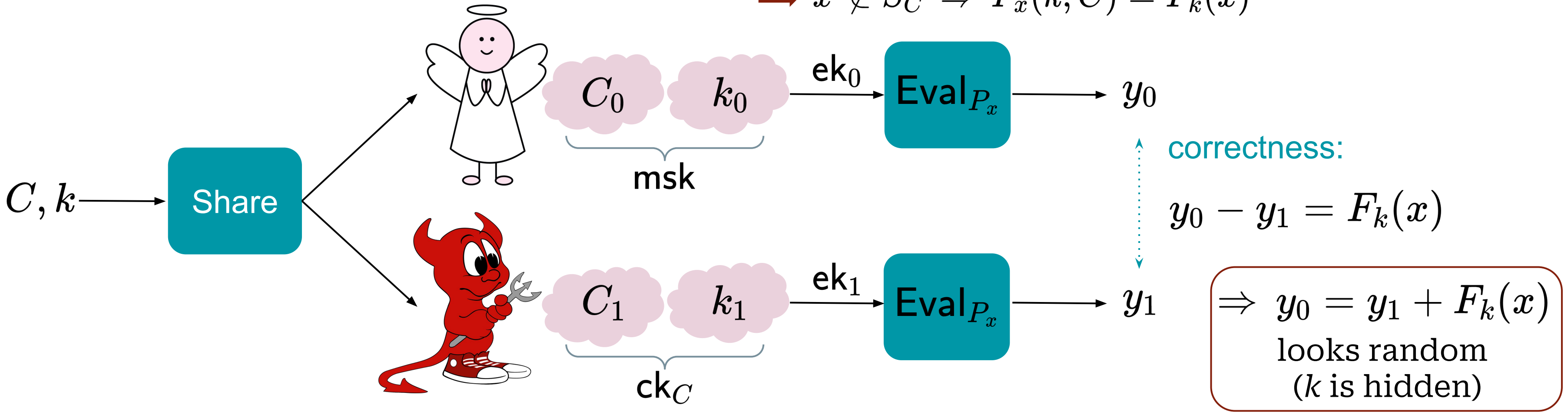
Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

$x \in S_C \Rightarrow P_x(k, C) = 0$ ⟿ Equal outputs

➡ $x \notin S_C \Rightarrow P_x(k, C) = F_k(x)$



$\Rightarrow y_0 = y_1 + F_k(x)$
looks random
(k is hidden)

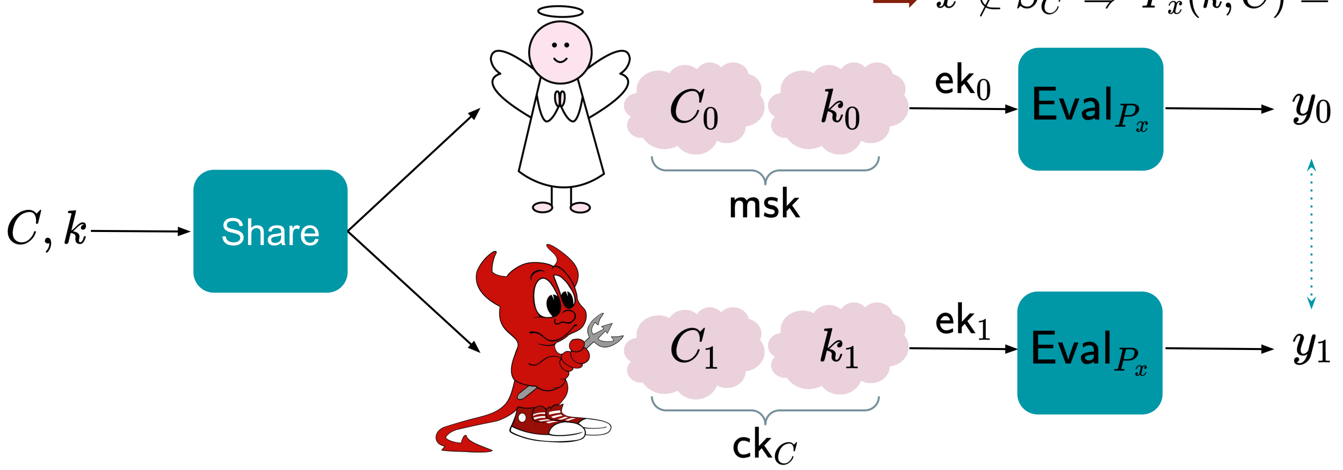
Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $\underbrace{P_x : (k, C) \mapsto C(x) \cdot F_k(x)}$.

$x \in S_C \Rightarrow P_x(k, C) = 0$ ⟶ Equal outputs

➡ $x \notin S_C \Rightarrow P_x(k, C) = F_k(x)$ ⟶ Random-looking function output



correctness:

$y_0 - y_1 = F_k(x)$

$\Rightarrow \; y_0 = y_1 + F_k(x)$
looks random
(k is hidden)

Constrained PRF from Homomorphic Secret Sharing

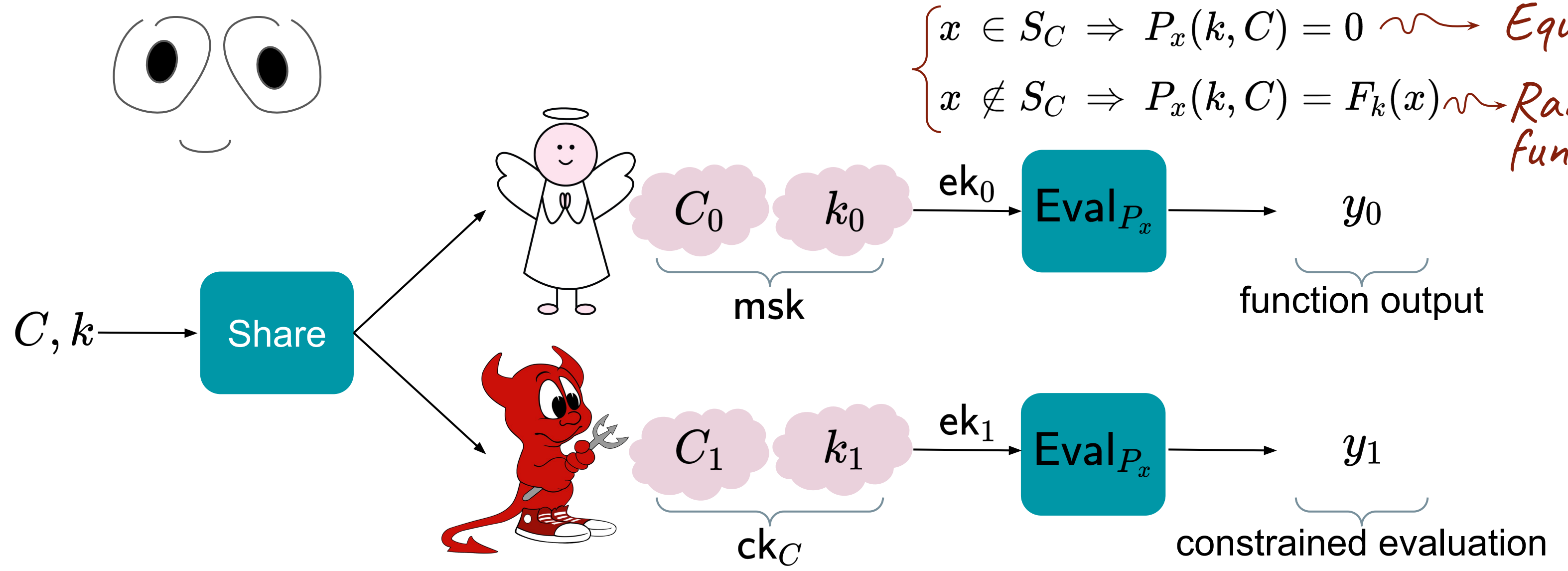
For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it. ✓

Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.

$$\begin{cases} x \in S_C \Rightarrow P_x(k,C) = 0 \\ x \notin S_C \Rightarrow P_x(k,C) = F_k(x) \end{cases}$$

Equal outputs

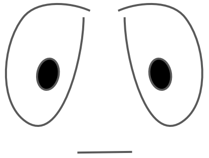
Random-looking function output



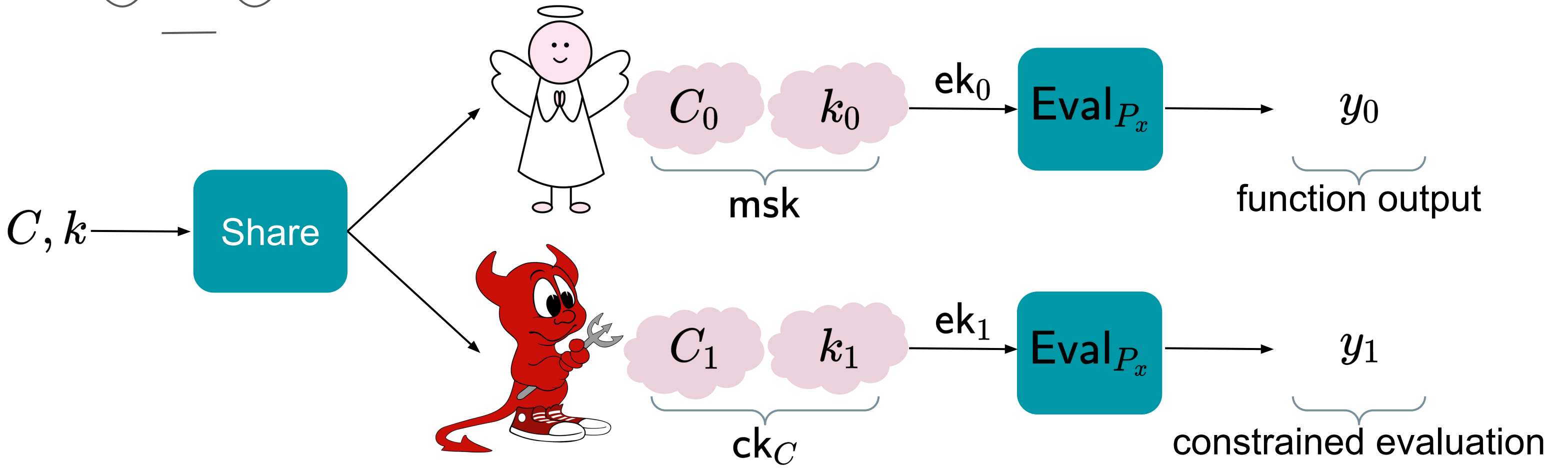
Constrained PRF from Homomorphic Secret Sharing

For a constraint C: $\mathcal{X}$ -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.



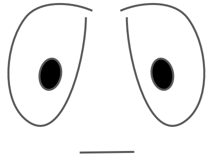
msk depends on C



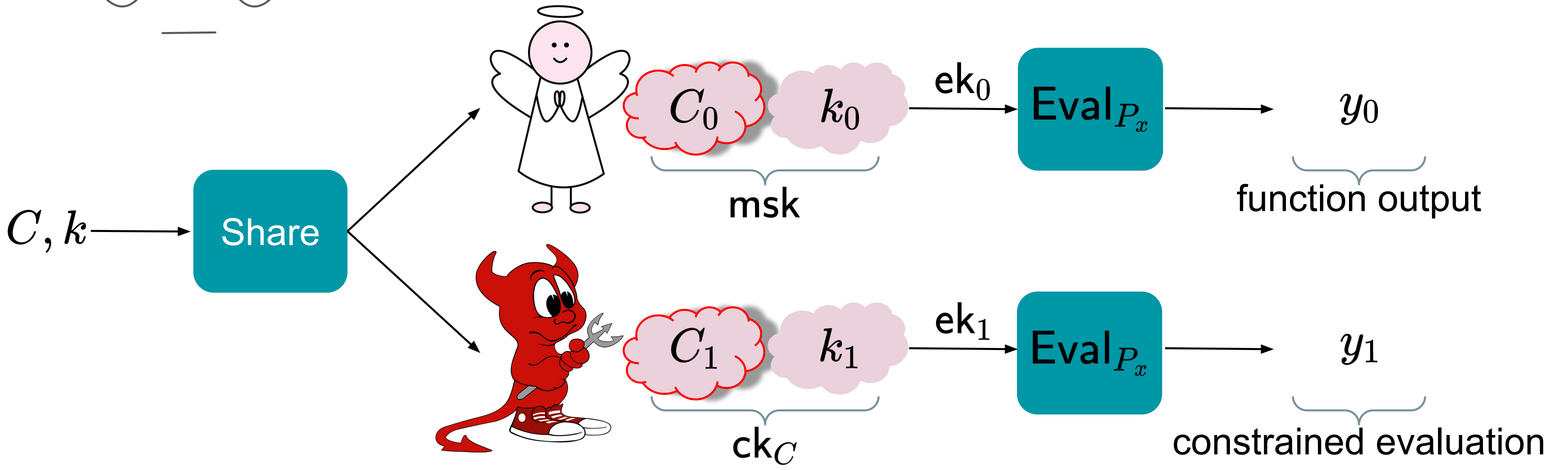
Constrained PRF from Homomorphic Secret Sharing

For a constraint $C: \mathcal{X} \to \{0,1\}$: $S_C = \{x \in \mathcal{X} : C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Take a PRF F with key k, and use an HSS to compute $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$.



msk depends on C

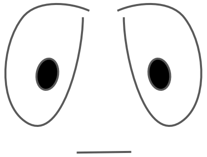


Constrained PRF from Homomorphic Secret Sharing

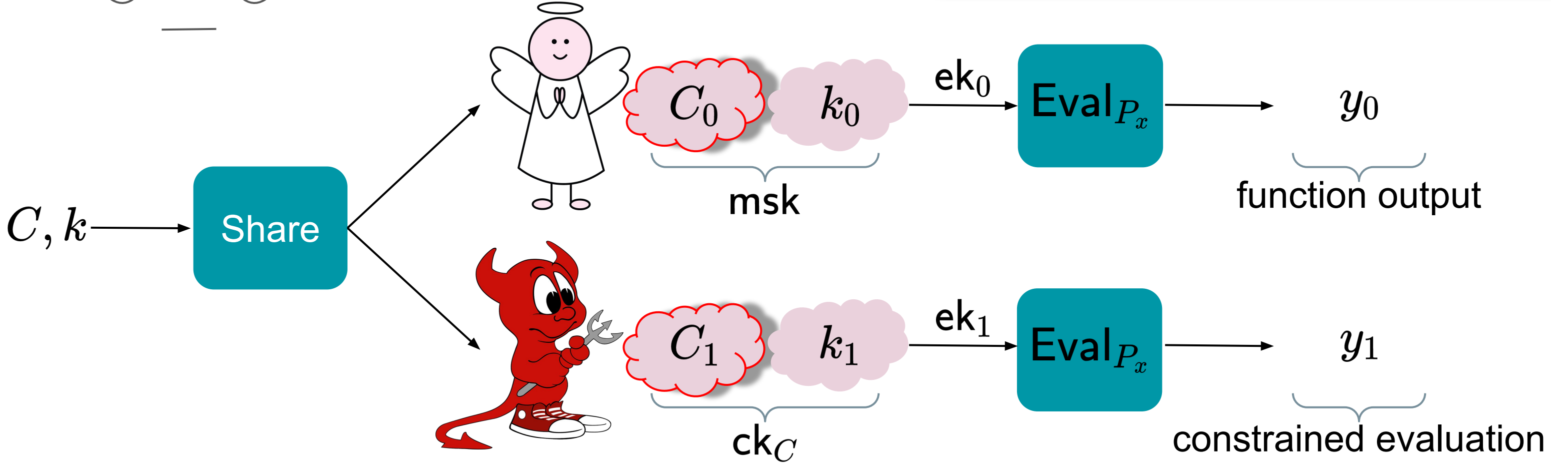
For a constraint C: $\mathcal{X} \rightarrow \{0,1\}$: $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$
The adversary can evaluate on S_C, while learning nothing about the output outside of it.

Question

How to generate C_0 before knowing C?



msk depends on C



Constrained PRF from Homomorphic Secret Sharing

What really happens!

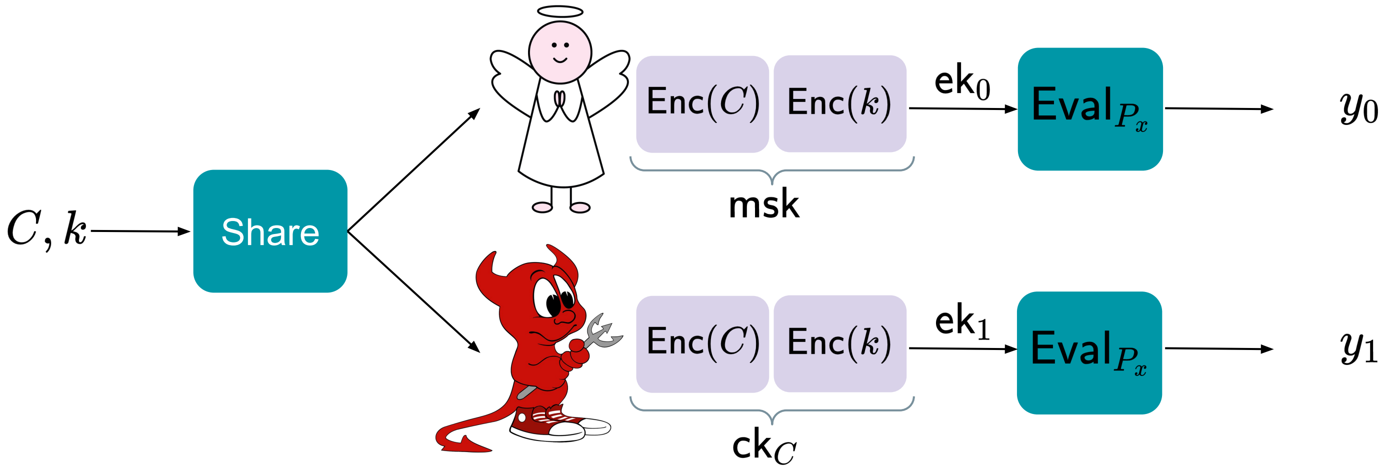
Homomorphic Secret Sharing supporting $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Homomorphic Secret Sharing supporting NC^1 programs

Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

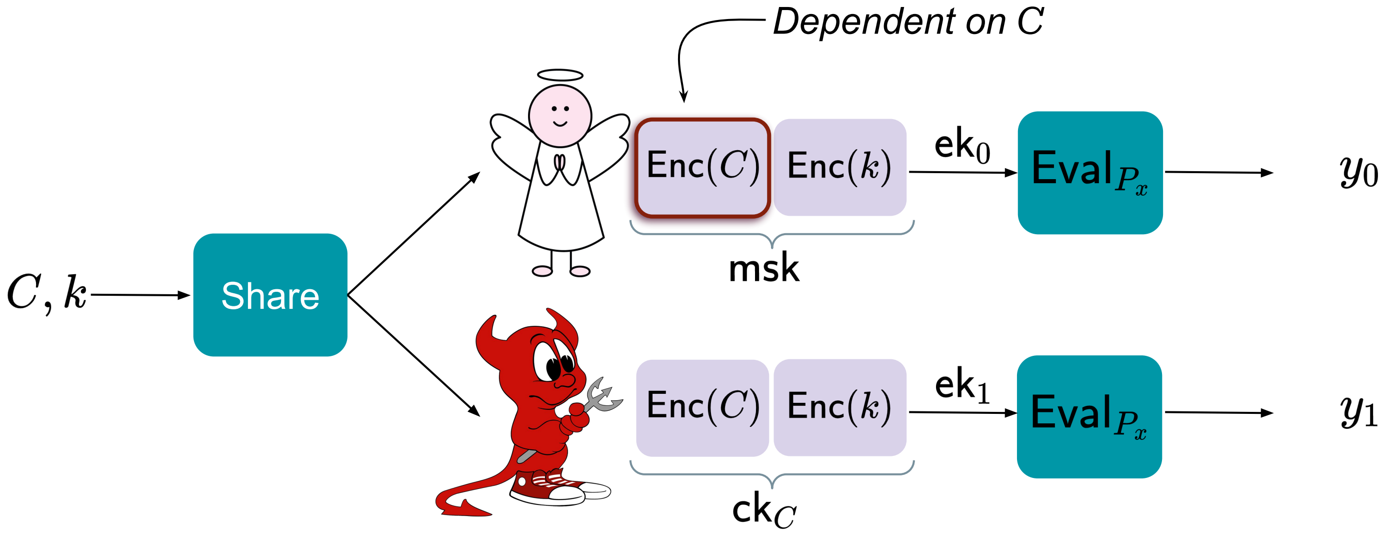
Shares: Encryptions



Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

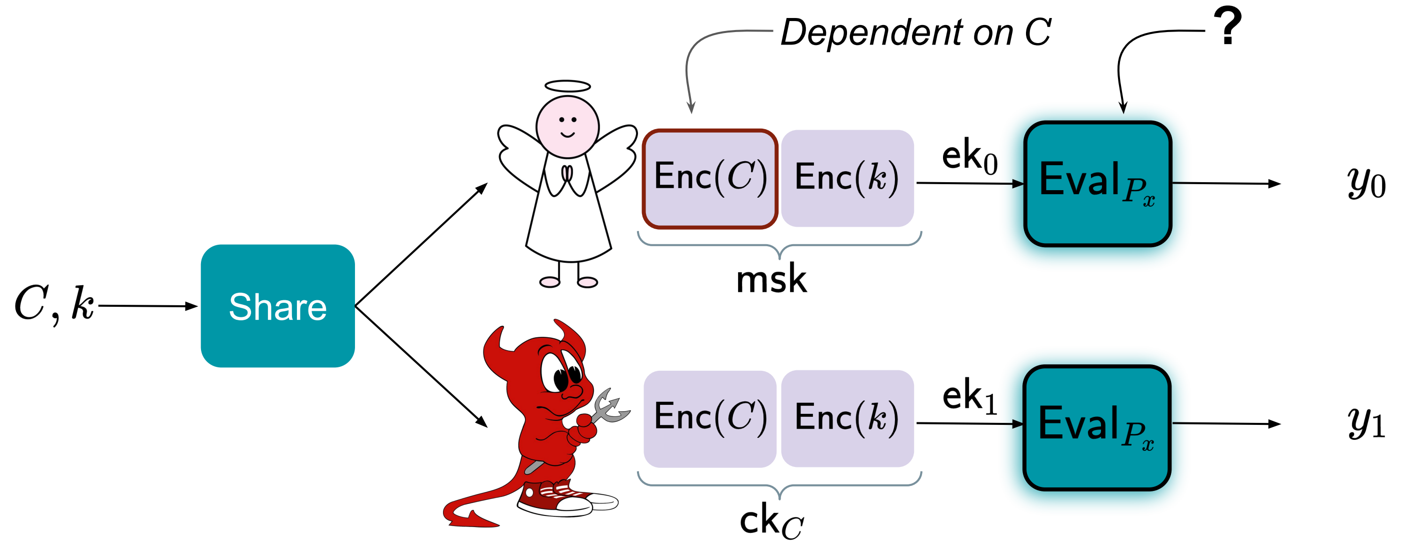
Shares: Encryptions



Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

Shares: Encryptions



Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

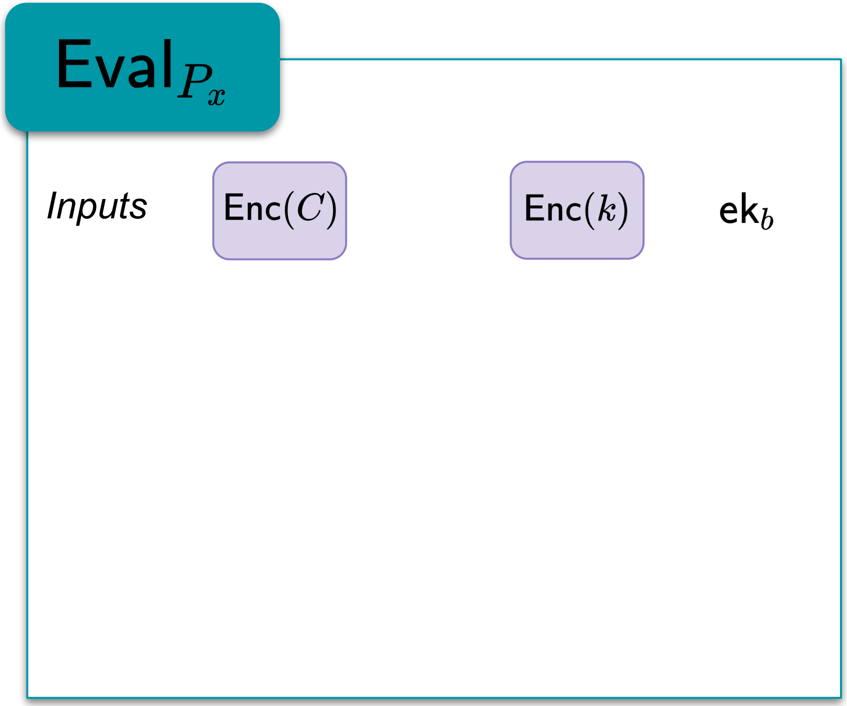
$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

Eval_{P_x}

Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

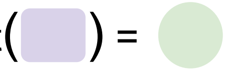


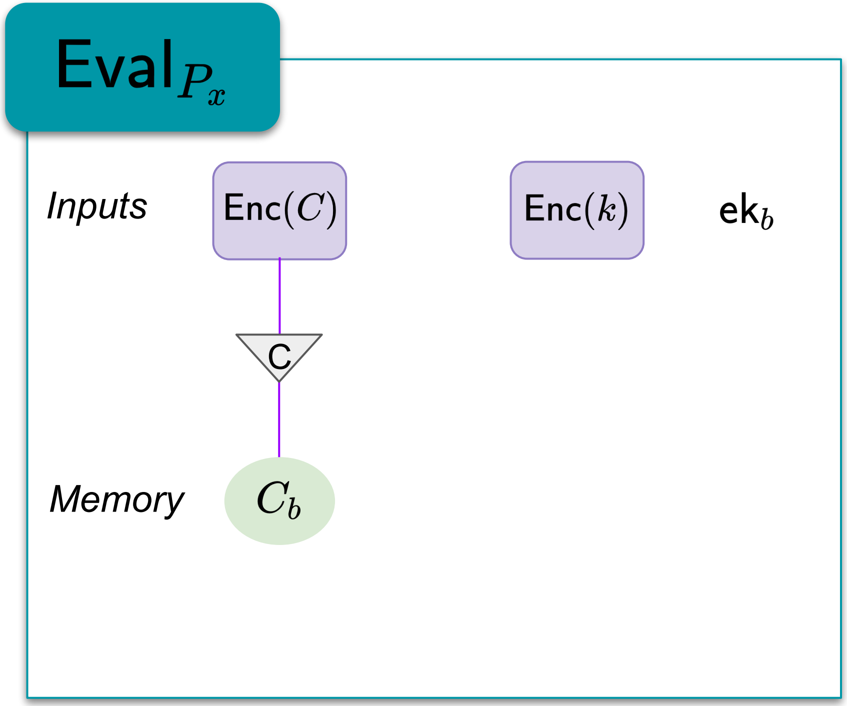
Homomorphic Secret Sharing supporting NC¹ programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation: Convert( ) =



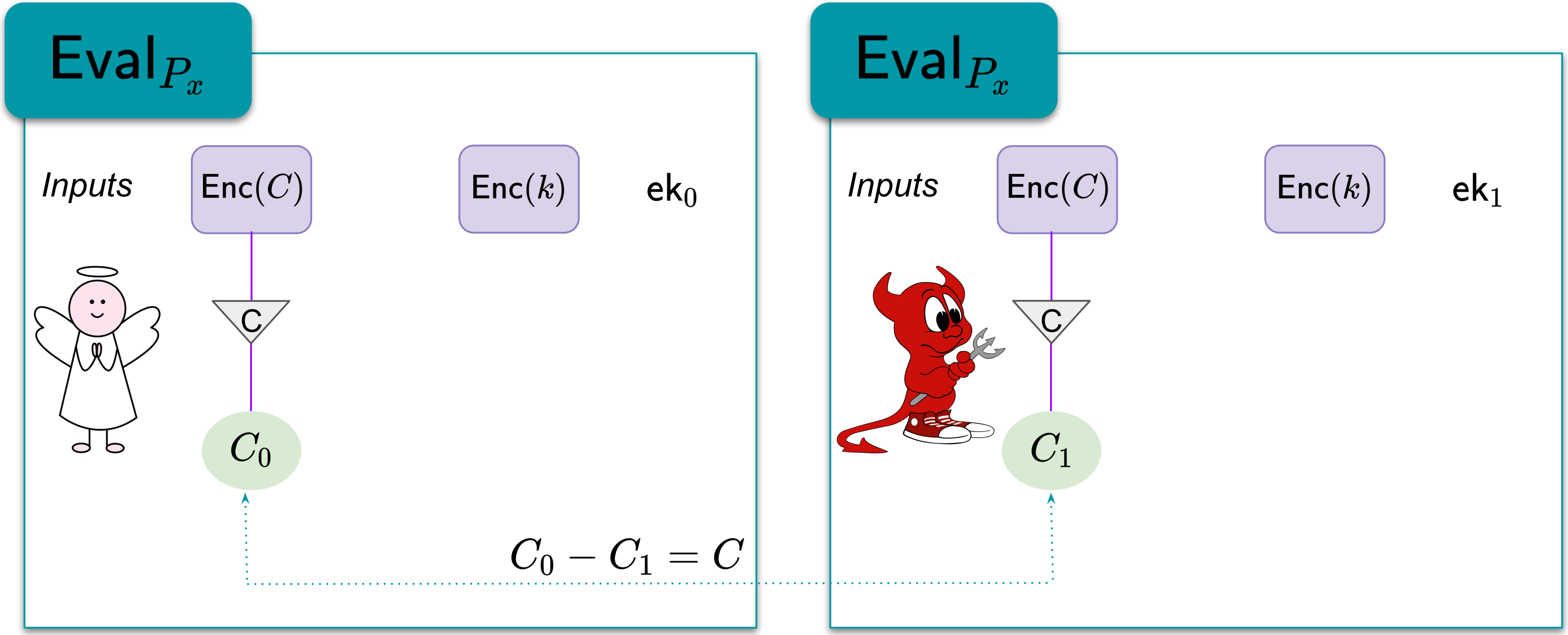


Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation: Convert(🟪) = 🟢



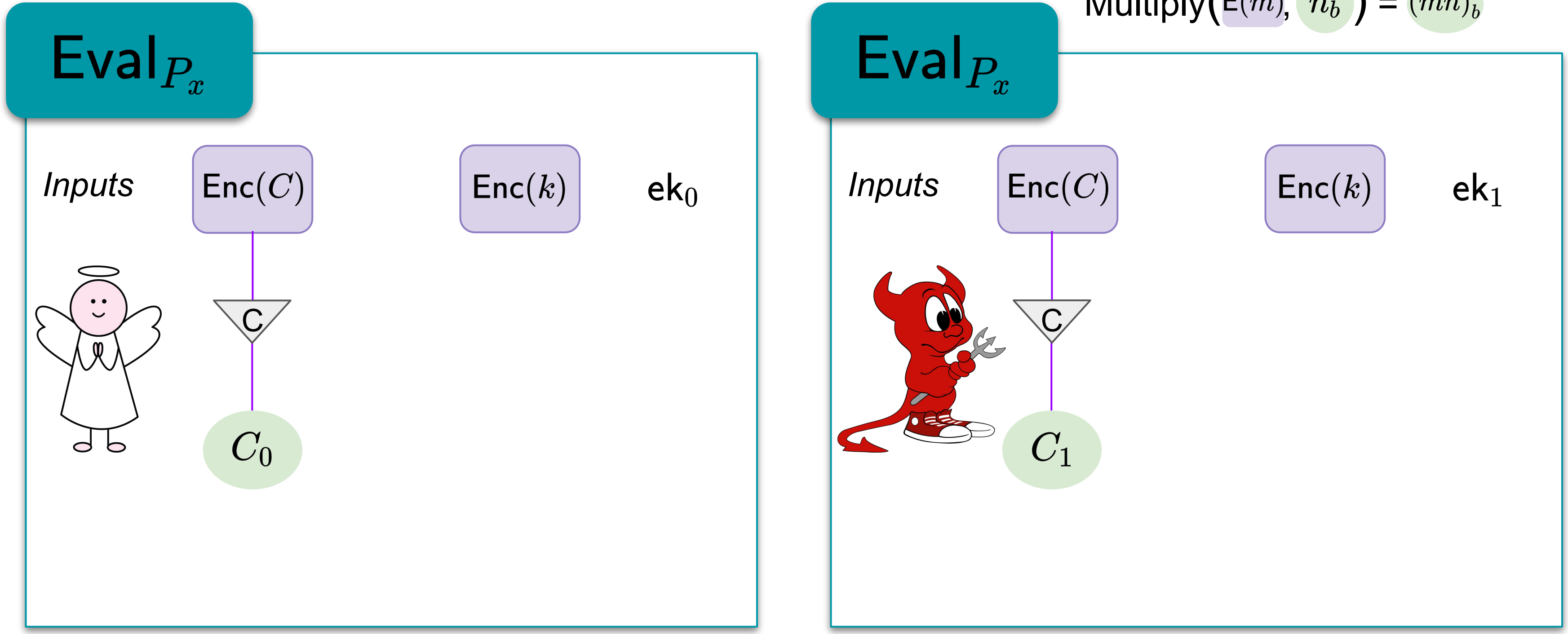
Homomorphic Secret Sharing supporting NC¹ programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operations: Convert( ) =

Multiply(E(m), n_b) = $(mn)_b$



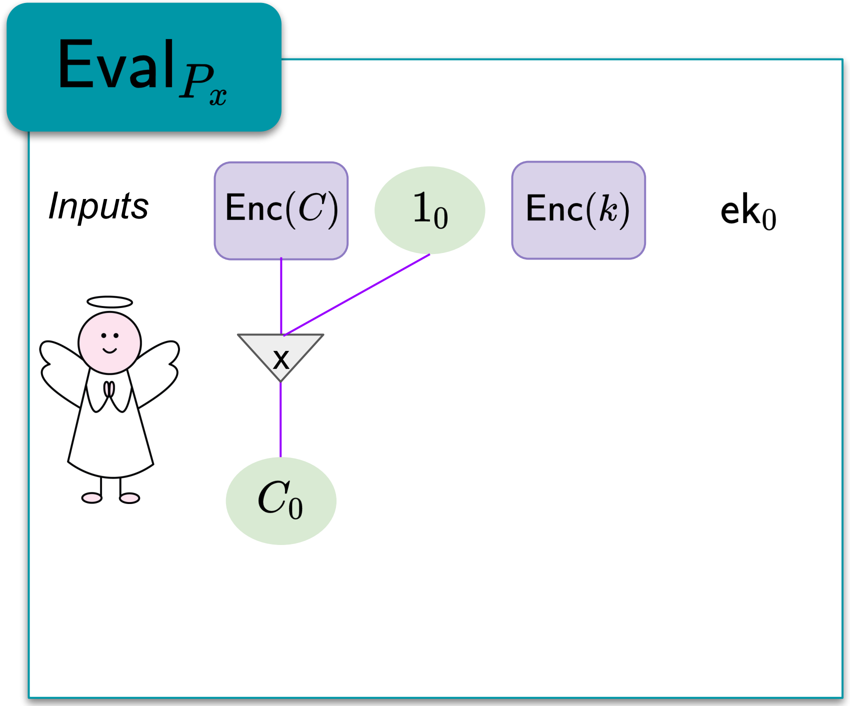
Homomorphic Secret Sharing supporting NC^1 programs

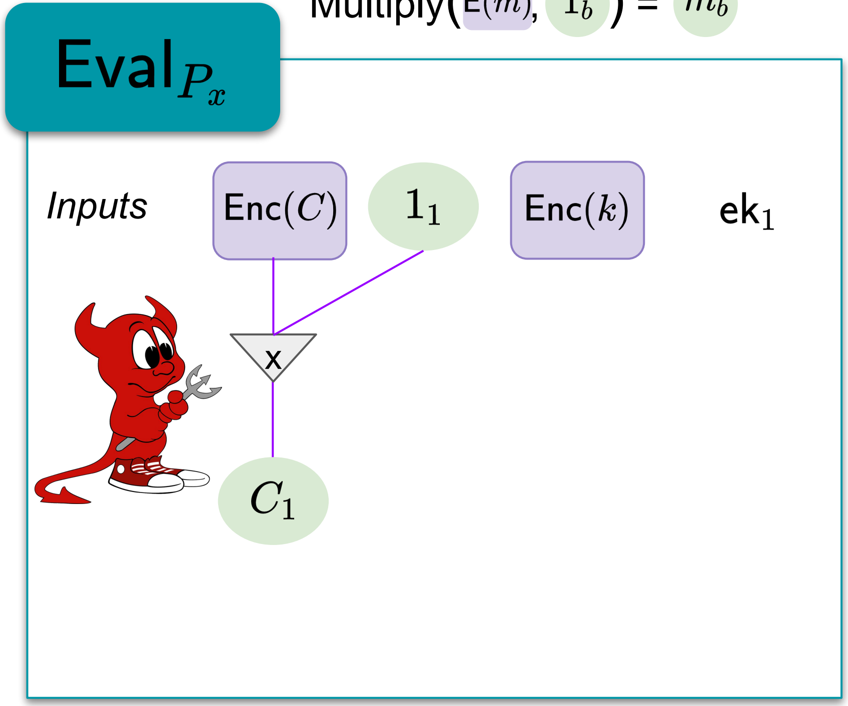
Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation: Convert(🟪) = 🟢

Multiply(E(m), 1_b) = m_b





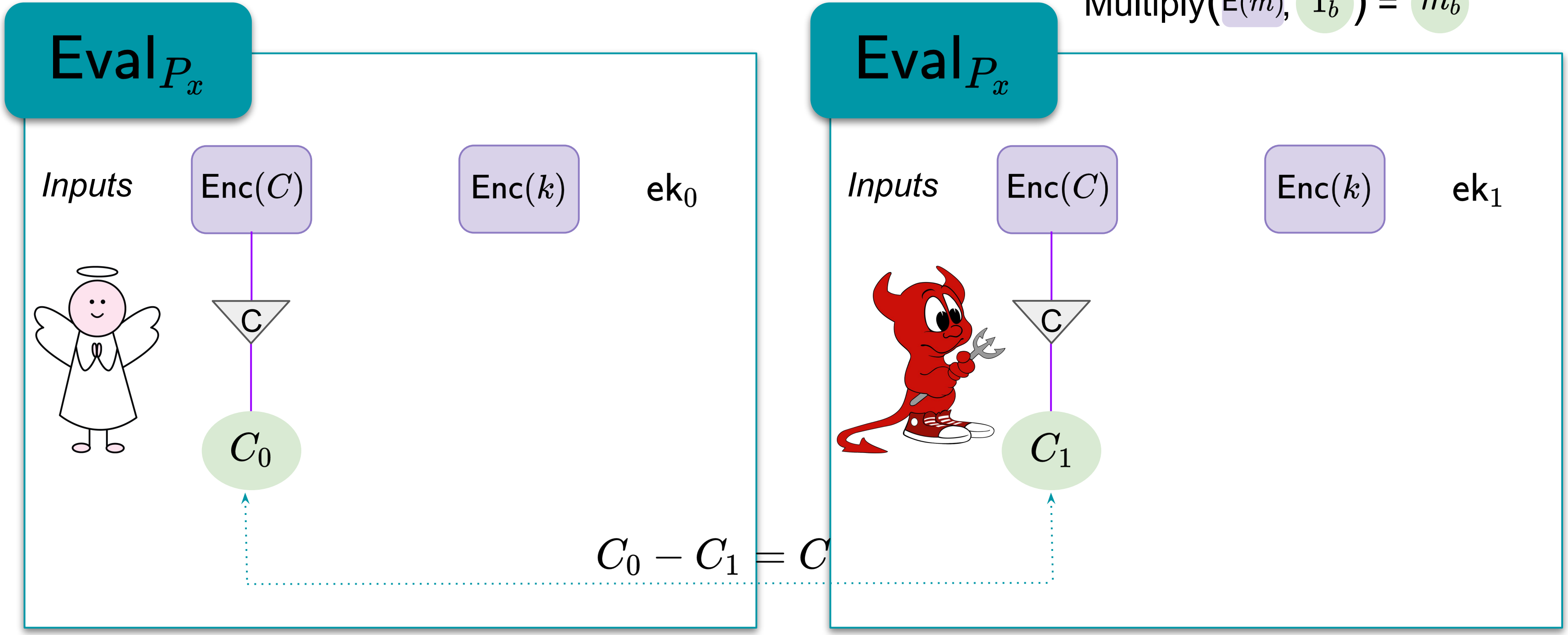
Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation: Convert(🟪) = 🟢

Multiply(🟪 E(m), 🟢 1_b) = 🟢 m_b



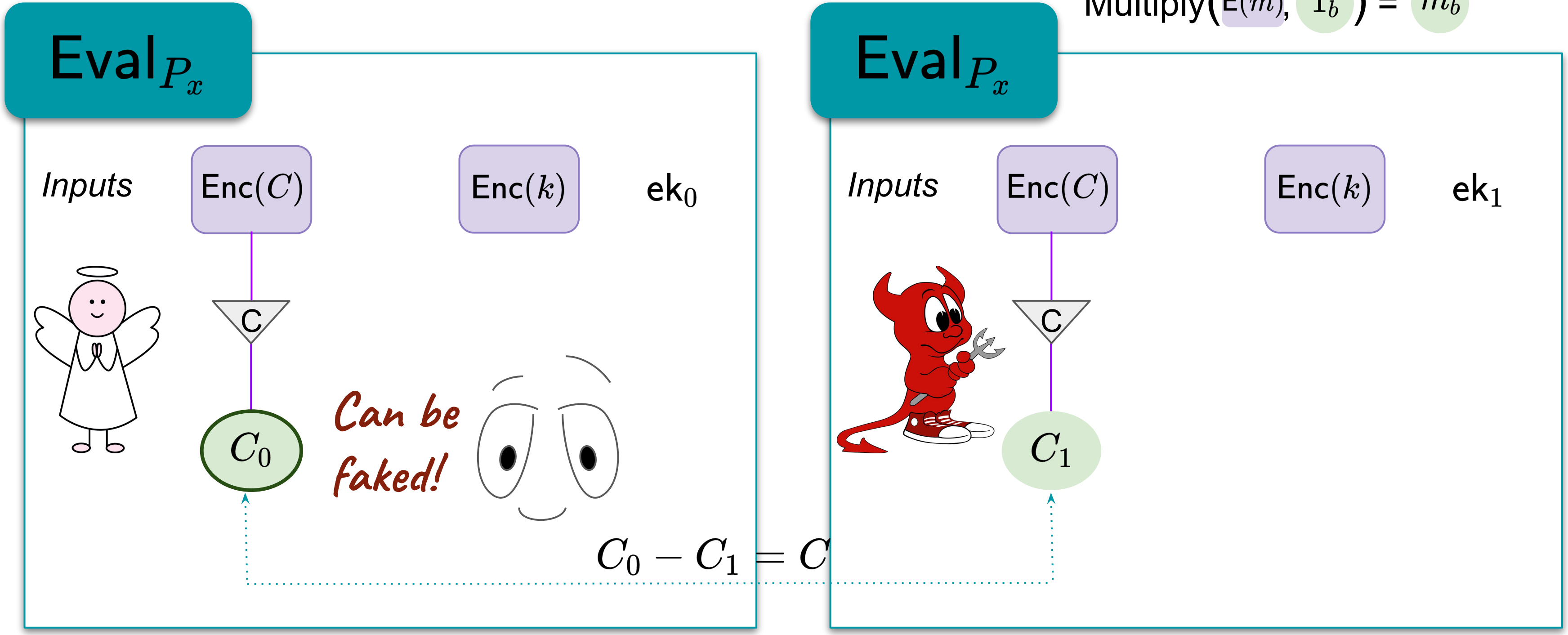
Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation: Convert( ) =

Multiply($E(m)$, 1_b) = m_b

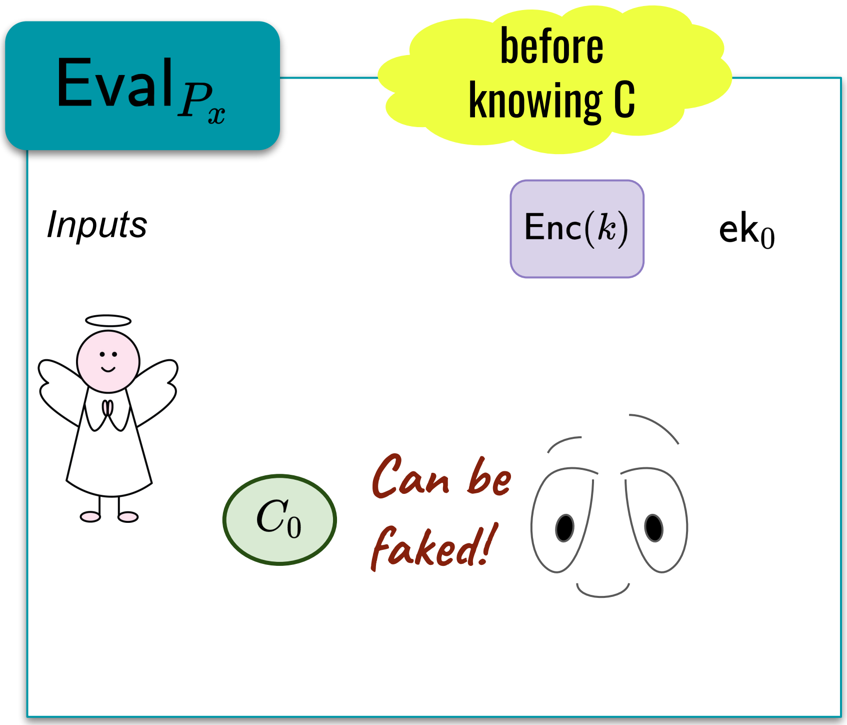


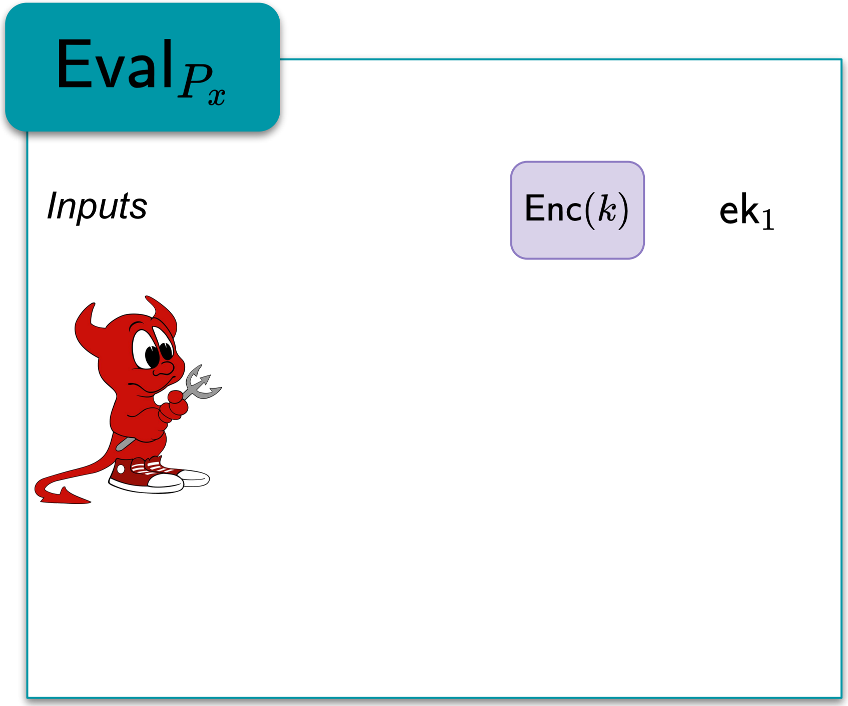
Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation:



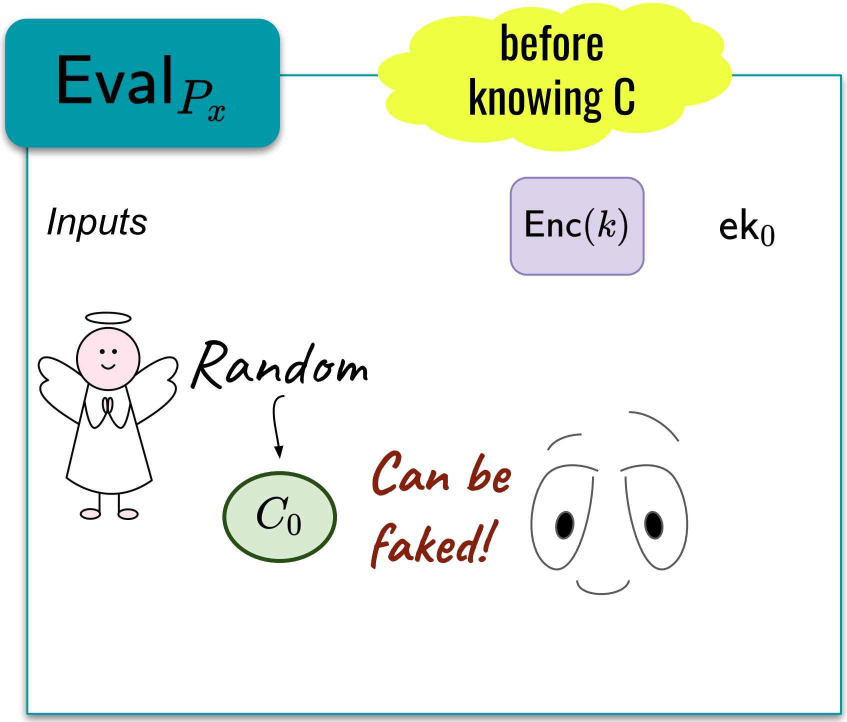


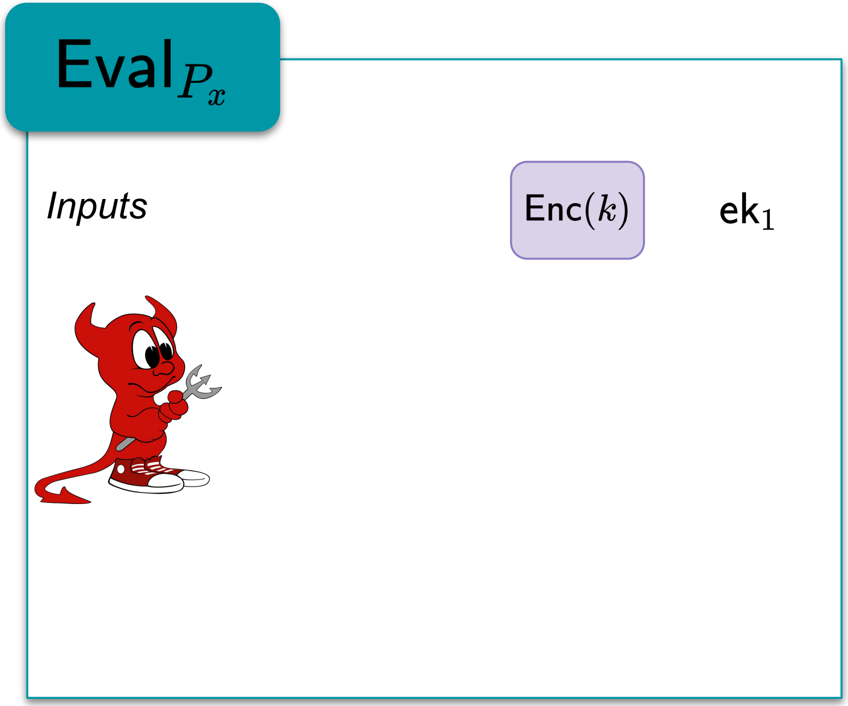
Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation:



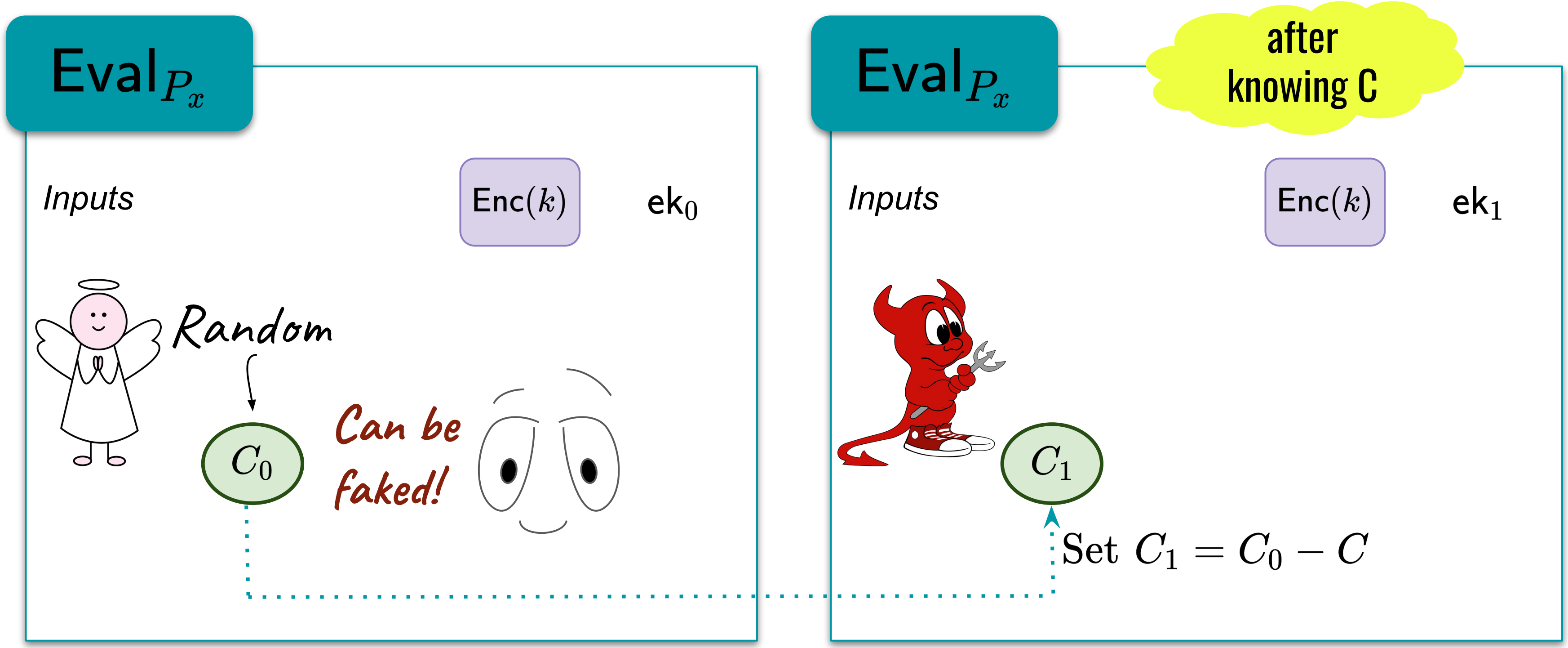


Homomorphic Secret Sharing supporting NC^1 programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation:

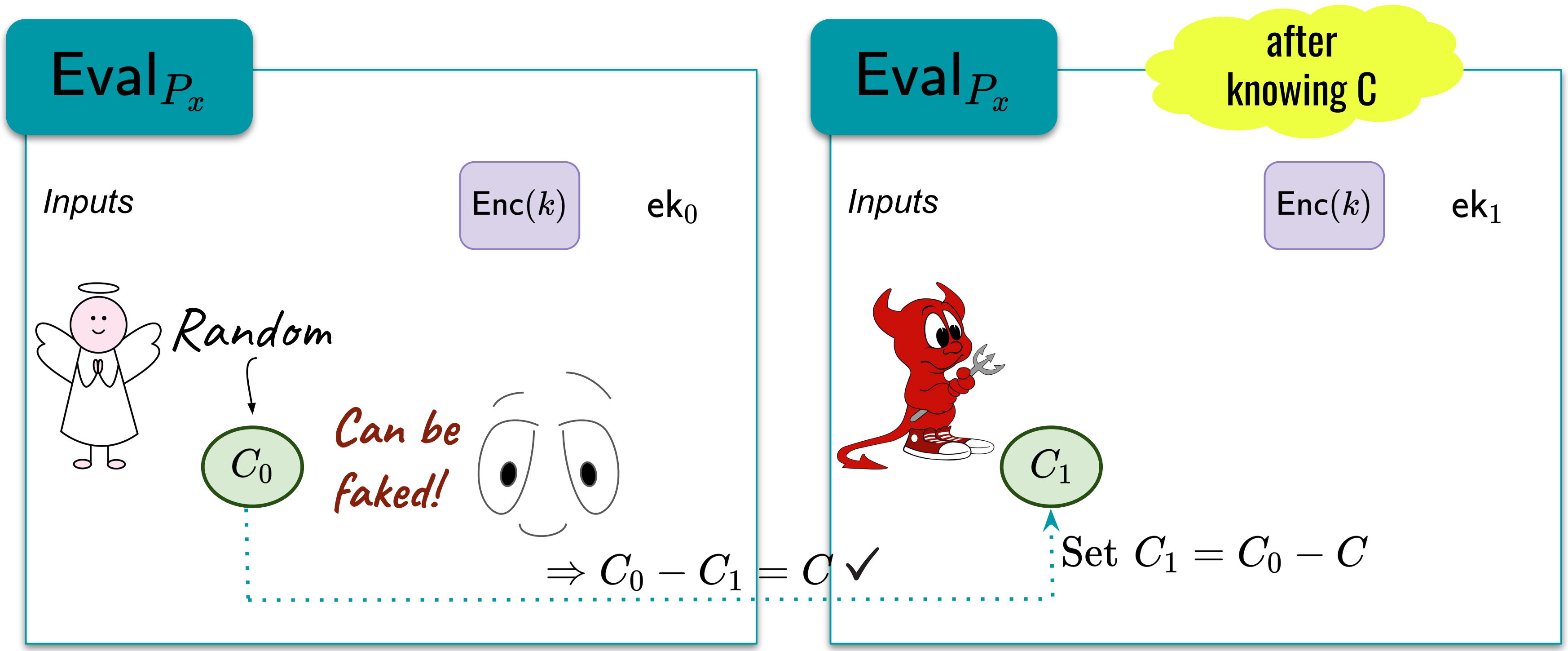


Homomorphic Secret Sharing supporting NC¹ programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation:



Homomorphic Secret Sharing supporting NC¹ programs

Using (additively homomorphic) public-key encryption scheme.

$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Allowed operation:

Eval$_{P_x}$

Inputs

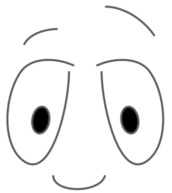
Enc(k)

ek$_0$

Random

C_0

Can be faked!



$\Rightarrow C_0 - C_1 = C$ ✓

Eval$_{P_x}$

after knowing C

Inputs

Enc(k)

ek$_1$

C_1

There's some hope!

Set $C_1 = C_0 - C$

Constrained PRF
from
Homomorphic Secret Sharing

For Inner-Product.

Constrained PRF for Inner-Product constraint

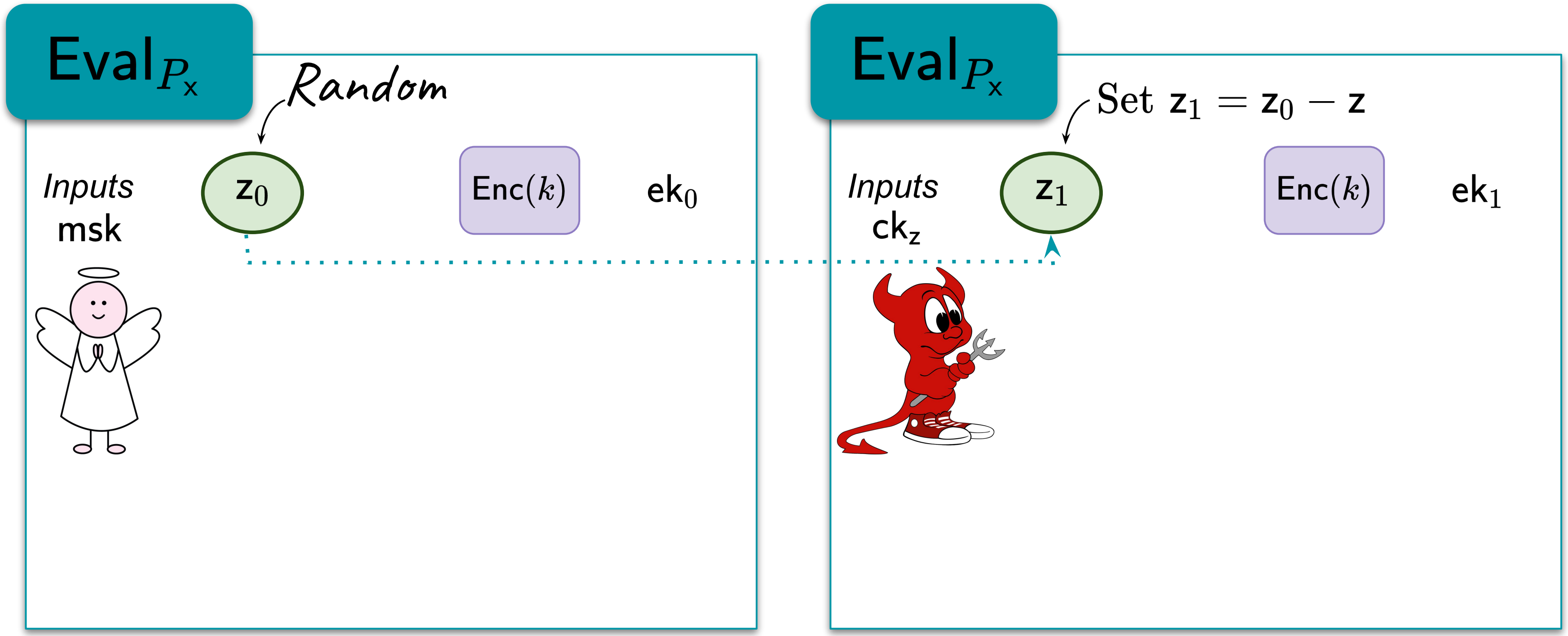
$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector $\mathbf{z}$.

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathsf{z}) \mapsto \langle \mathsf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**. Adversary can compute on **x** iff < **z**,**x** > =0.

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector $\mathbf{z}$.


$\mathsf{Eval}_{P_{\mathsf{x}}}$
Random
Inputs
msk
$\mathbf{z}_0$
$\mathsf{Enc}(k)$
ek_0
$\mathsf{Eval}_{P_{\mathsf{x}}}$
Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$
Inputs
$\mathsf{ck}_{\mathbf{z}}$
$\mathbf{z}_1$
$\mathsf{Enc}(k)$
ek_1

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**.

$\Big(\text{Convert}(\mathsf{E}(m)) = m_b \ := \ \text{Multiply}(\mathsf{E}(m), 1_b) = m_b \Big)$

$\mathsf{Eval}_{P_{\mathsf{x}}}$

Random

Inputs $\mathbf{z}_0$ $\mathsf{Enc}(k)$ ek_0

$\mathsf{Eval}_{P_{\mathsf{x}}}$

Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$

Inputs $\mathbf{z}_1$ $\mathsf{Enc}(k)$ ek_1

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**.

Convert(🟪) = 🟢 := Multiply(🟪, $\boxed{\mathbf{z}_b}$) = 🟢

$\mathsf{ExtEval}_{P_{\mathsf{x}}}$

Random

Inputs $\mathbf{z}_0$ $\mathsf{Enc}(k)$ ek_0

$\mathsf{ExtEval}_{P_{\mathsf{x}}}$

Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$

Inputs $\mathbf{z}_1$ $\mathsf{Enc}(k)$ ek_1

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**.

Convert(▭) = ● := Multiply($\mathsf{E}(m)$, $\mathbf{z}_b$) = $(\mathbf{z} \cdot m)_b$

$\mathsf{ExtEval}_{P_{\mathsf{x}}}$

Random

Inputs $\mathbf{z}_0$ $\mathsf{Enc}(k)$ ek_0

$\mathsf{ExtEval}_{P_{\mathsf{x}}}$

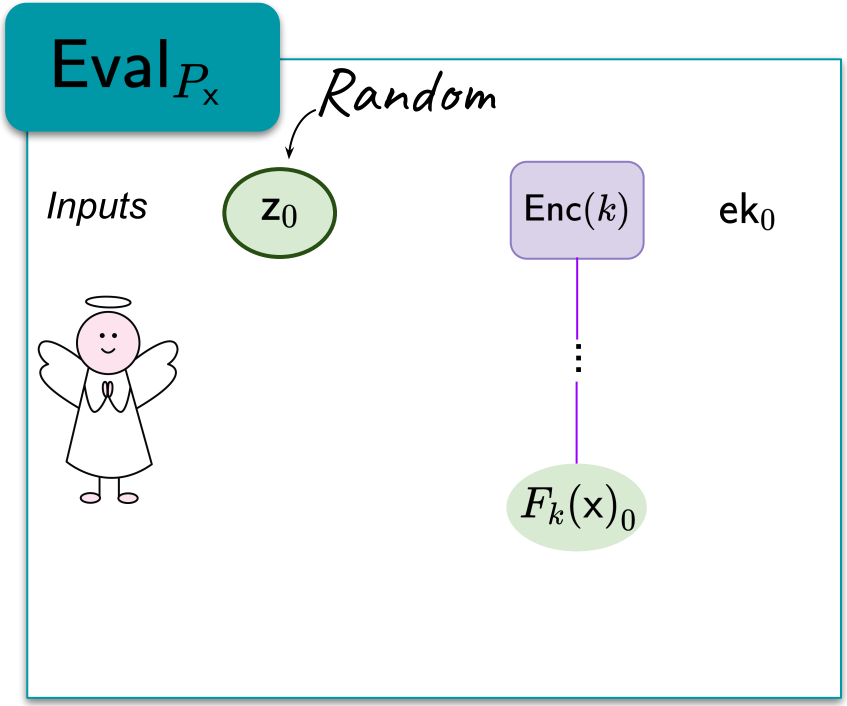
Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$

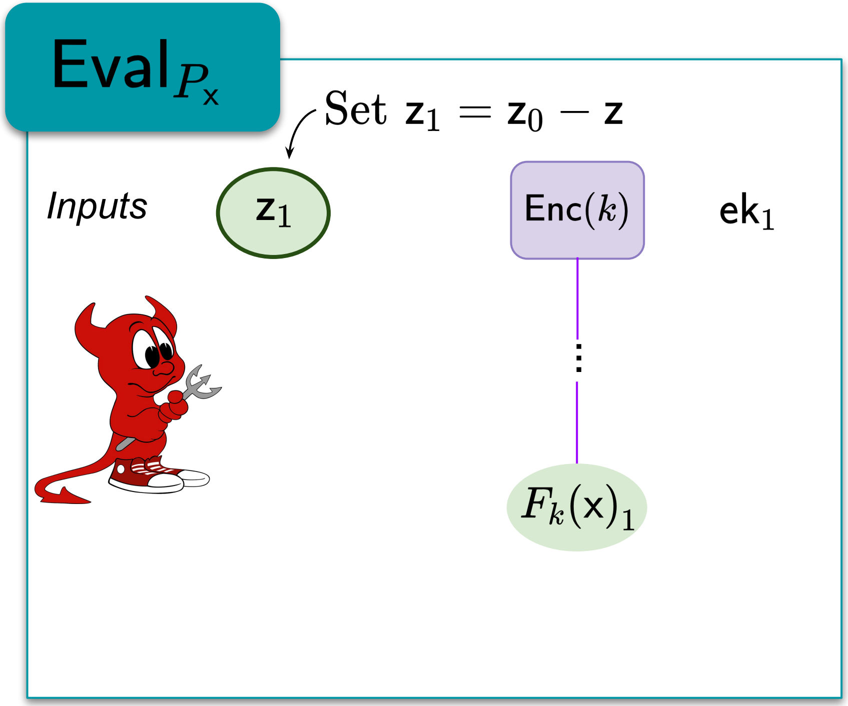
Inputs $\mathbf{z}_1$ $\mathsf{Enc}(k)$ ek_1

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**.

Convert(▨) = ● := Multiply($\mathsf{E}(m)$, 1_b) = m_b



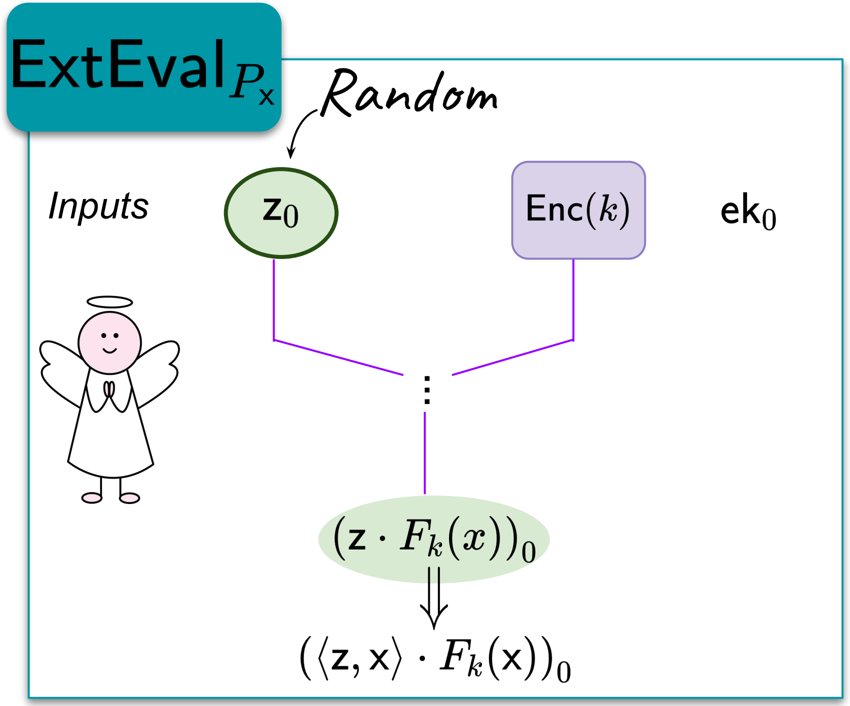


$\mathsf{Eval}_{P_{\mathsf{x}}}$
Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$
Inputs
$\mathbf{z}_1$
$\mathsf{Enc}(k)$
ek_1
$F_k(\mathsf{x})_1$

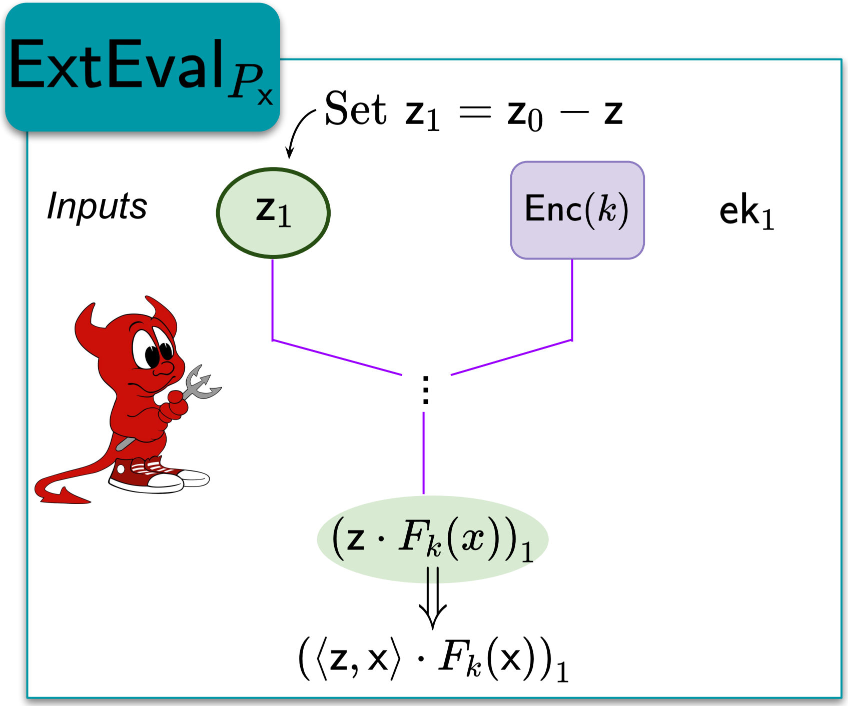
Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector **z**.

$\left(\text{Convert}(\,🟪\,) = 🟢 := \text{Multiply}(\mathsf{E}(m), \mathbf{z}_b) = (\mathbf{z} \cdot m)_b\right)$



$\mathsf{ExtEval}_{P_{\mathsf{x}}}$
Random
Inputs
$\mathbf{z}_0$
$\mathsf{Enc}(k)$
ek_0
$\vdots$
$(\mathbf{z} \cdot F_k(x))_0$
$\Downarrow$
$(\langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x}))_0$

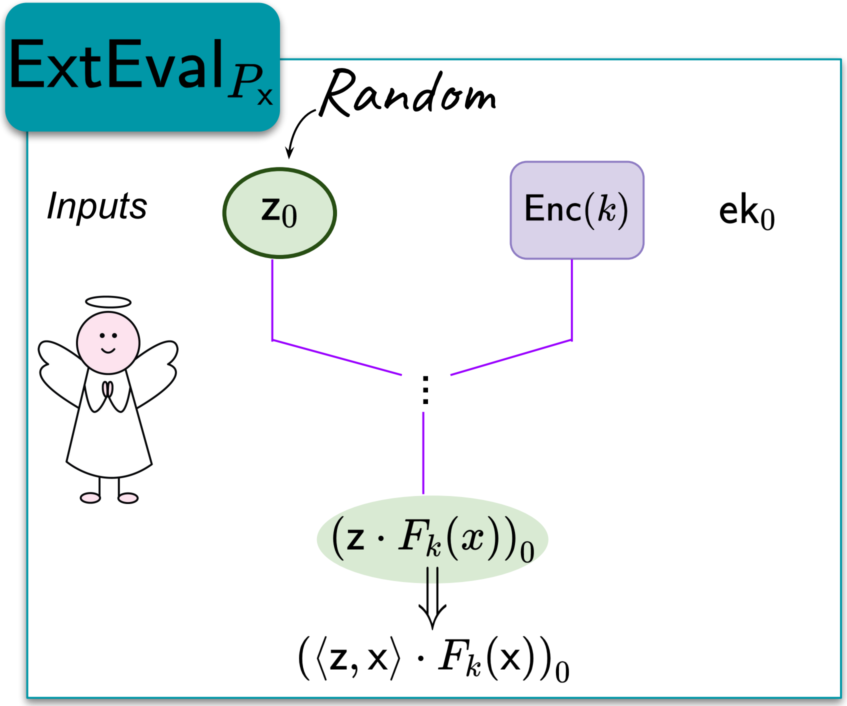


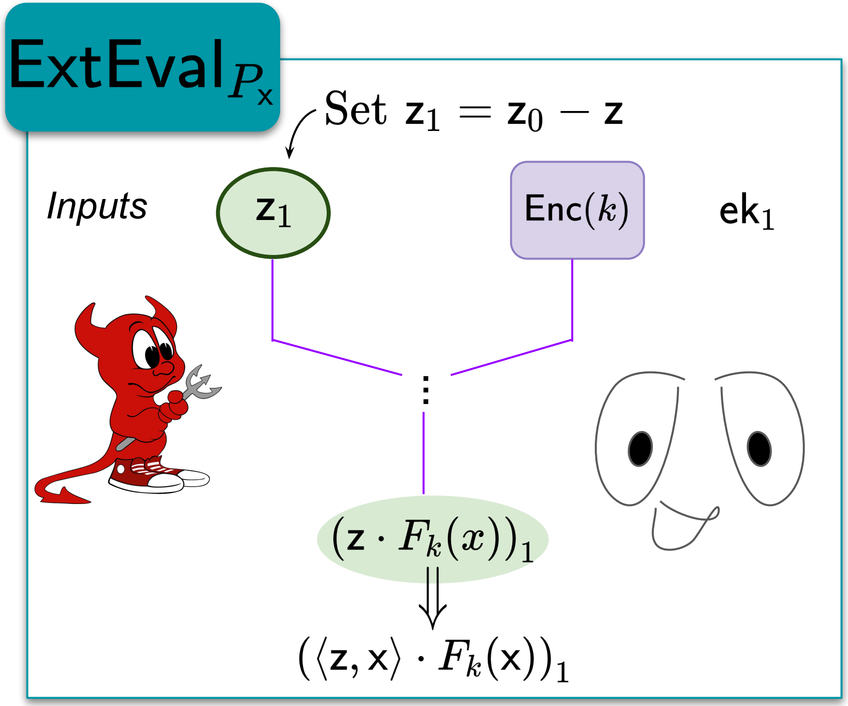
$\mathsf{ExtEval}_{P_{\mathsf{x}}}$
Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$
Inputs
$\mathbf{z}_1$
$\mathsf{Enc}(k)$
ek_1
$\vdots$
$(\mathbf{z} \cdot F_k(x))_1$
$\Downarrow$
$(\langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x}))_1$

Constrained PRF for Inner-Product constraint

$P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$ for a vector $\mathbf{z}$.

$\text{Convert}(▨) = ● := \text{Multiply}(\mathsf{E}(m), \boxed{\mathbf{z}_b}) = (\mathbf{z} \cdot m)_b$

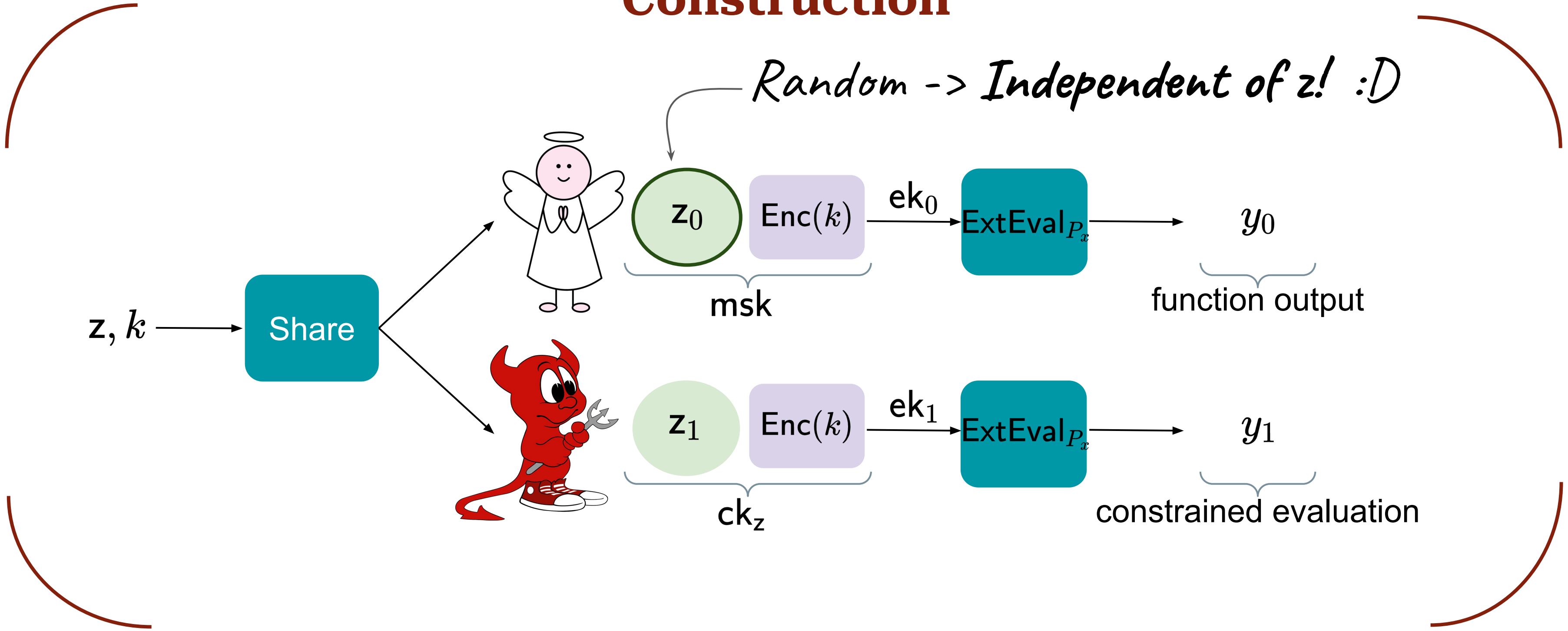

$\mathsf{ExtEval}_{P_{\mathsf{x}}}$
Random
Inputs
$\mathbf{z}_0$
$\mathsf{Enc}(k)$
ek_0
$\vdots$
$(\mathbf{z} \cdot F_k(x))_0$
$\Downarrow$
$(\langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x}))_0$


$\mathsf{ExtEval}_{P_{\mathsf{x}}}$
Set $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{z}$
Inputs
$\mathbf{z}_1$
$\mathsf{Enc}(k)$
ek_1
$\vdots$
$(\mathbf{z} \cdot F_k(x))_1$
$\Downarrow$
$(\langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x}))_1$

Constrained PRF from Homomorphic Secret Sharing

Inner-Product Constraint constraint C : vector $\mathbf{z}$ $P_{\mathsf{x}} : (k, \mathsf{z}) \mapsto \langle \mathsf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$

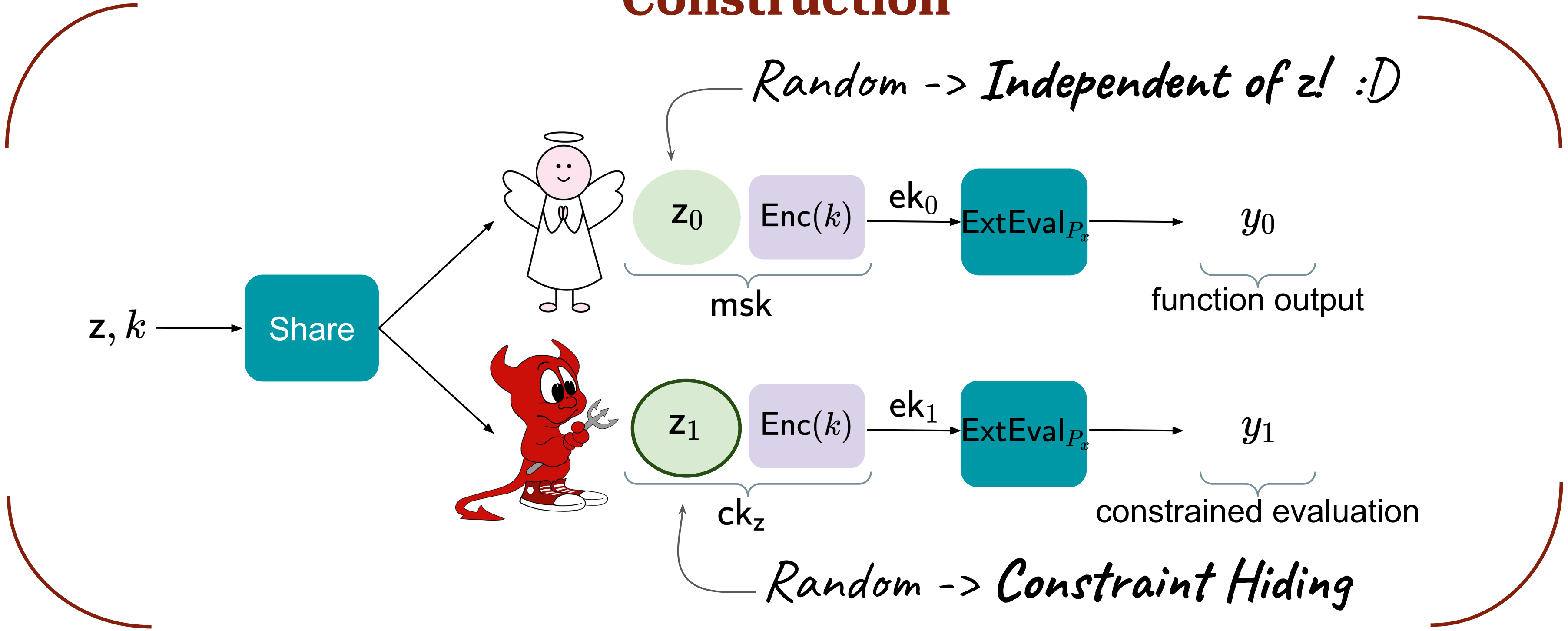
Construction



Constrained PRF from Homomorphic Secret Sharing

Inner-Product Constraint   constraint C : vector $\mathbf{z}$ $P_{\mathsf{x}} : (k, \mathbf{z}) \mapsto \langle \mathbf{z}, \mathsf{x} \rangle \cdot F_k(\mathsf{x})$

Construction



Constrained PRF from Homomorphic Secret Sharing

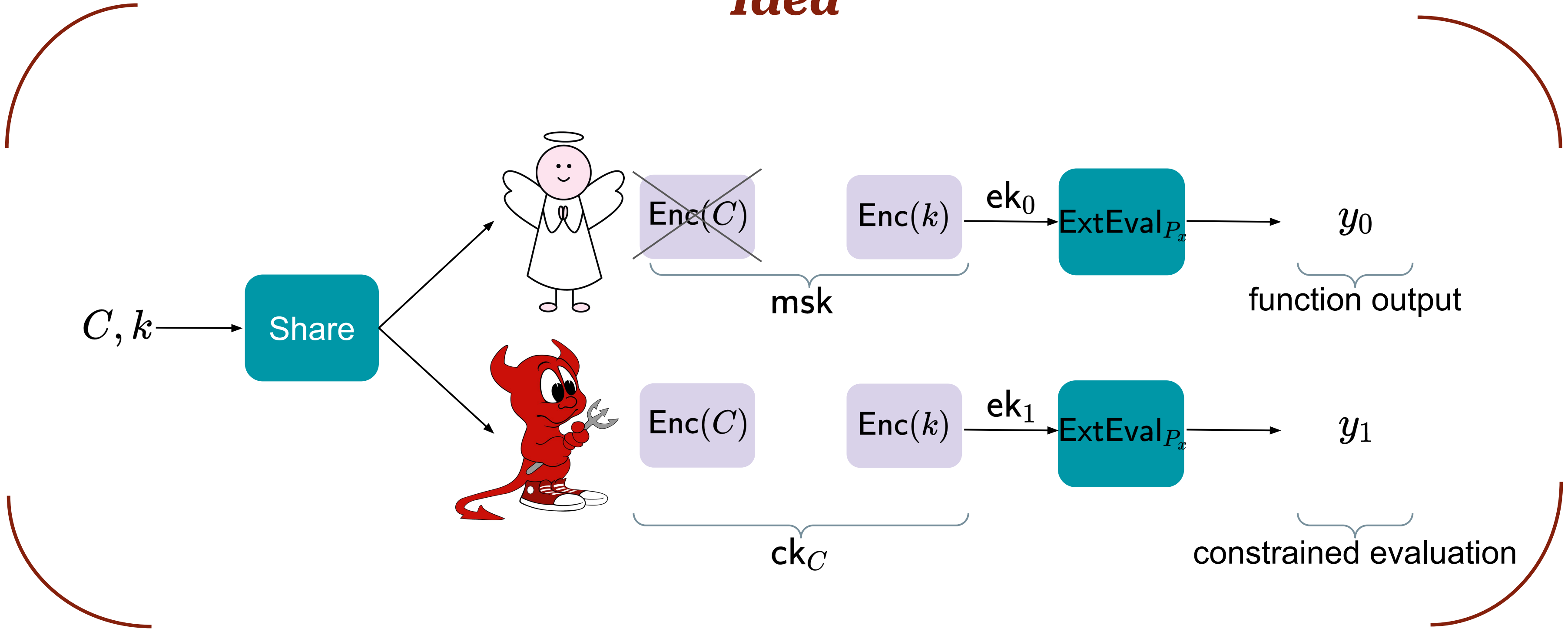
For NC^1

Constrained PRF from Homomorphic Secret Sharing

NC^1 Constraint

$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

Idea

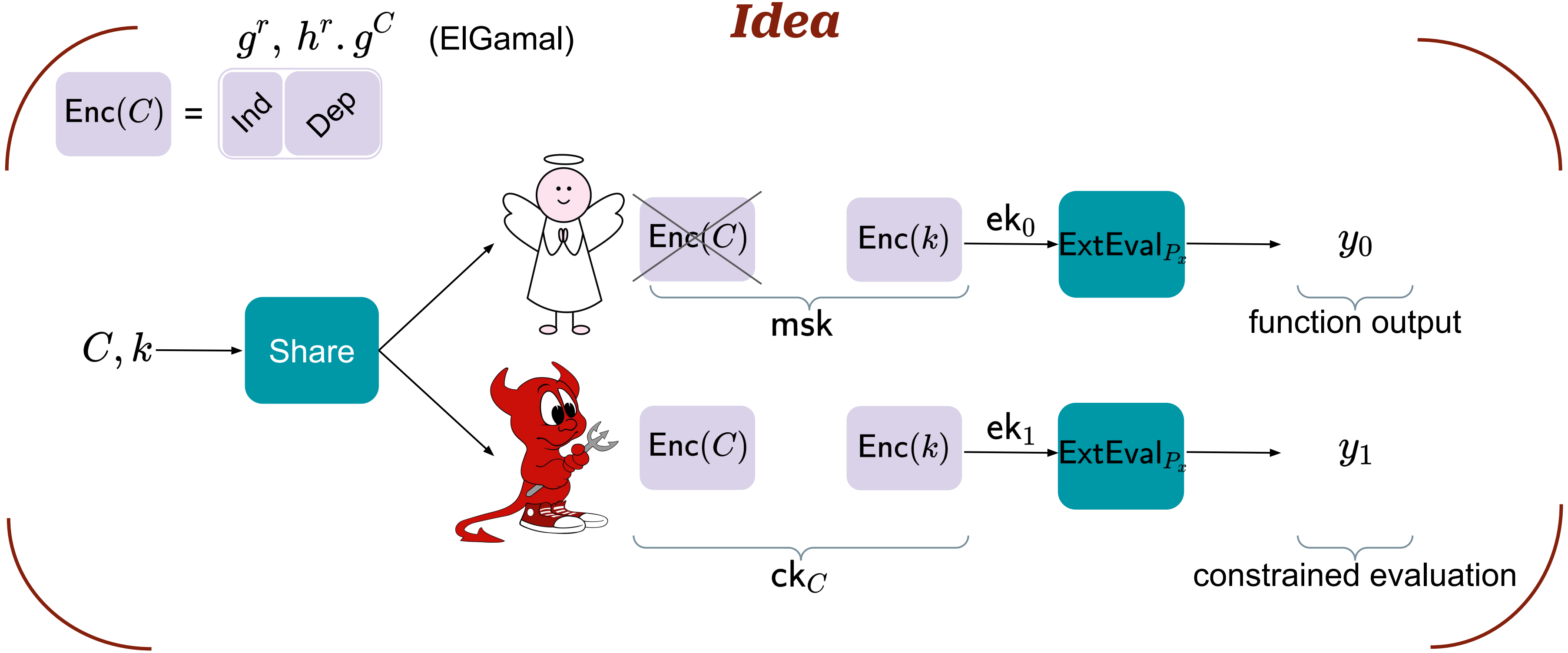


Constrained PRF from Homomorphic Secret Sharing

NC¹ Constraint

$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

Idea

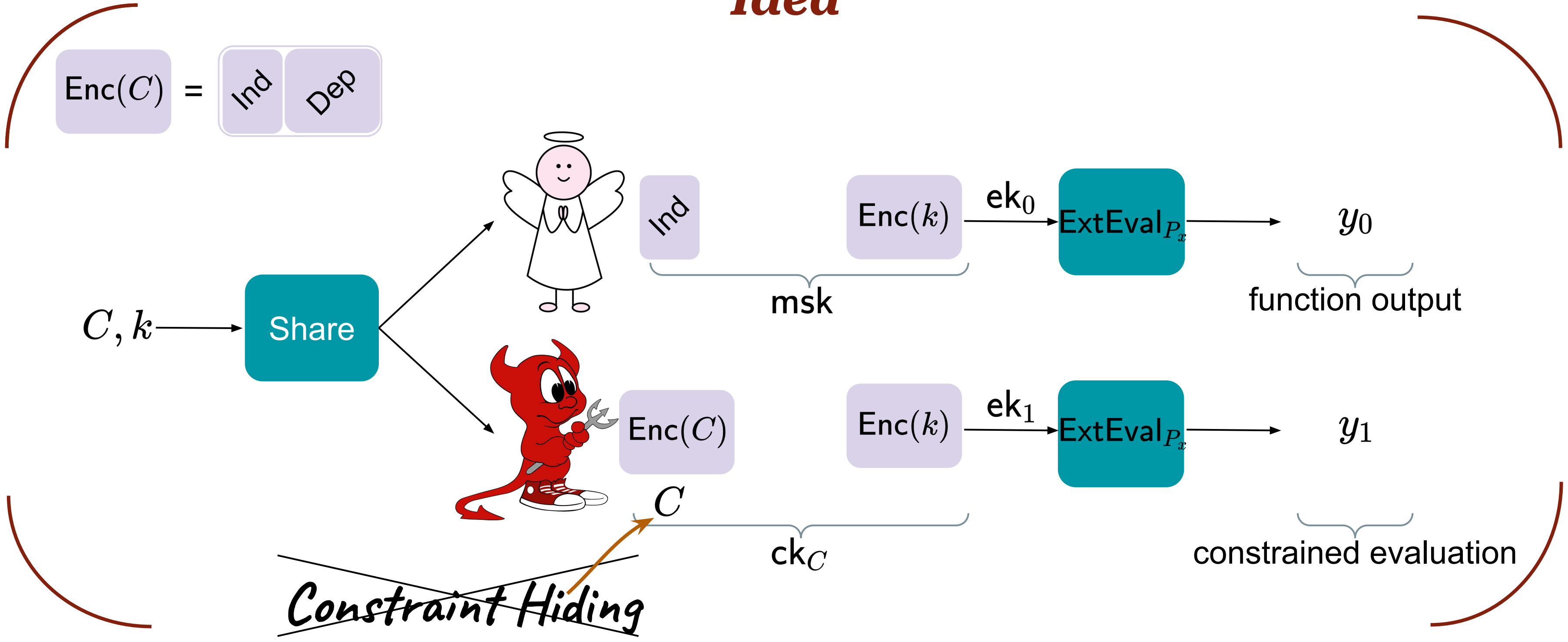


Constrained PRF from Homomorphic Secret Sharing

NC^1 Constraint

$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

Idea

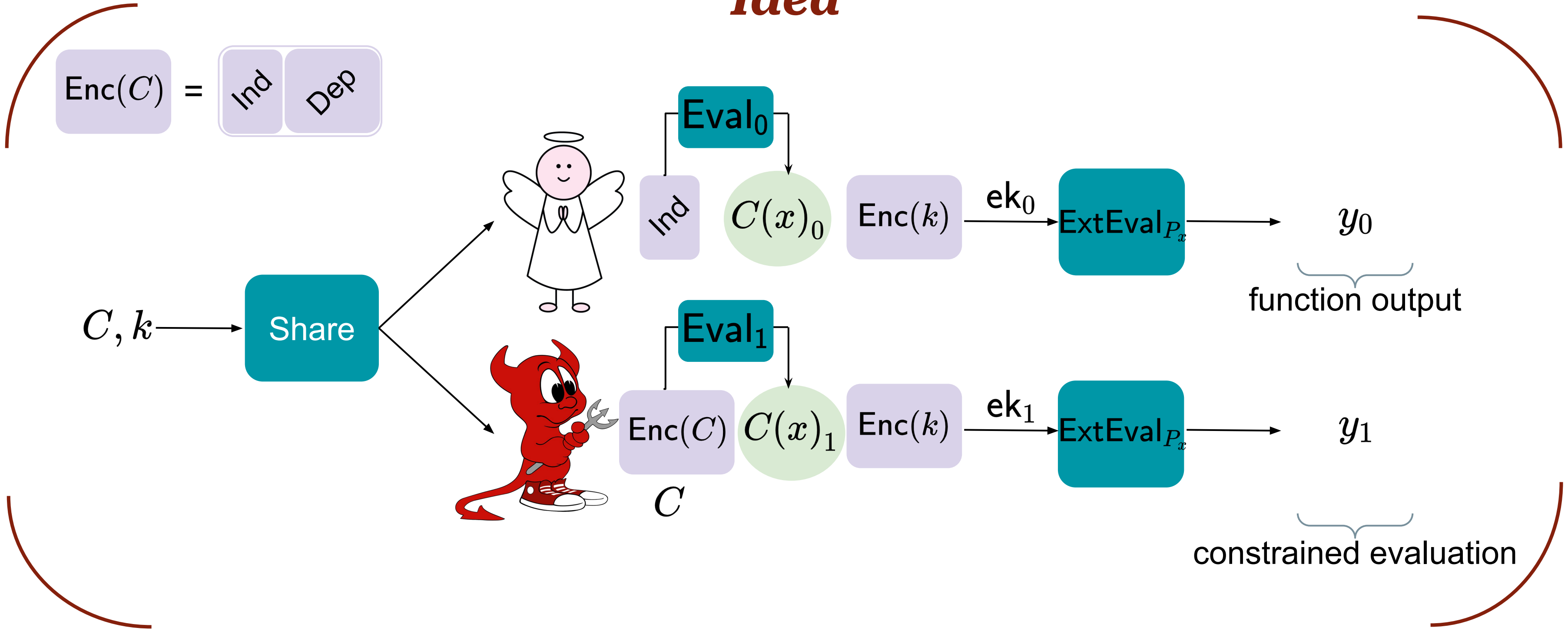


Constrained PRF from Homomorphic Secret Sharing

NC^1 Constraint

$$P_x : (k, C) \mapsto C(x) \cdot F_k(x)$$

Idea



Conclusion

- HSS + (some level of) Programmability -> Constrained PRF (for inner-product and NC^1)

- New constructions of constrained PRF.

  (1) **D**ecisional **C**omposite **R**esiduoisity, (2) **LWE** with superpolynomial modulus, (3) Hardness of the **Joye-Libert** encryption scheme, (4) **DDH & DXDH** over class groups, (5) **H**ard **M**embership **S**ubgroup over class groups

- Revisiting Applications of HSS to Secure Computation.

  - Secure computation with silent preprocessing. (one party can preprocess even before knowing the identity of the other party)
  - One-sided statistically secure computation with sublinear communication. (without FHE!)

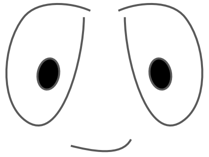
Conclusion

- HSS + (some level of) Programmability -> Constrained PRF (for inner-product and NC^1)
- New constructions of constrained PRF.

  (1) **D**ecisional **C**omposite **R**esiduoisity, (2) **LWE** with superpolynomial modulus, (3) Hardness of the **Joye-Libert** encryption scheme, (4) **DDH & DXDH** over class groups, (5) **H**ard **M**embership **S**ubgroup over class groups

- Revisiting Applications of HSS to Secure Computation.
  - Secure computation with silent preprocessing. (one party can preprocess even before knowing the identity of the other party)
  - One-sided statistically secure computation with sublinear communication. (without FHE!)

Thank You!



eprint.iacr.org/2023/387