Randomness Delegation
&
its Connections with Secure Computation

Mahshid Riahinia
ENS, CNRS

Journées Nationales 2025 du GDR Sécurité Informatique
24 Juin 2025

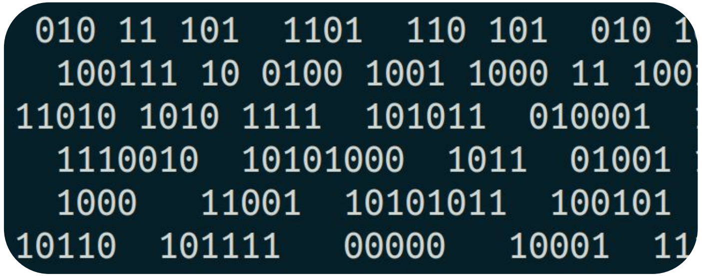
Pseudorandom Functions

Pseudorandom Functions (PRFs) [GGM86]

Definition. Deterministic keyed functions indistinguishable from truly random functions.

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

Set of outputs with msk



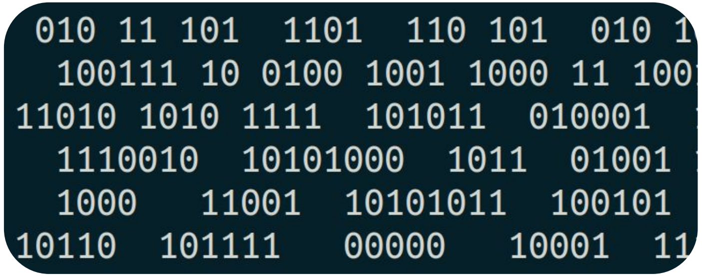
Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

Pseudorandom Functions (PRFs) [GGM86]

Definition. Deterministic keyed functions indistinguishable from truly random functions.

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

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

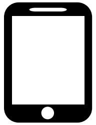
Set of outputs without msk

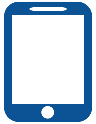
Pseudorandom Functions (PRFs) [GGM86]

Definition. Deterministic keyed functions indistinguishable from truly random functions.

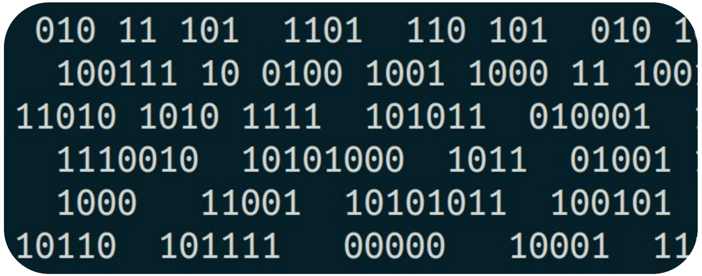
$F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Application





Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

Set of outputs without msk

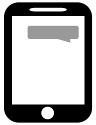
Pseudorandom Functions (PRFs) [GGM86]

Definition. Deterministic keyed functions indistinguishable from truly random functions.

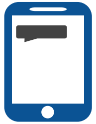
$F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

Application

Secret-Key Encryption



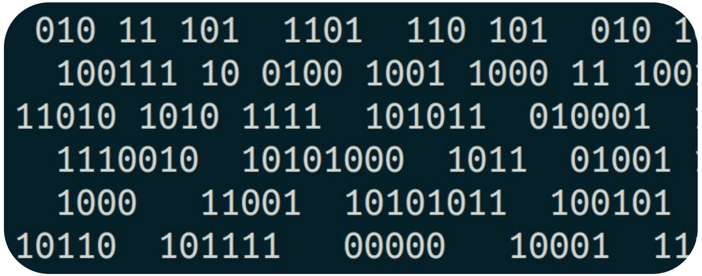
$\text{now}, c = m \oplus F_k(\text{now})$



m

$m \leftarrow c \oplus F_k(\text{now})$

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

Set of outputs without msk

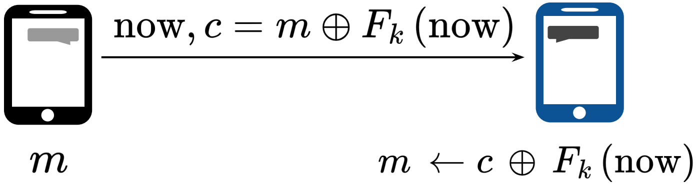
Pseudorandom Functions (PRFs) [GGM86]

Definition. Deterministic keyed functions indistinguishable from truly random functions.

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

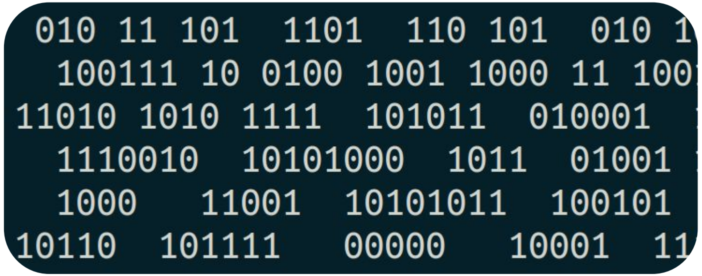
Application

Secret-Key Encryption



+ many more

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

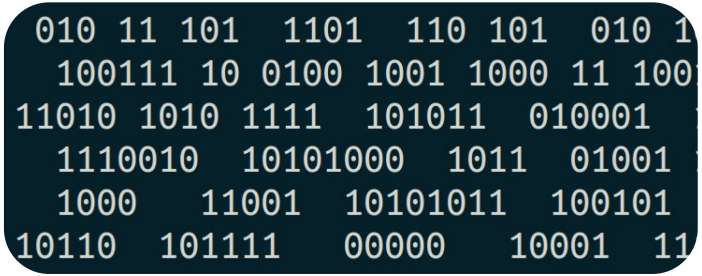
Set of outputs without msk

Constrained Pseudorandom Functions

Constrained Pseudorandom Functions (CPRFs) [BW13,KPTZ13,BGI14]

Pseudorandom Functions *with constrained access to the evaluation.*

Set of outputs with msk



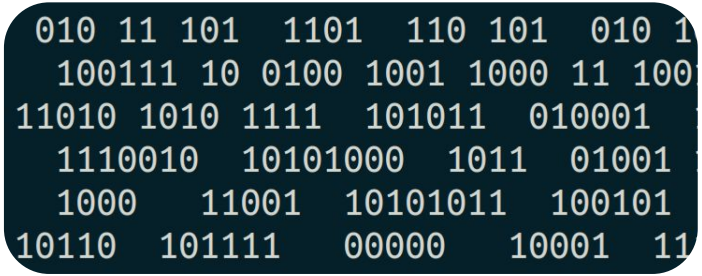
Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

Set of outputs without msk

Constrained Pseudorandom Functions (CPRFs)[BW13,KPTZ13,BGI14]

Pseudorandom Functions *with constrained access to the evaluation.*

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

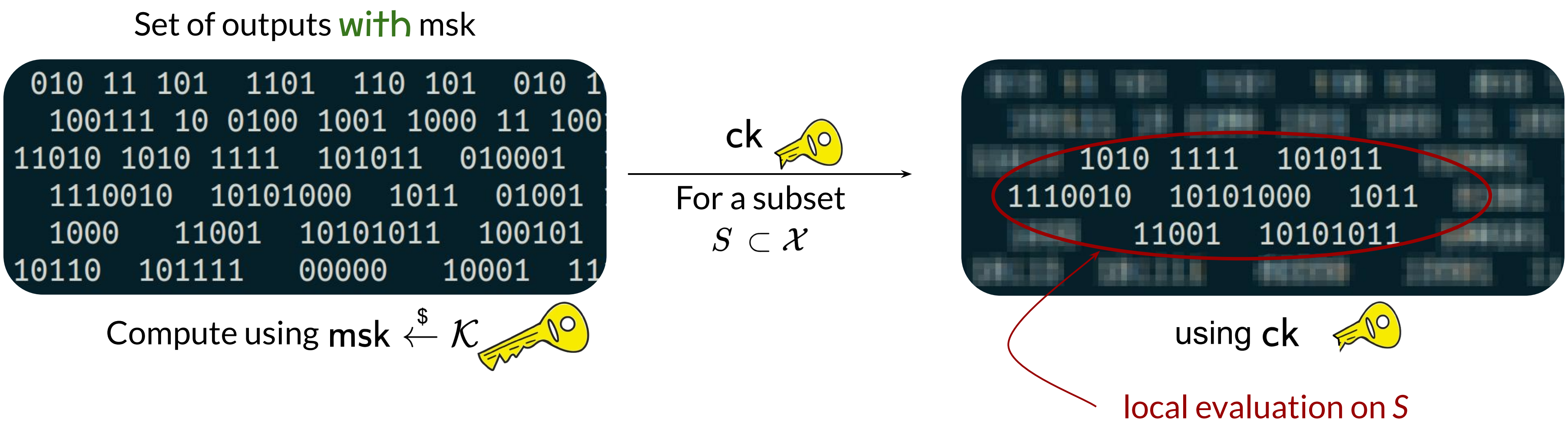
ck 

For a subset $S \subset \mathcal{X}$

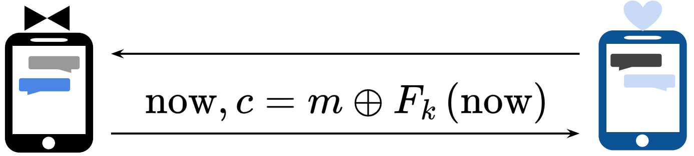
Set of outputs without msk

Constrained Pseudorandom Functions (CPRFs) [BW13,KPTZ13,BGI14]

Pseudorandom Functions *with constrained access to the evaluation.*



Application

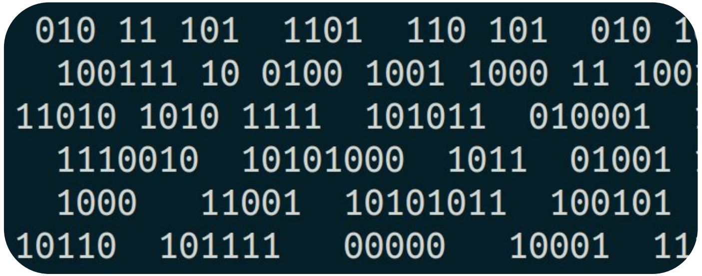


$\mathrm{now}, c = m \oplus F_k(\mathrm{now})$

m

$m \leftarrow c \oplus F_k(\mathrm{now})$

Set of outputs with msk

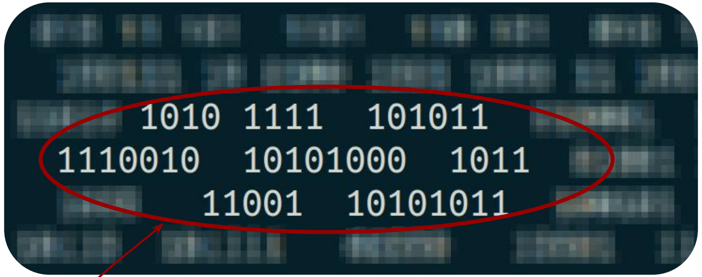


Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

ck

For a subset

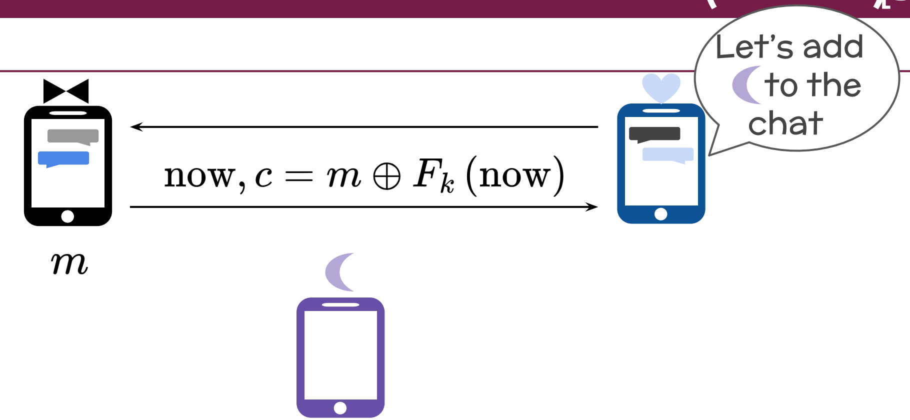
$S \subset \mathcal{X}$



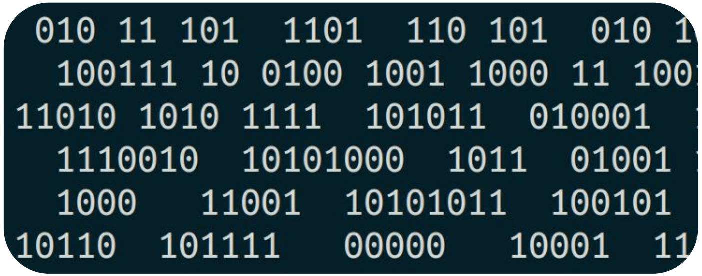
using **ck**

local evaluation on *S*

Application



Set of outputs with msk

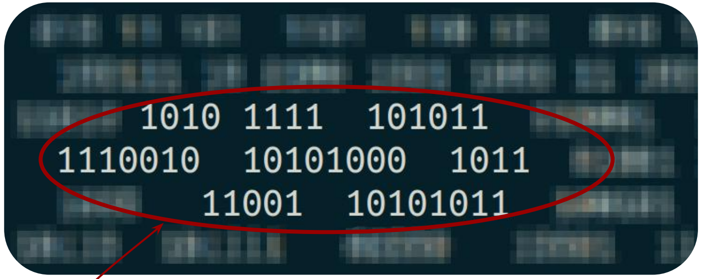


Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

ck

For a subset

$S \subset \mathcal{X}$

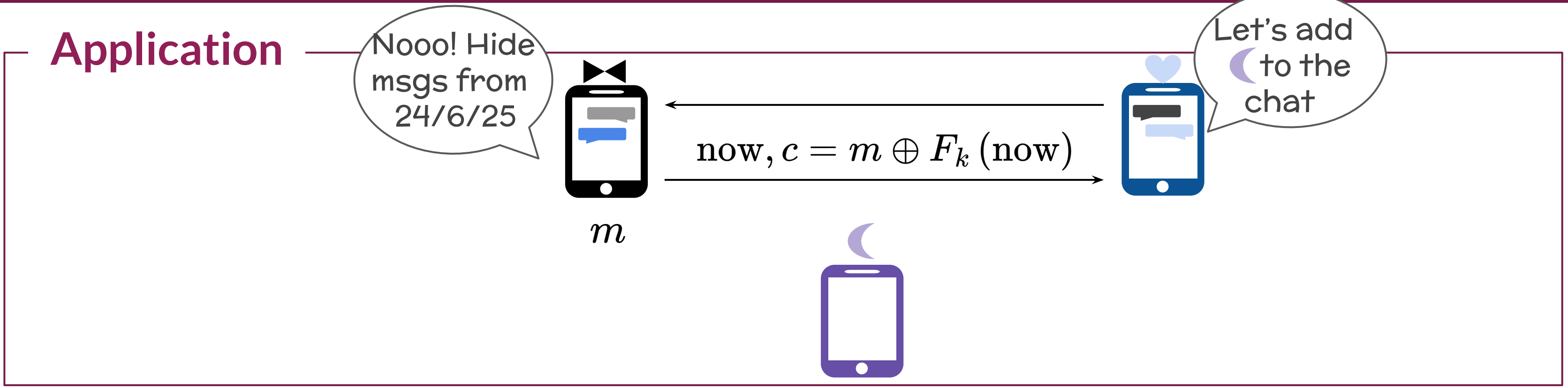


using **ck**

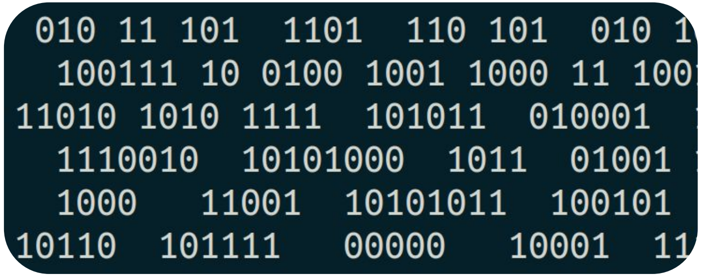
local evaluation on *S*

Constrained Pseudorandom Functions (CPRFs) [BW13,KPTZ13,BGI14]

Application



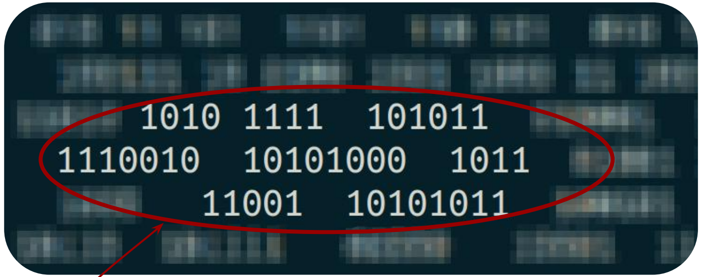
Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

ck

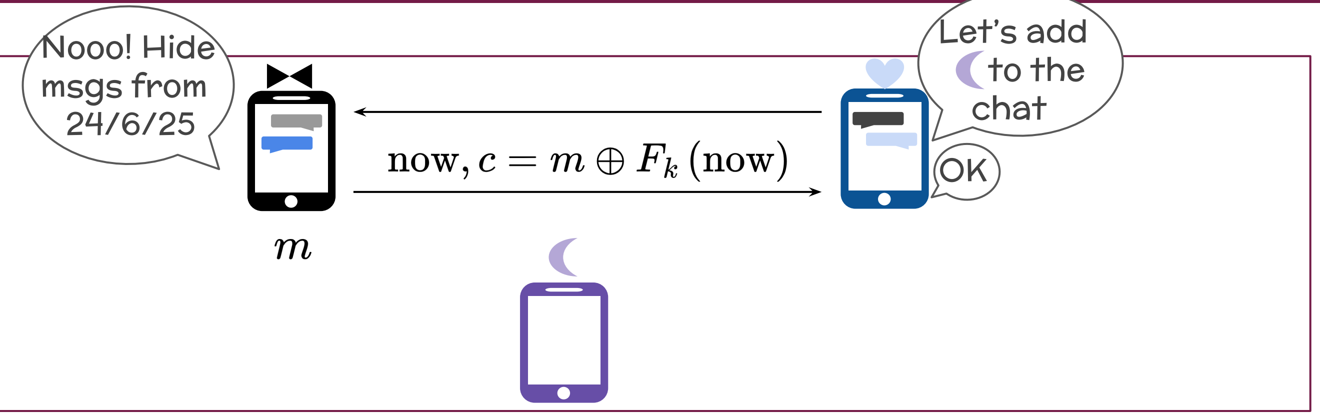
For a subset $S \subset \mathcal{X}$



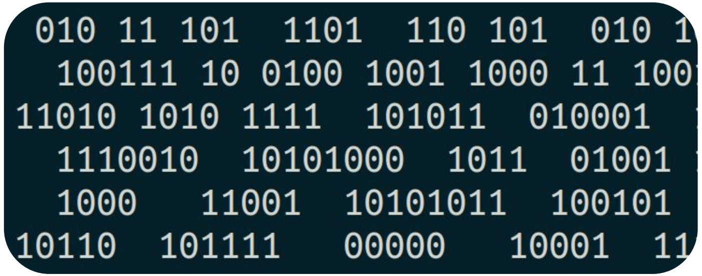
using **ck**

local evaluation on S

Application



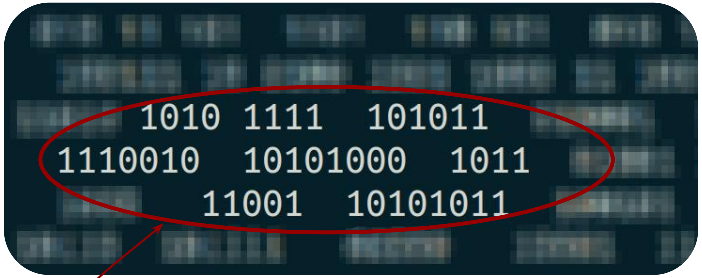
Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

ck

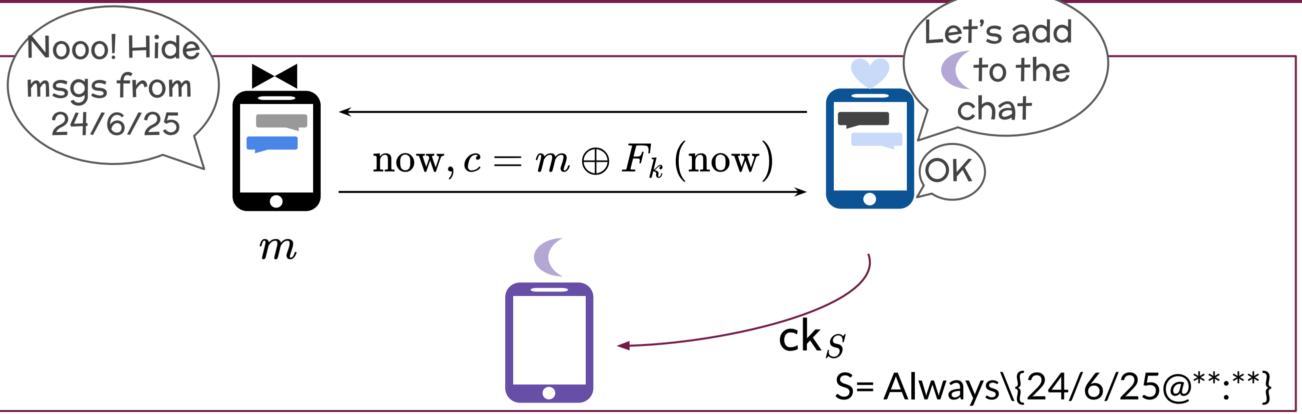
For a subset $S \subset \mathcal{X}$



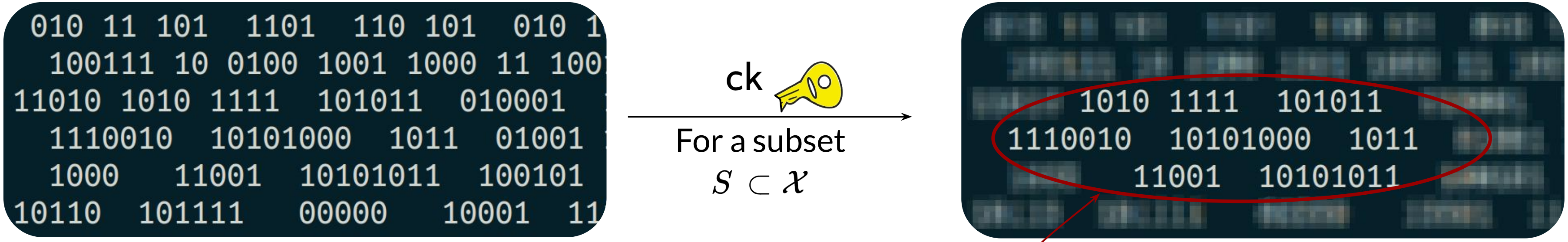
using **ck**

local evaluation on *S*

Application



Set of outputs with msk



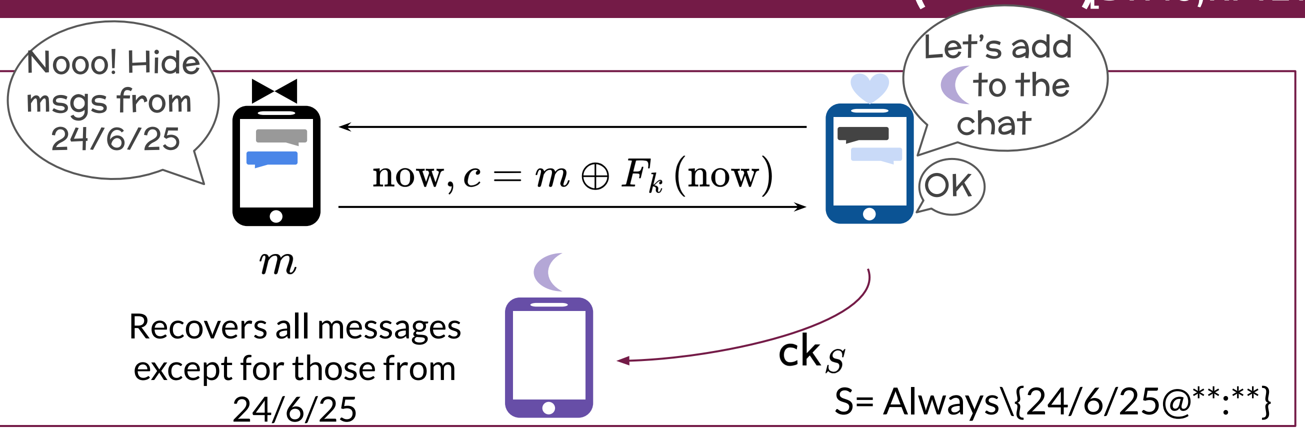
Compute using msk $\xleftarrow{\$} \mathcal{K}$

using ck

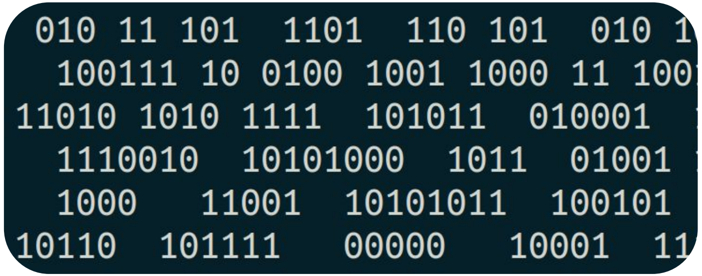
local evaluation on S

Constrained Pseudorandom Functions (CPRFs)[BW13,KPTZ13,BGI14]

Application

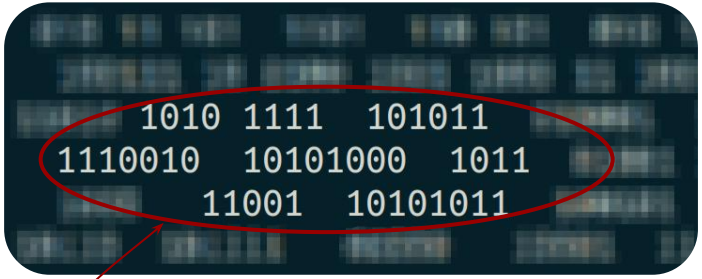


Set of outputs with msk



ck

For a subset $S \subset \mathcal{X}$



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$

using **ck**

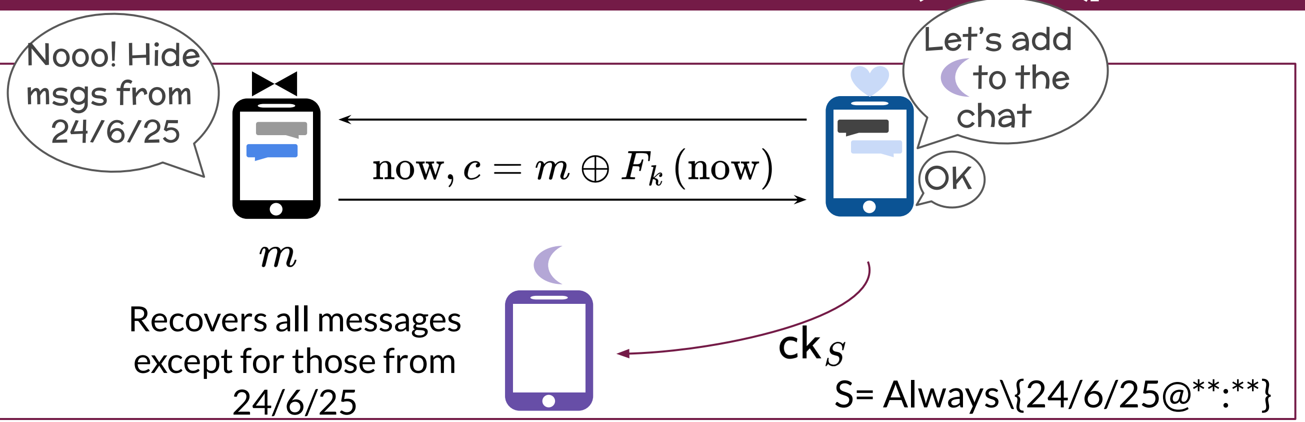
local evaluation on *S*

Constrained Pseudorandom Functions (CPRFs)[BW13,KPTZ13,BGI14]

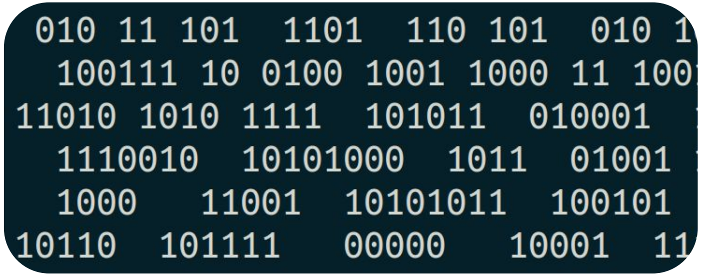
Application

Secret-Key
Attribute-Based
Encryption

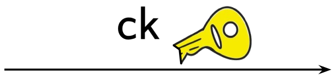
+ many more



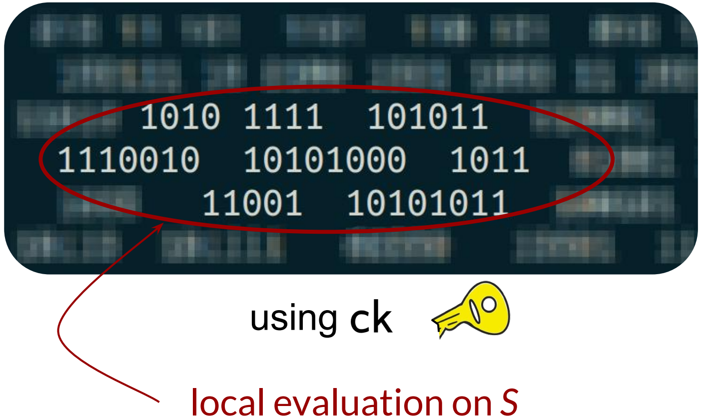
Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$



For a subset
$S \subset \mathcal{X}$



using **ck**

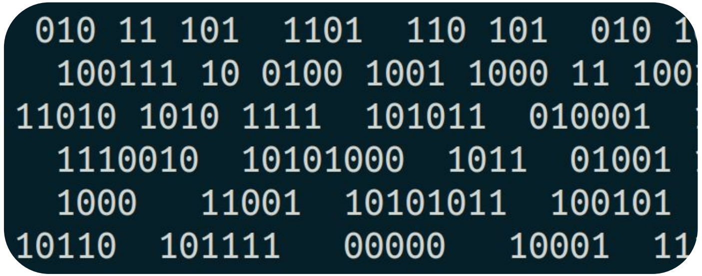
local evaluation on *S*

Constrained Pseudorandom Functions (CPRFs)[BW13,KPTZ13,BGI14]

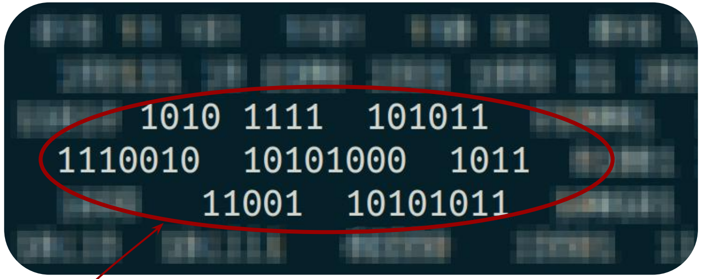
Pseudorandom Functions *with constrained access to the evaluation.*

- Every predicate $C : \mathcal{X} \to \{0,1\}$ defines a subset $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$ 

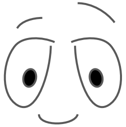


using **ck**$_C$ 

local evaluation on S_C

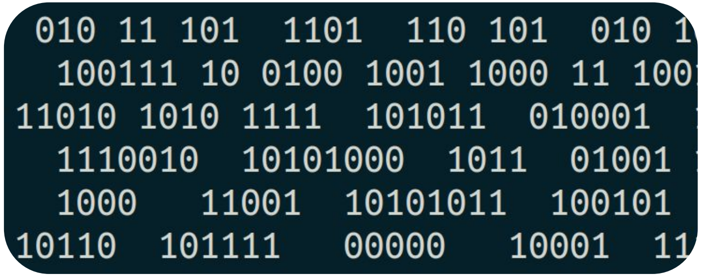
Pseudorandom Functions *with constrained access to the evaluation.*

- Every predicate $C : \mathcal{X} \to \{0,1\}$ defines a subset $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

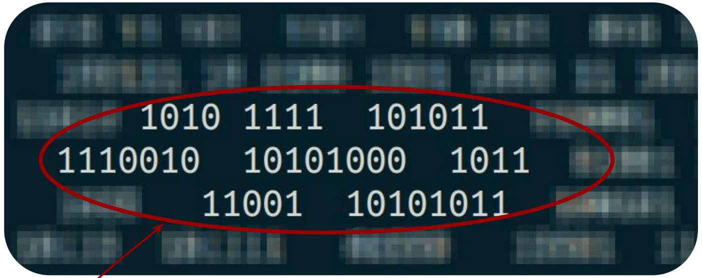


CPRFs that support expressive predicates ; P/Poly, NC^i

Set of outputs with msk



Compute using **msk** $\xleftarrow{\$} \mathcal{K}$



using **ck**$_C$

local evaluation on S_C

CPRFs meet Secure Computation

CPRFs meet Secure Computation

A Generic Idea

CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$

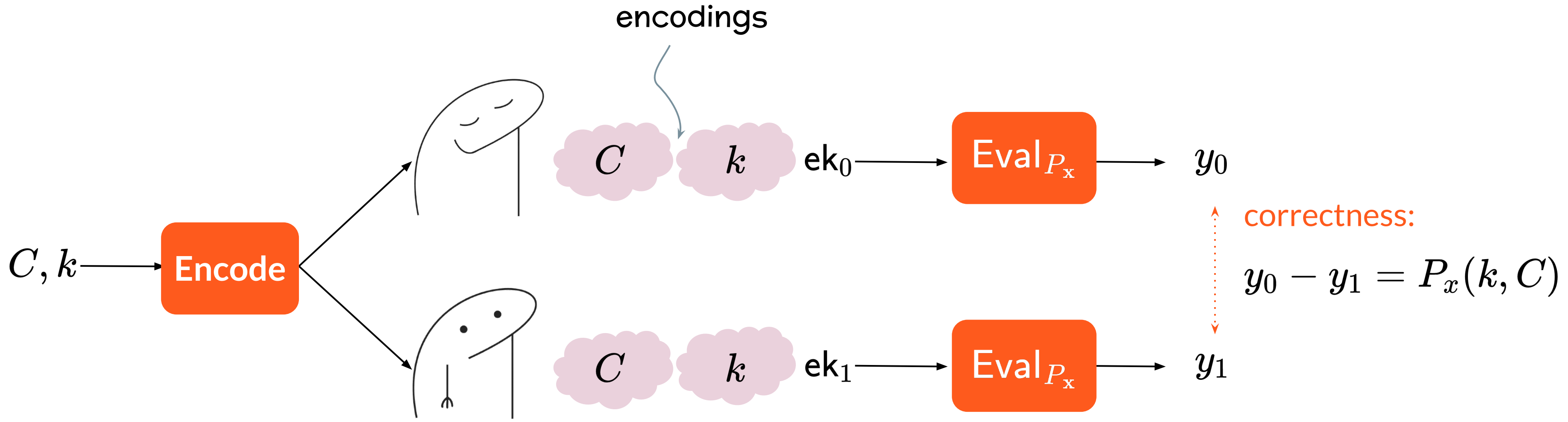
ck$_S$ can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

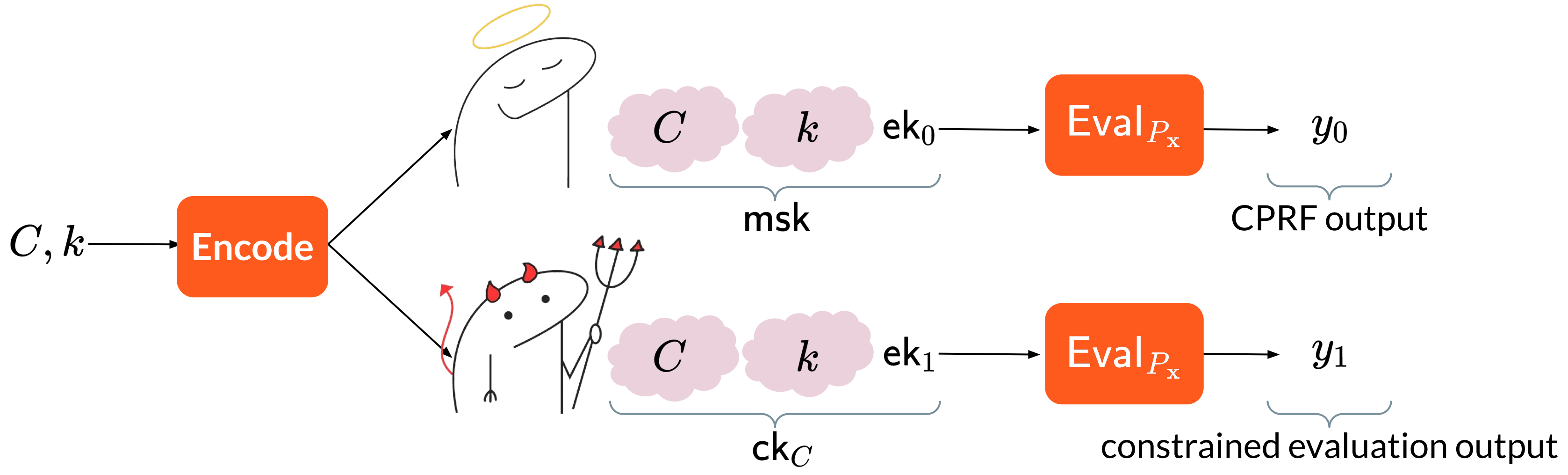
Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$



For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \; C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

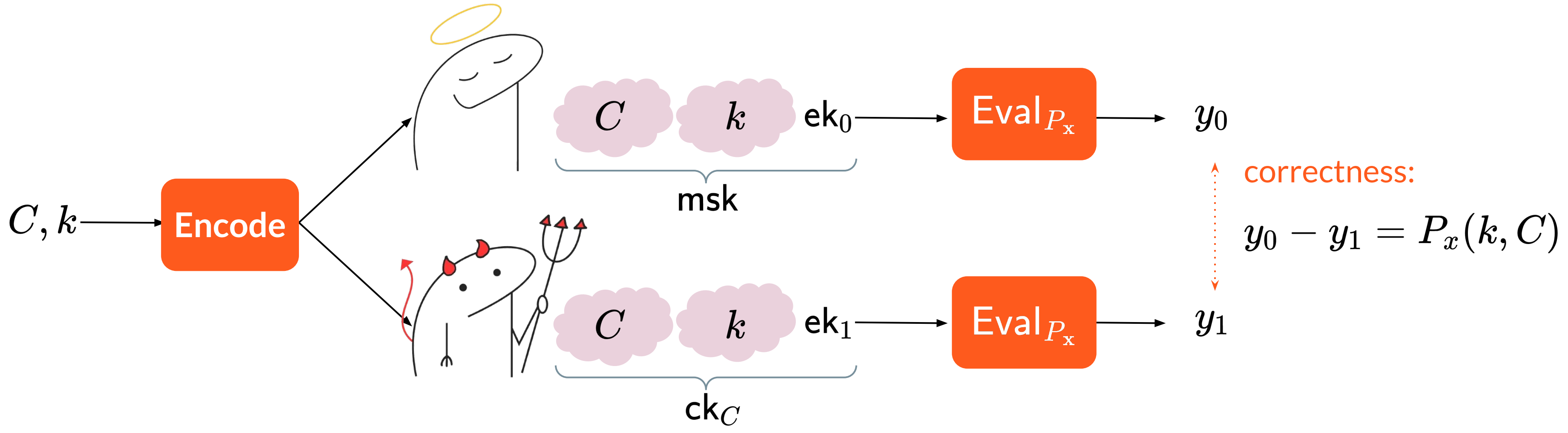


CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$



CPRFs meet Secure Computation

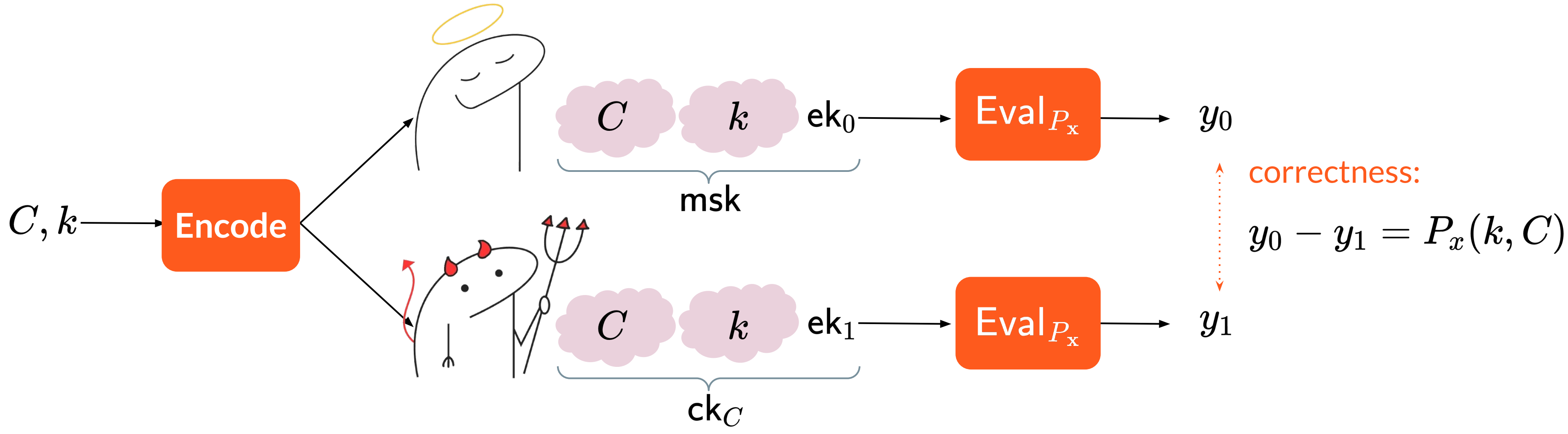
For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

idea

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



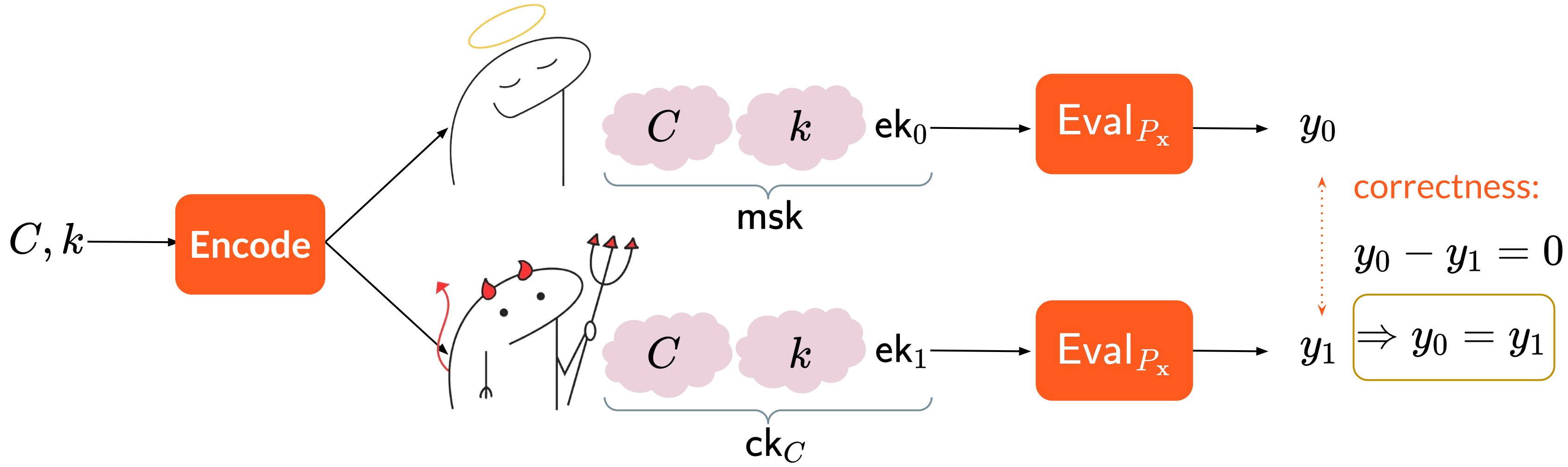
CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

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



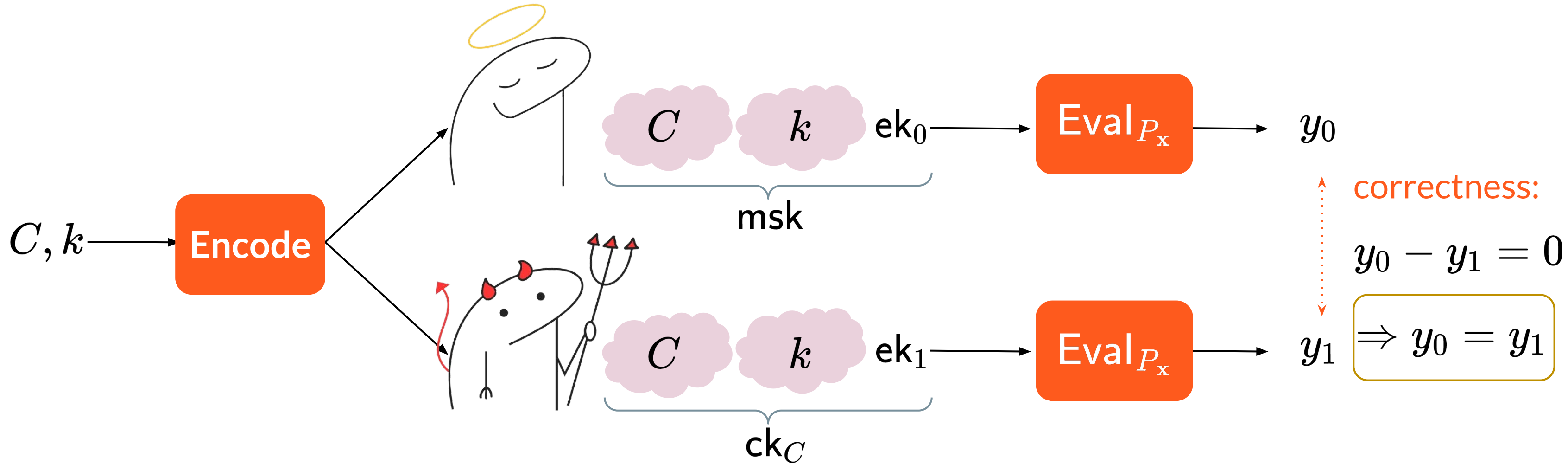
CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

$\mathsf{ck_S}$ can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

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



CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

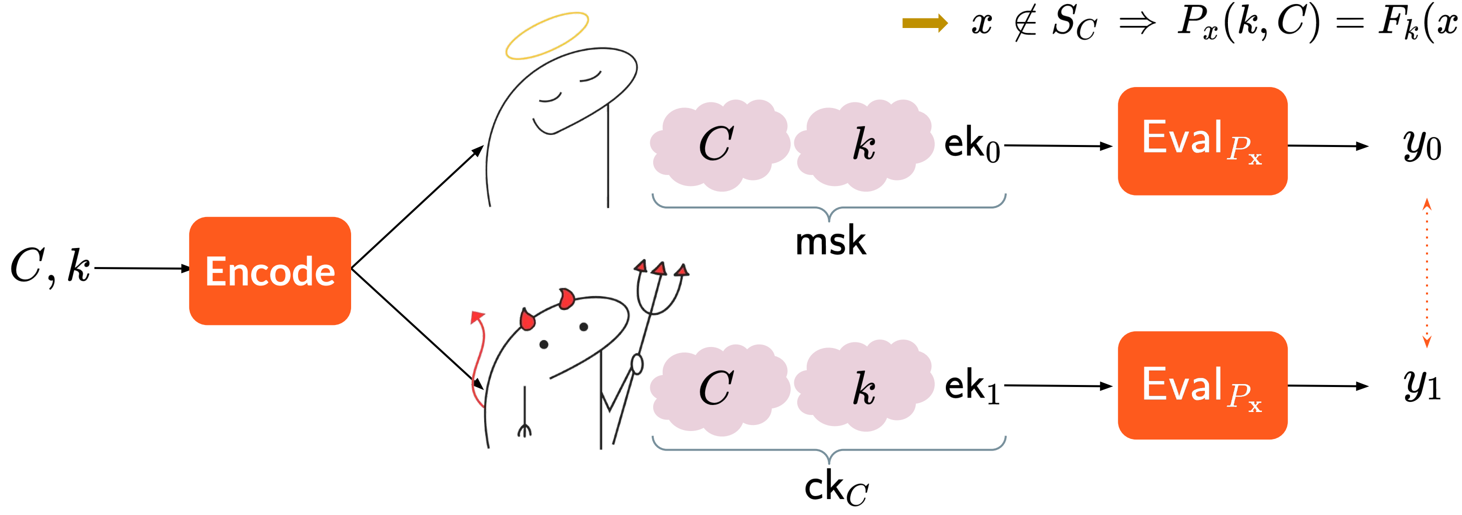
ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

idea

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

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



correctness:

$y_0 - y_1 = F_k(x)$

CPRFs meet Secure Computation

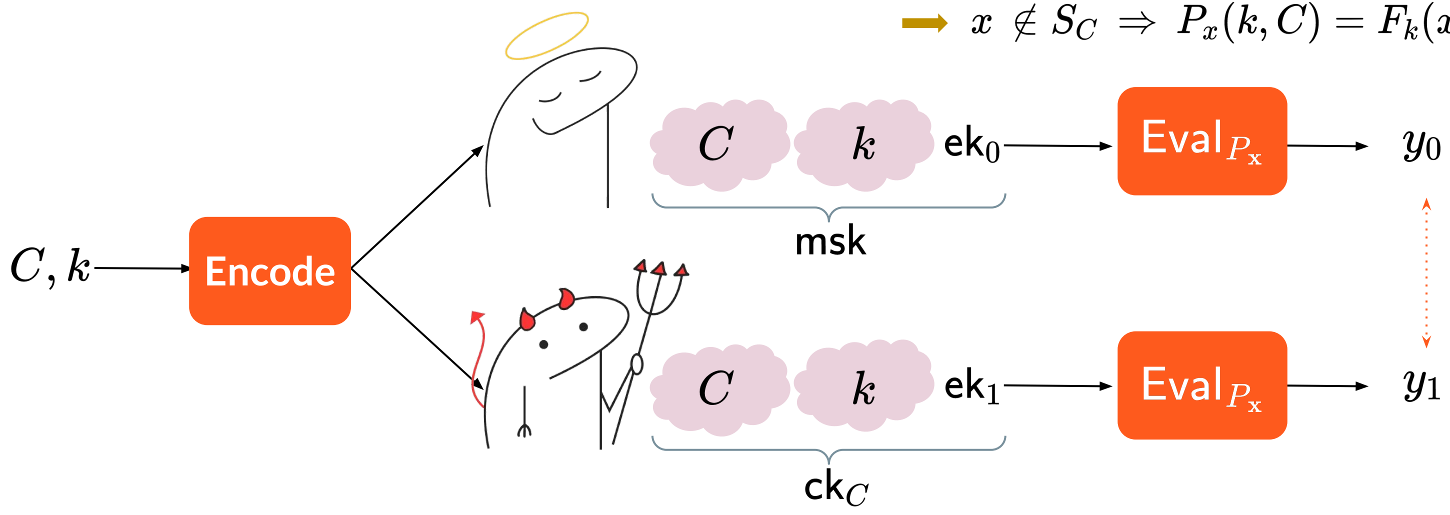
For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

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

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



correctness:

$y_0 - y_1 = F_k(x)$

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

CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

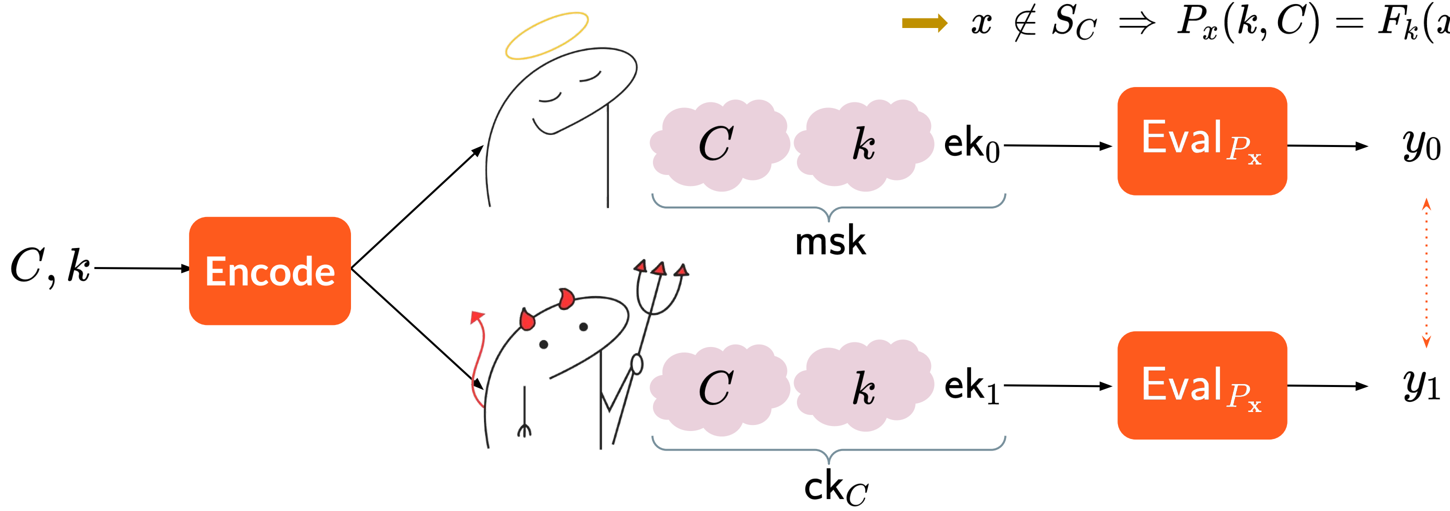
ck$_S$ can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

idea

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

$x \notin S_C \Rightarrow P_x(k, C) = F_k(x)$ → CPRF output looks random



correctness:

$y_0 - y_1 = F_k(x)$

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

CPRFs meet Secure Computation

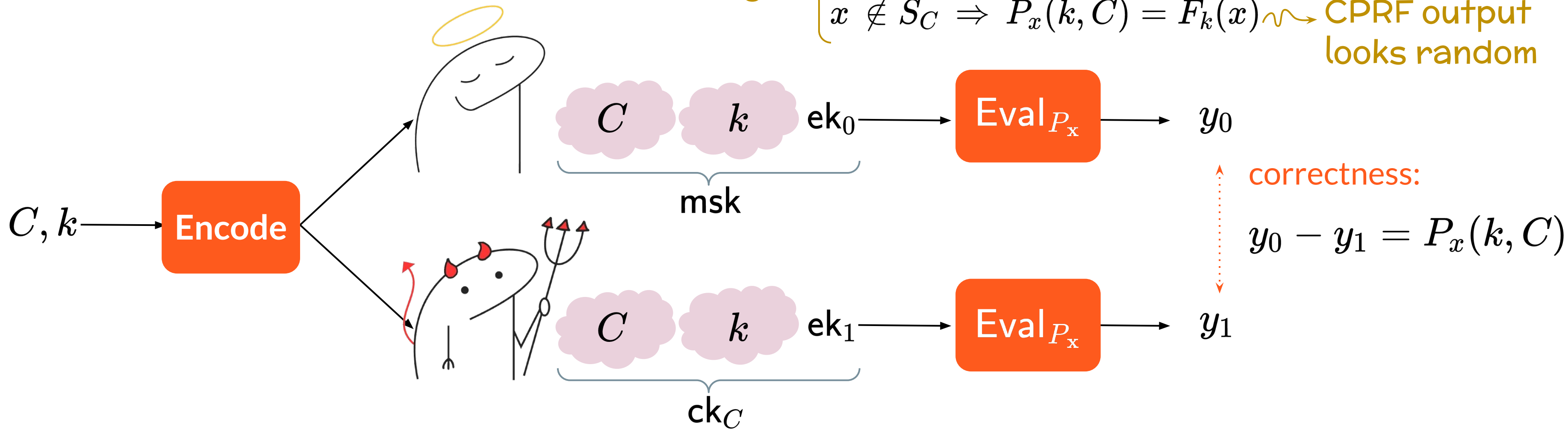
For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

$\mathrm{ck_S}$ can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

correctness + hiding k

$$\begin{cases} x \in S_C \Rightarrow P_x(k, C) = 0 & \text{Equal outputs} \\ x \notin S_C \Rightarrow P_x(k, C) = F_k(x) & \text{CPRF output looks random} \end{cases}$$



CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Instantiations

- Fully Homomorphic Encryption (based on LWE)
  [PS18] — *P/Poly constraints*

- Homomorphic Secret Sharing (from 5 separate assumptions)
  [CMP**R**23] — *NC^1 constraints*

CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : \ C(x) = 0\}$

ck_S can evaluate on S_C while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

Instantiations

- Fully Homomorphic Encryption (based on LWE)
  [PS18] — *P/Poly constraints*

- Homomorphic Secret Sharing (from 5 separate assumptions)
  [CMP**R**23] — *NC¹ constraints*

- **Receiver-Deniable** Fully Homomorphic Encryption **(Sparse LPN)**

  with P. Fallahpour, L. Kohl — *NC^1 constraints &* ***delegatable***

CPRFs meet Secure Computation

For a constraint C: X -> {0,1} : $S_C = \{x \in \mathcal{X} : C(x) = 0\}$

ck_S can evaluate on S_G while learning nothing about the output outside of it.

Take a PRF F with key k, and compute subtractive shares of $P_x : (k, C) \mapsto C(x) \cdot F_k(x)$

idea

Instantiations

- Fully Homomorphic Encryption (based on LWE)
  [PS18] — *P/Poly constraints*
- Homomorphic Secret Sharing (from 5 separate assumptions)
  [CMP**R**23] — *NC¹ constraints*
- **Receiver-Deniable** Fully Homomorphic Encryption **(Sparse LPN)**

with P. Fallahpour, L. Kohl — *NC¹ constraints* & ***delegatable***

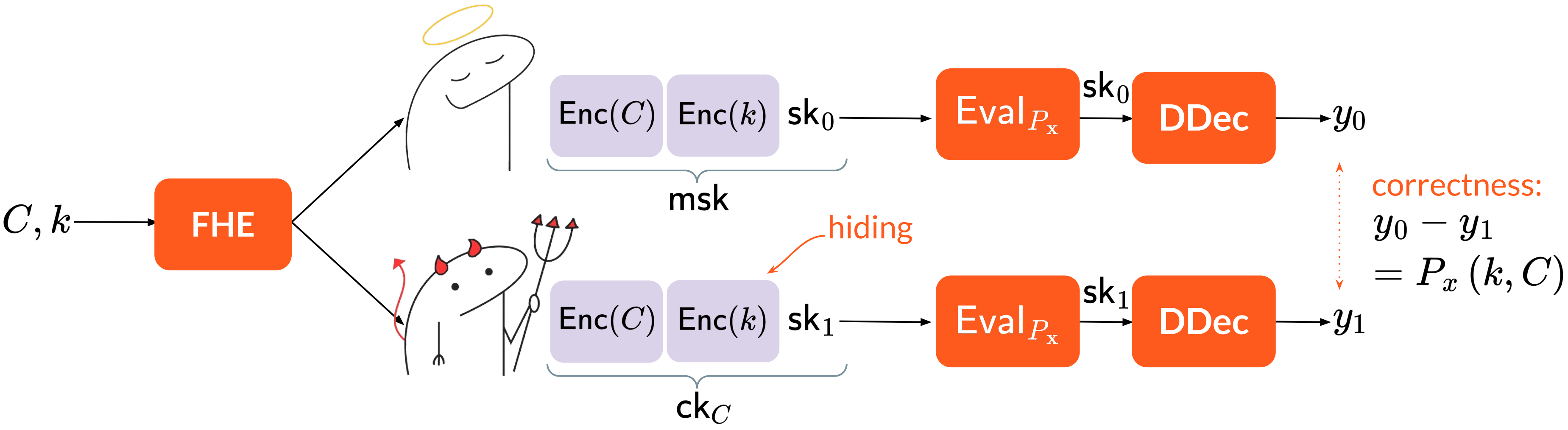
msk $\rightarrow$ ck$_{C_1}$ $\rightarrow$ ck$_{C_2}$ $\rightarrow \ldots$ $\quad \ldots \subseteq S_{C_2} \subseteq S_{C_1}$

CPRFs from FHE

Want: FHE with **distributed decryption**

$sk_0 - sk_1 = sk$ —> $DDec(ct, sk_0) - DDec(ct, sk_1) = DDec(ct, sk)$

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



CPRFs meet Secure Computation

Receiver-Deniable
FHE

Receiver-Deniable FHE

Definition (FHE). Computation can be performed over encryted data.

P : program

$$\mathsf{Enc}(m) \longrightarrow \mathsf{Eval}_P \longrightarrow \mathsf{Enc}(P(m))$$

CPRFs from FHE

Want: FHE with **distributed decryption**

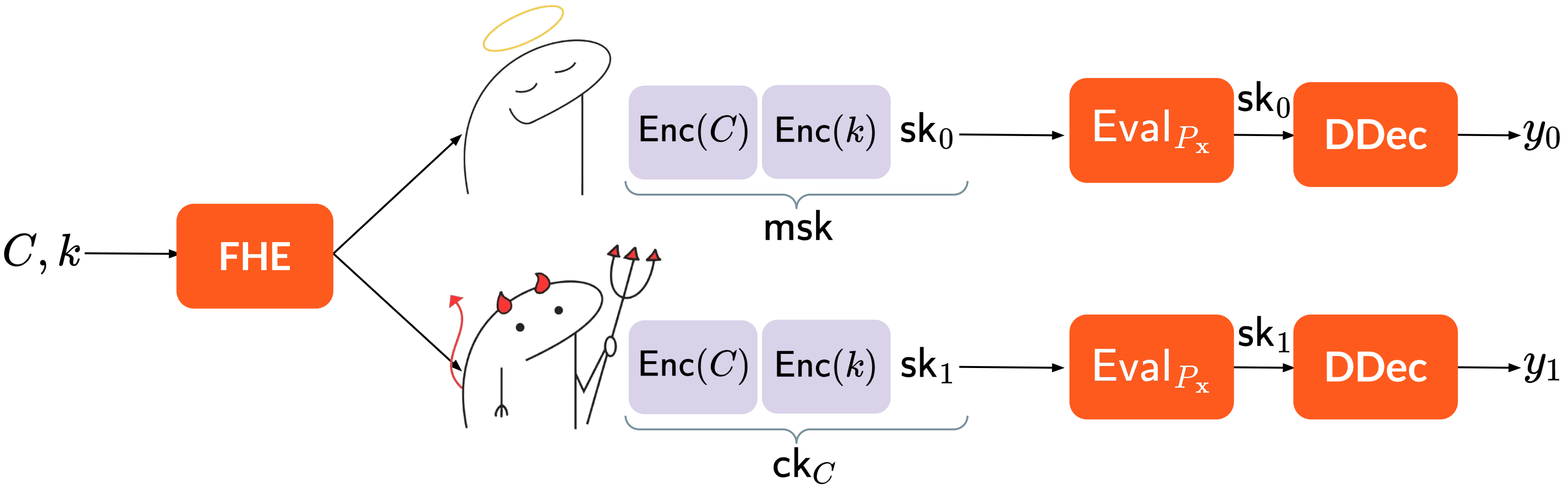
$sk_0 - sk_1 = sk \quad \longrightarrow \quad \mathrm{DDec}(ct, sk_0) - \mathrm{DDec}(ct, sk_1) = \mathrm{DDec}(ct, sk)$

CPRFs from FHE

Want: FHE with **distributed decryption**

sk$_0$ - sk$_1$ = sk —> DDec(ct, sk$_0$) - DDec(ct, sk$_1$) = DDec(ct, sk)

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

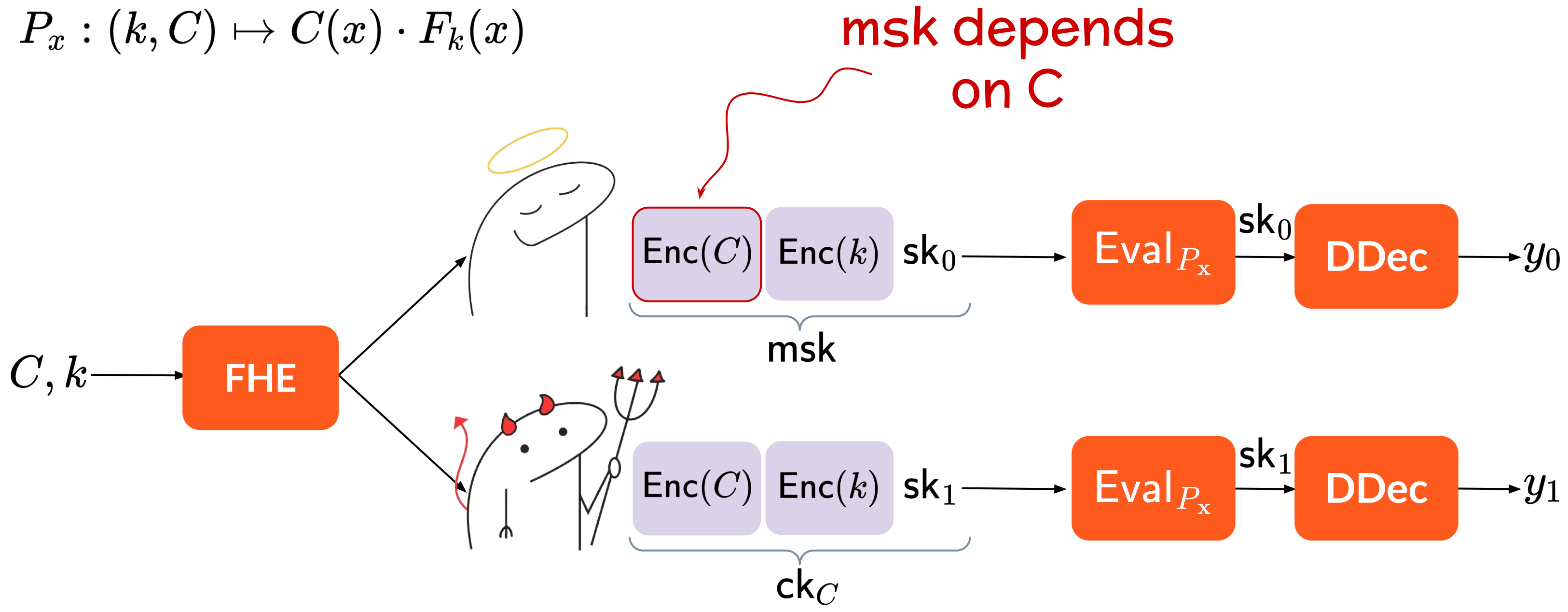


CPRFs from FHE

Want: FHE with **distributed decryption**

$sk_0 - sk_1 = sk$ —> $\mathrm{DDec}(ct, sk_0) - \mathrm{DDec}(ct, sk_1) = \mathrm{DDec}(ct, sk)$

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



Receiver-Deniable FHE

Definition (FHE). Computation can be performed over encryted data.

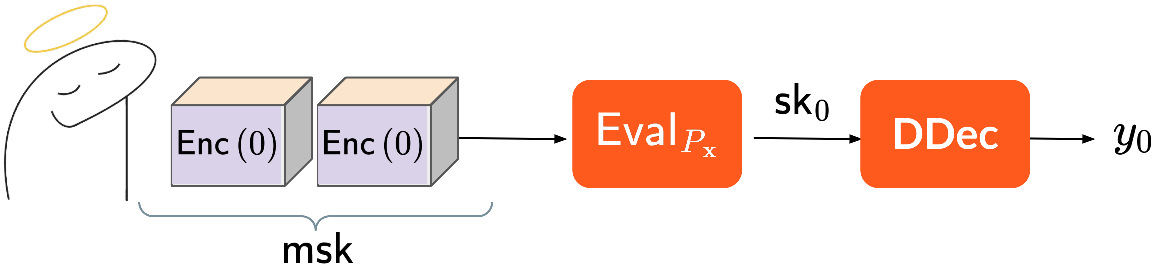
P : program

$\mathsf{Enc}\,(m) \longrightarrow \mathsf{Eval}_P \longrightarrow \mathsf{Enc}\,(P\,(m))$

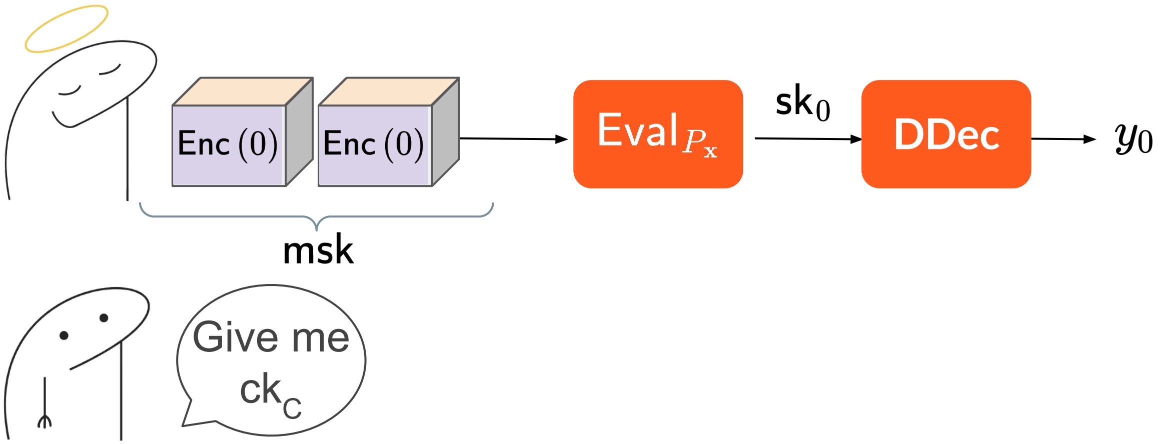
Receiver-Deniability

Given a ciphertext **ct** and message **m**, find **sk** that validates **ct** as encryption of **m**.

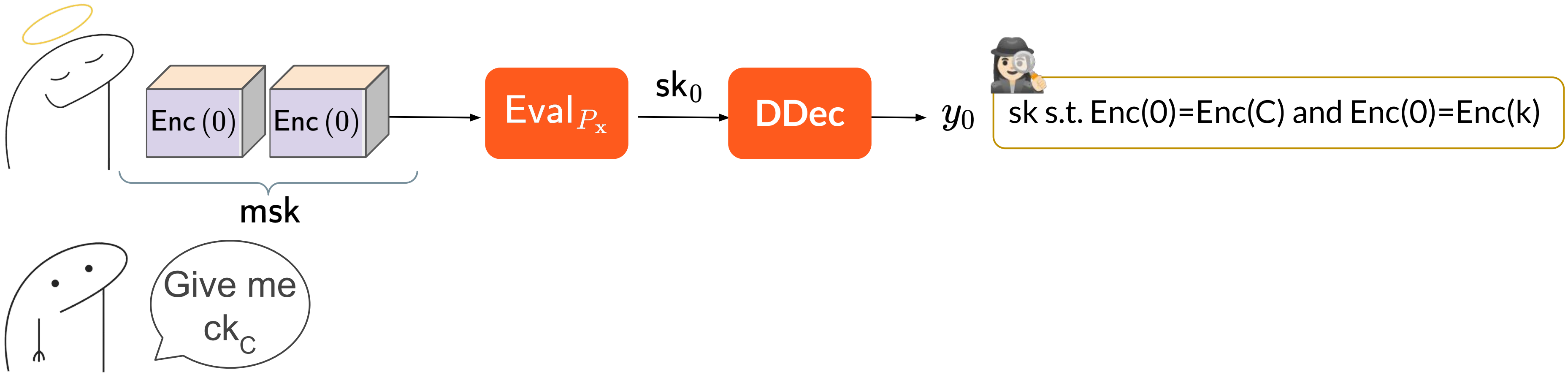
Constrained PRFs from Receiver-Deniable FHE



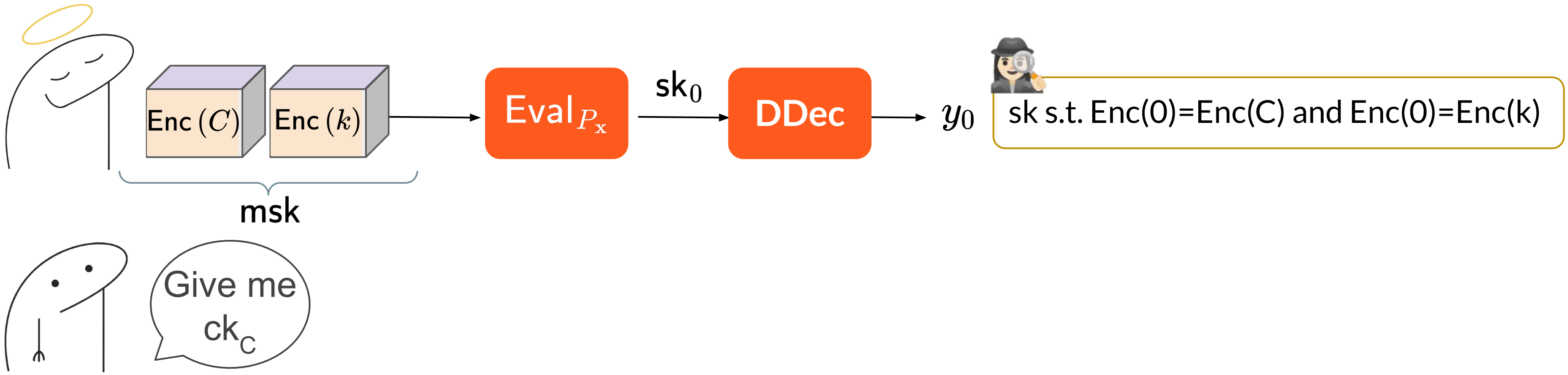
Constrained PRFs from Receiver-Deniable FHE



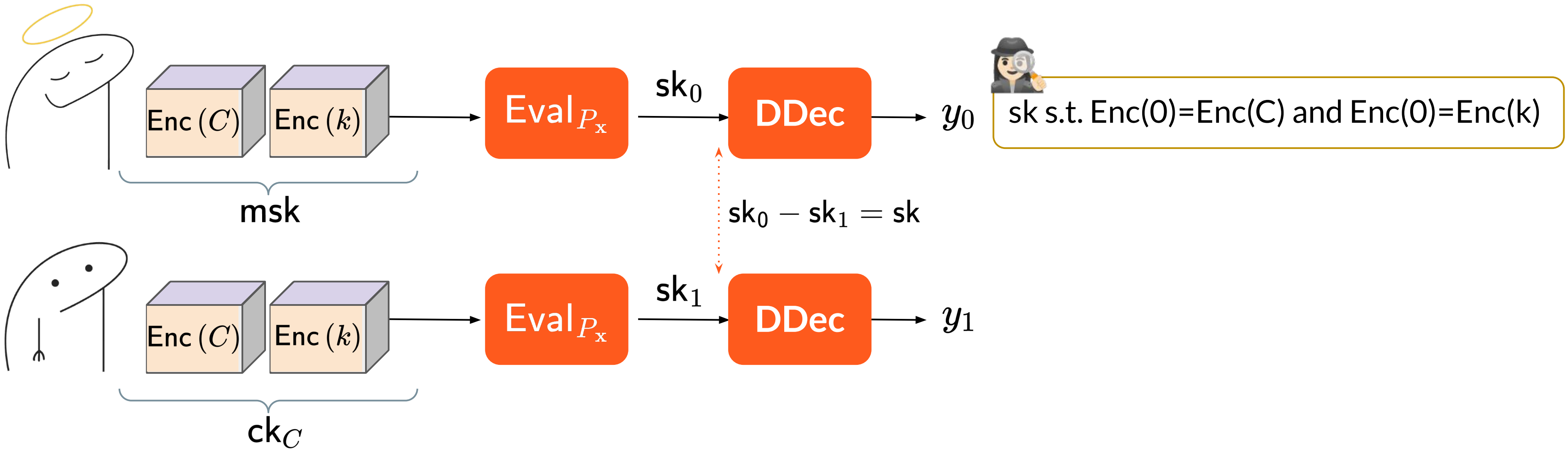
Constrained PRFs from Receiver-Deniable FHE



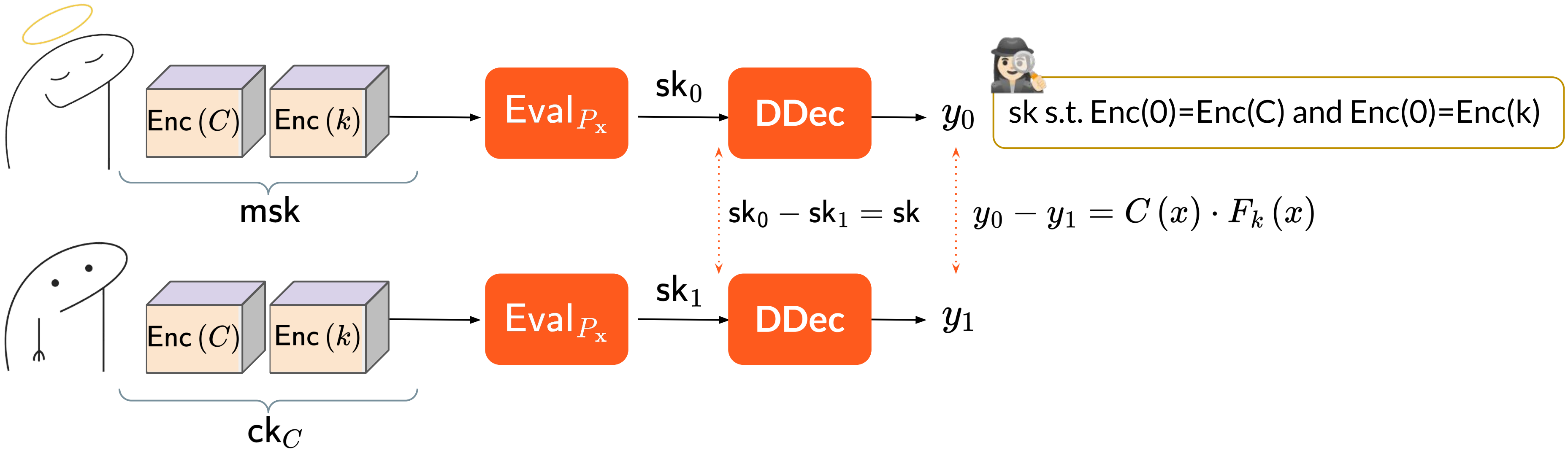
Constrained PRFs from Receiver-Deniable FHE



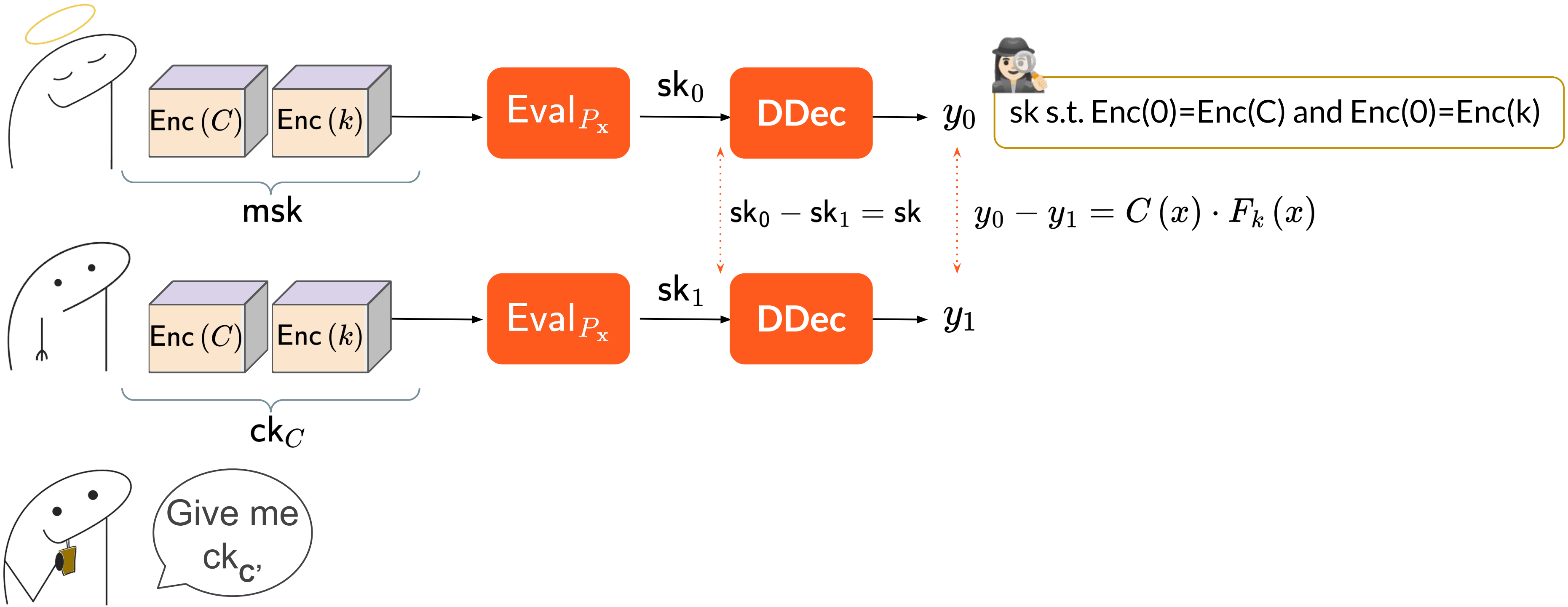
Constrained PRFs from Receiver-Deniable FHE



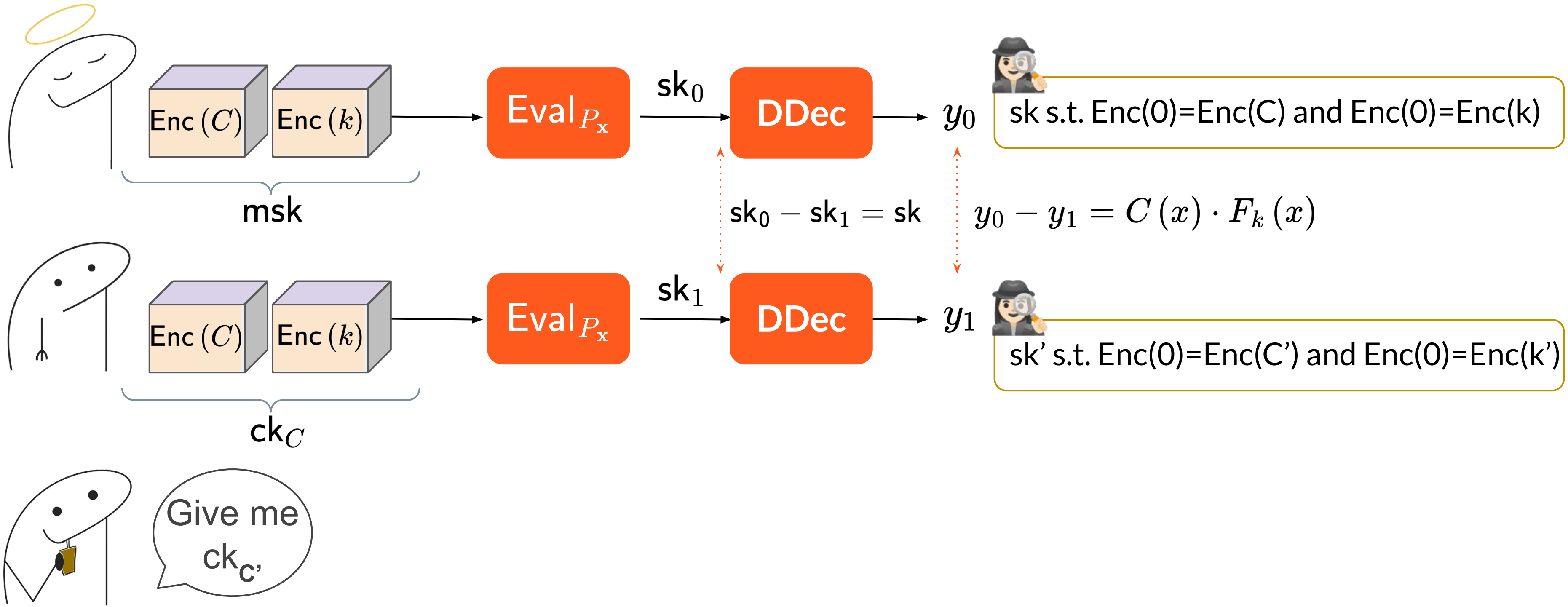
Constrained PRFs from Receiver-Deniable FHE



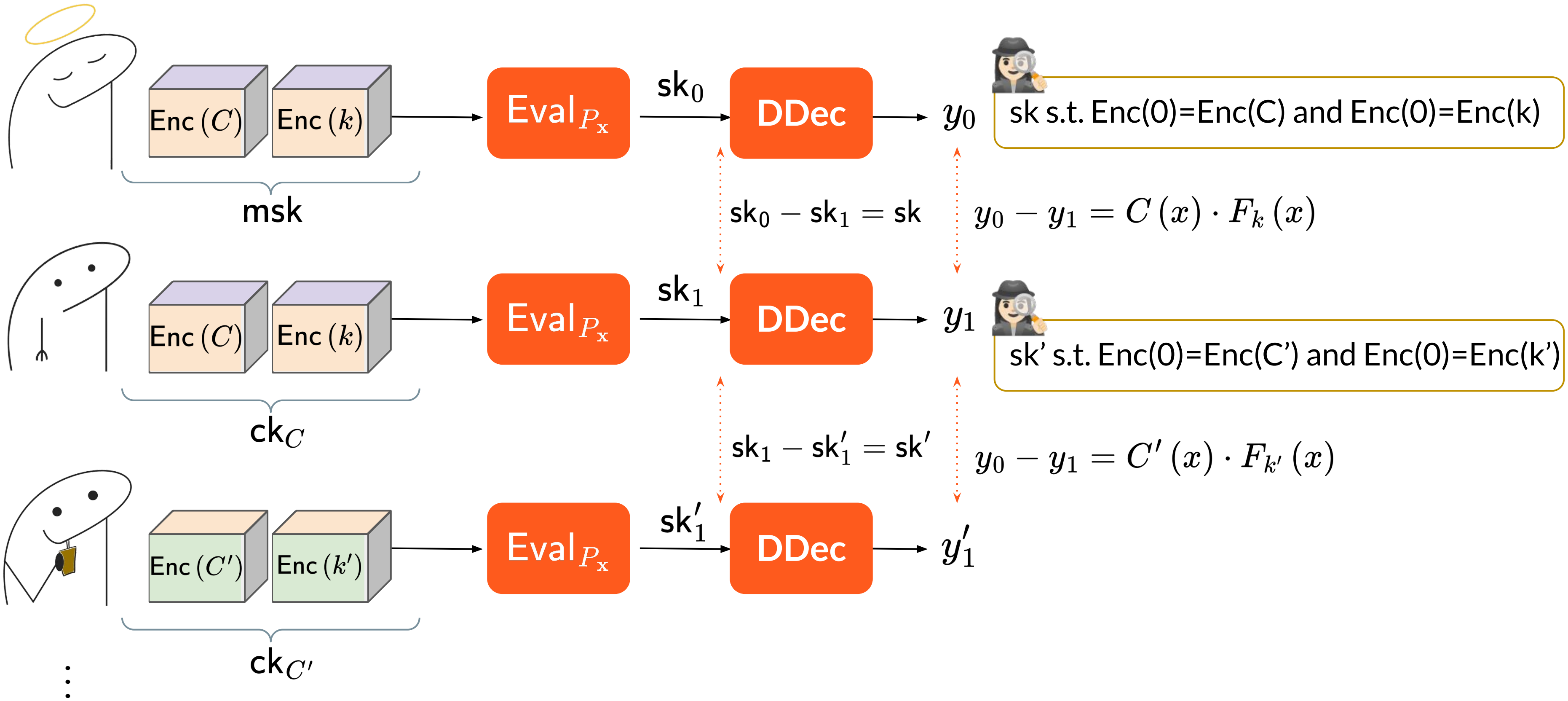
Constrained PRFs from Receiver-Deniable FHE



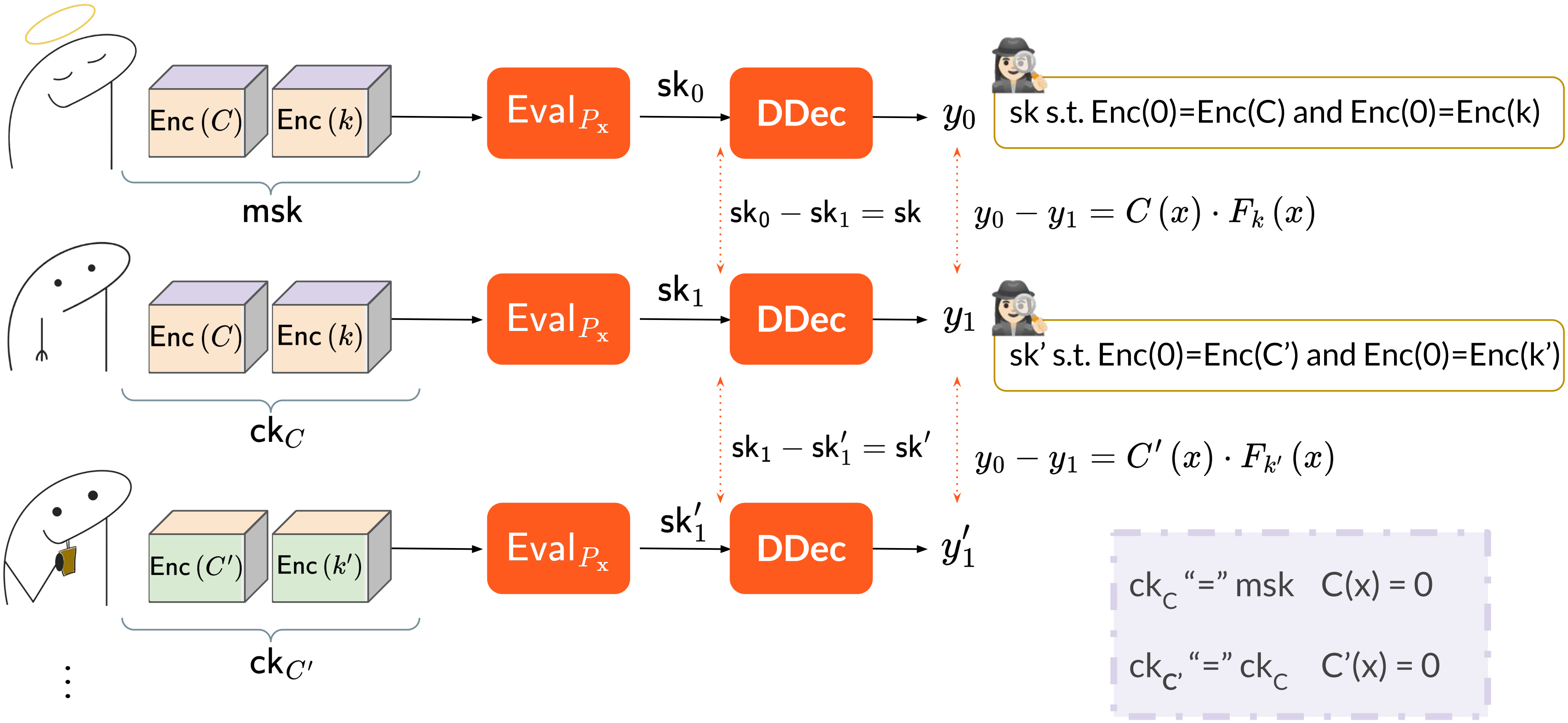
Constrained PRFs from Receiver-Deniable FHE



Constrained PRFs from Receiver-Deniable FHE



Constrained PRFs from Receiver-Deniable FHE



Conclusion

Homomorphic Secret Sharing

5 new constructions

Fully-Homomorphic Encryption

delegatable

inner-product & NC^1

Constrained PRFs

pseudorandom predicates

(public) puncturing

construction $\times 10^4$ faster than others

public-key **Pseudorandom Correlation Functions**

for oblivious transfer correlations

Pseudorandom Correlation Functions

for VOLE correlations

Conclusion

Homomorphic Secret Sharing

5 new constructions

inner-product & NC^1

delegatable

Fully-Homomorphic Encryption

Constrained PRFs

Thank You!

pseudorandom predicates

(public) puncturing

construction x10^4 faster than others

public-key **Pseudorandom Correlation Functions**

for oblivious transfer correlations

Pseudorandom Correlation Functions

for VOLE correlations

References

[BGI14] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions.

[BW13] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications.

[CMPR23] G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. Constrained Pseudorandom Functions from Homomorphic Secret Sharing.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions.

[KPTZ13] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications.

[PS18] C. Peikert, S. Shiehian. Privately Constraining and Programming PRFs, the LWE Way.

Helper Slides

CPRFs: State-of-the-Art

Assumption	What is known		
Lattice-Based	P/poly	NC^1	Inner-Product
	✓ *Hiding*	✓ *Hiding*	✓ *Hiding & Adaptive*
Group-Based	✗	✓	
Heavy Tools (iO,multilinear maps)	✓ *Hiding, Adaptive & Multi-Key*	✓ *Adaptive*	

Not at the same time