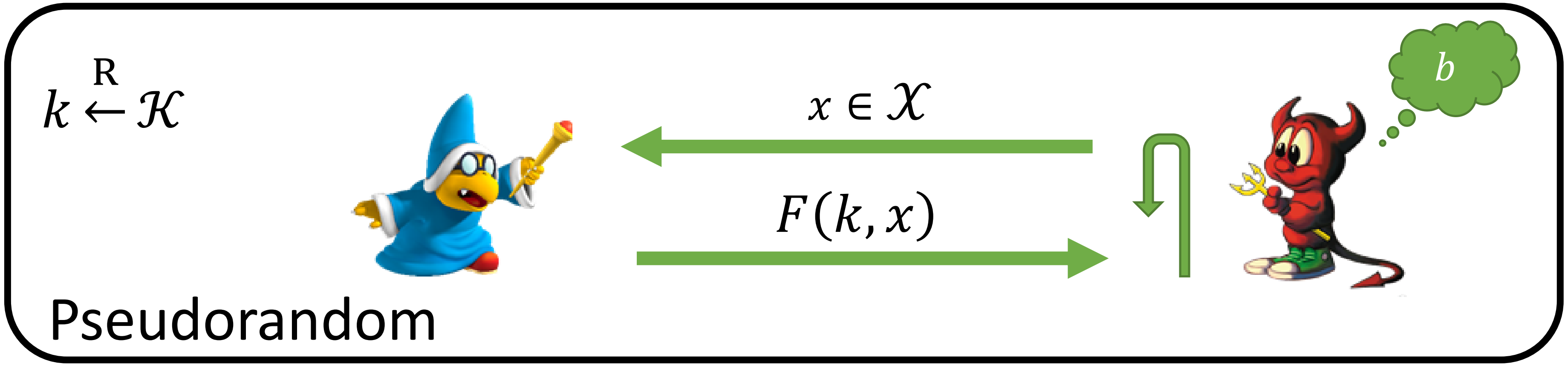
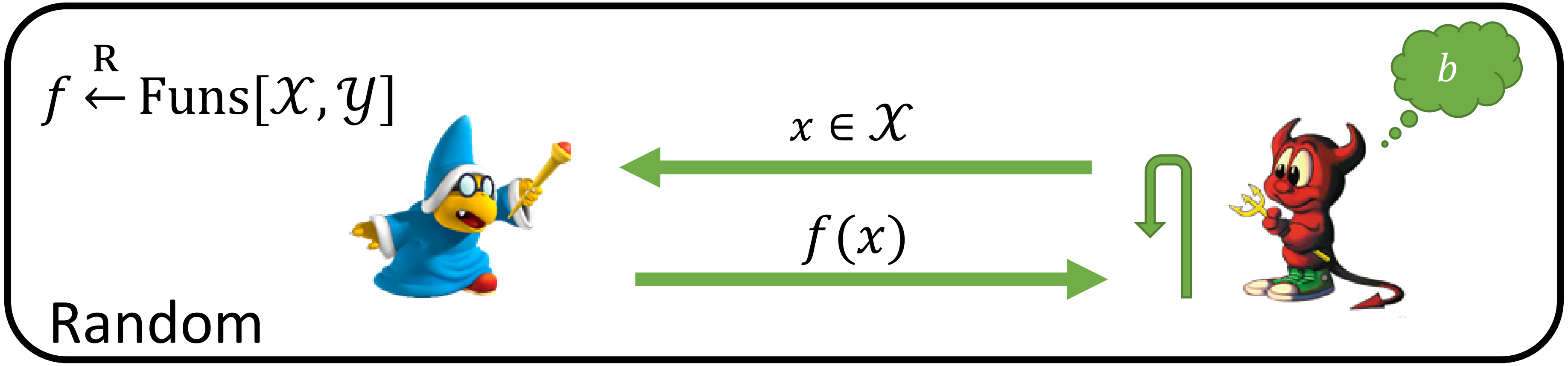
Constrained Keys for Invertible Pseudorandom Functions

Dan Boneh, Sam Kim, and David J. Wu
Stanford University

Pseudorandom Functions (PRFs) [GGM84]



$$\approx_c$$



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

Constrained PRFs [BW13, BGI13, KPTZ13]

Constrained PRF: PRF with additional "constrain" functionality

PRF key

Constrained key

Can be used to evaluate at all points $x \in \mathcal{X}$ where $C(x) = 1$

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

Constrained PRFs [BW13, BGI13, KPTZ13]

<u>Correctness</u>: constrained evaluation at $x \in \mathcal{X}$ where $C(x) = 1$ yields PRF value at x

<u>Security</u>: PRF value at points $x \in \mathcal{X}$ where $C(x) = 0$ are indistinguishable from random *given* the constrained key

Constrained PRFs [BW13, BGI13, KPTZ13]

Many applications:

- Punctured programming paradigm [SW14]
- Identity-based key exchange, broadcast encryption [BW13]
- Multiparty key exchange, traitor tracing [BZ14]

Constrained PRFs [BW13, BGI13, KPTZ13]

Known constructions:

- Puncturable PRFs from one-way functions [BW13, BGI13, KPTZ13]

Punctured key can be used to evaluate the PRF at all but one point

Constrained PRFs [BW13, BGI13, KPTZ13]

Known constructions:

- Puncturable PRFs from one-way functions [BW13, BGI13, KPTZ13]
- Circuit-constrained PRFs from LWE [BV15]

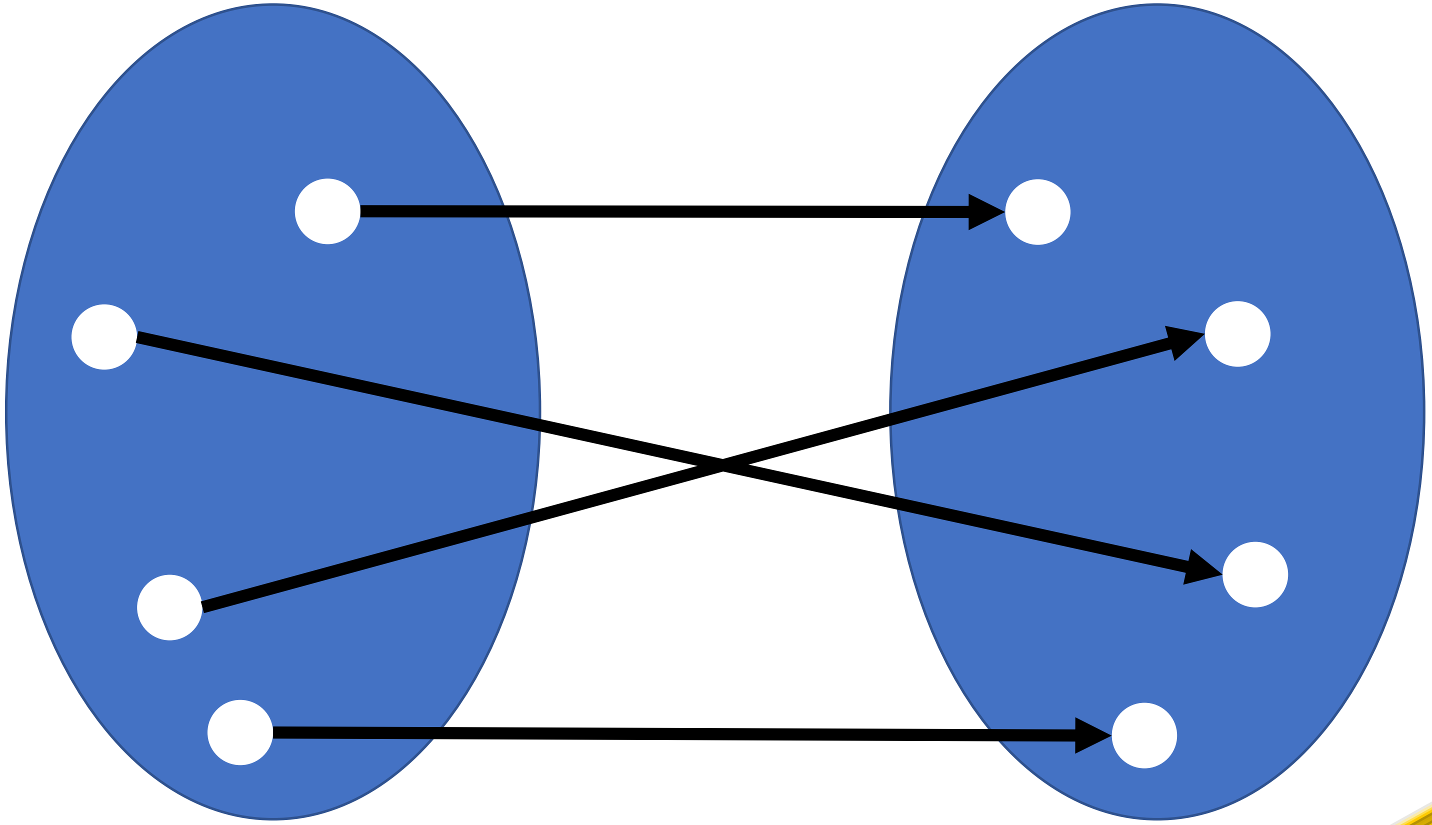
Can we constrain other cryptographic primitives, such as pseudorandom permutations (PRPs)?

Our Results

- Constrained PRPs for many natural classes of constraints *do not exist*

- However, the relaxed notion of a constrained *invertible pseudorandom function* (IPF) do exist

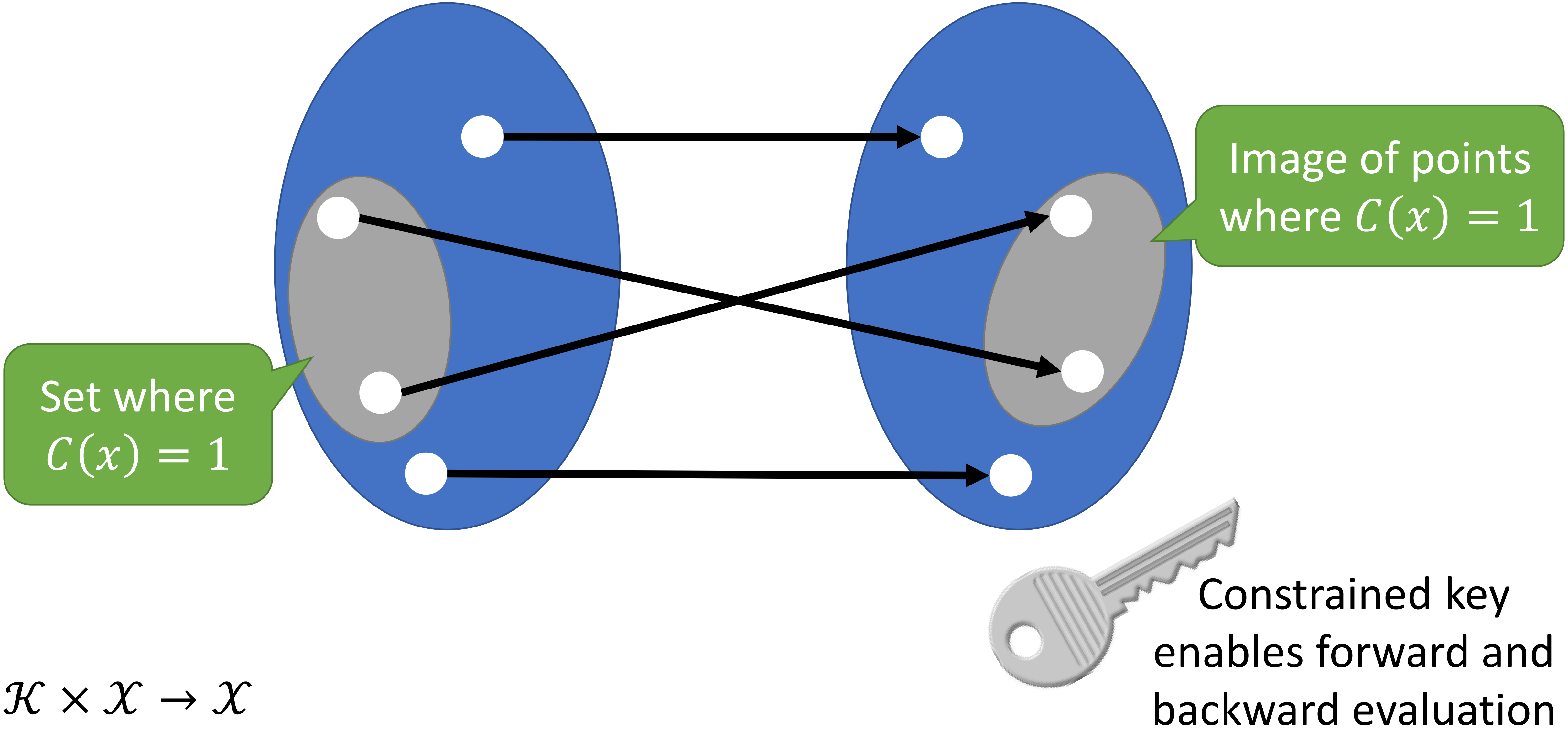
Constrained PRPs



$F(k,\cdot)$ implements a permutation over $\mathcal{X}$

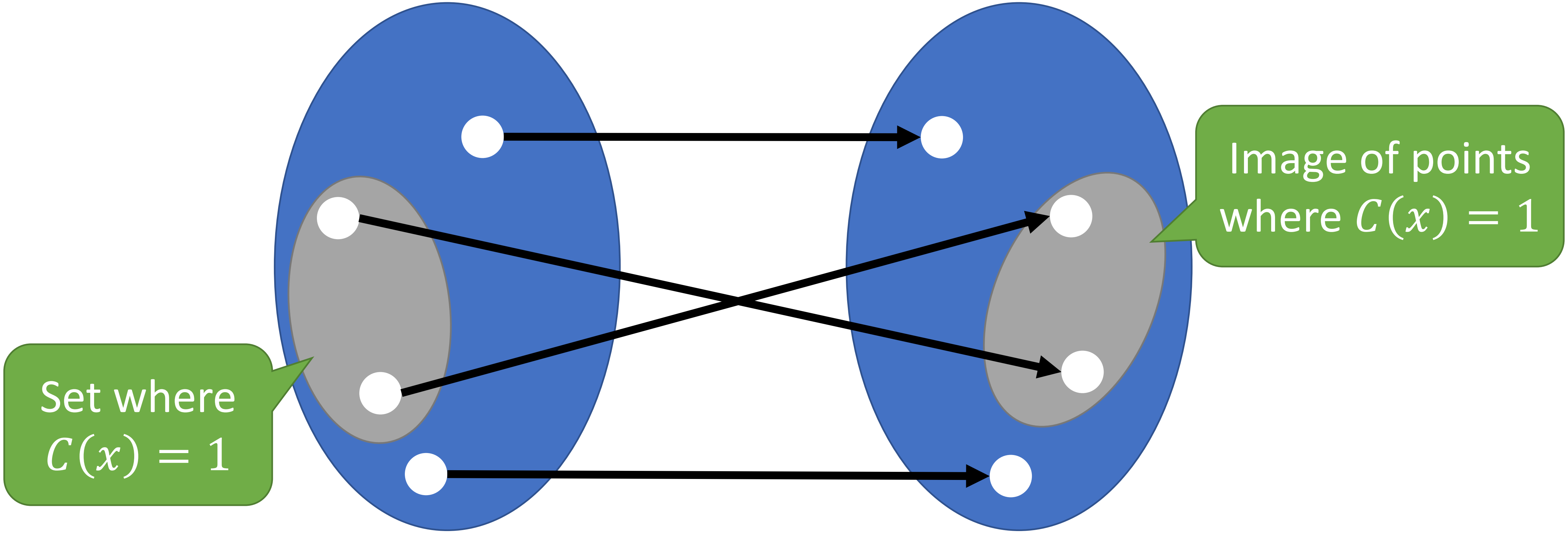
$F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$

Constrained PRPs



$F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$

Constrained PRPs



Correctness:

- Forward evaluation when $C(x) = 1$
- Backward evaluation on points y if $y = F(k, x)$ and $C(x) = 1$

Constrained PRP Security

$k \xleftarrow{R} \mathcal{K}$

$f \xleftarrow{R} \text{Perm}[\mathcal{X}]$

Challenger

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

Constrain queries

C

$k_C = \text{Constrain}_C(k)$

Evaluation queries

x

$F(k, x)$

Inversion queries

x

$F^{-1}(k, x)$

Challenge queries

x^*

y^*

Adversary

Random: $y^* = f(x^*)$

Pseudorandom: $y^* = F(k, x^*)$

Constrained PRP Security

Admissibility conditions:

- $C(x^*) = 0$
- No evaluation queries on x^*
- No inversion queries on y^*

C

$k_C = \text{Constrain}_C(k)$

x

$F(k, x)$

x

$F^{-1}(k, x)$

x^*

y^*

Adversary

Random: $y^* = f(x^*)$

Pseudorandom: $y^* = F(k, x^*)$

Constrained PRP Lower Bound

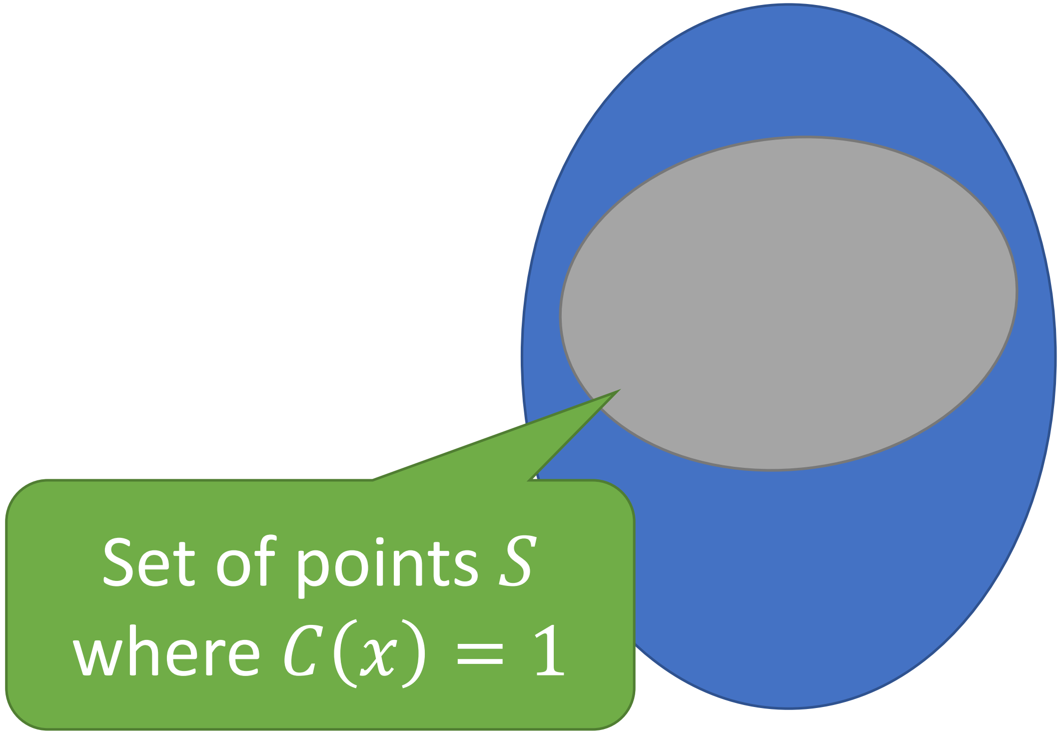
Warm-up: constrained PRPs on polynomial-size domains cannot satisfy constrained security

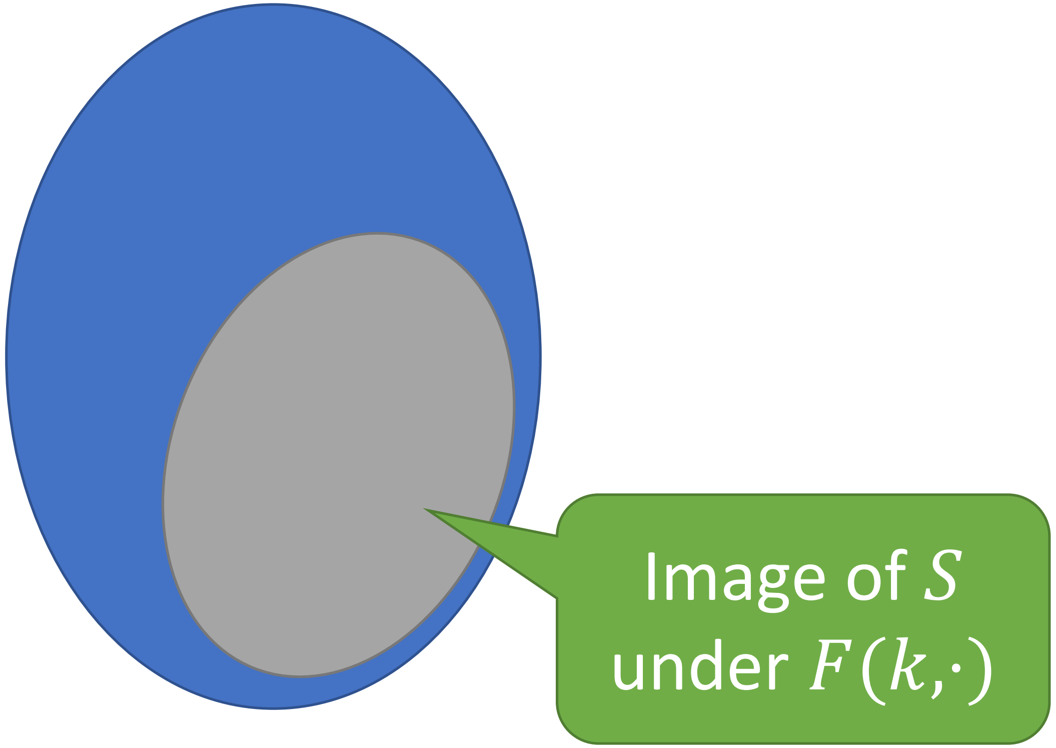
Concretely: evaluate PRP at x and issue challenge query for $x^* \neq x$

- Pseudorandom case: $F(k, x^*) \neq F(k, x)$
- Random case: $f(x^*) = F(k, x)$ with probability $1/|\mathcal{X}|$

Constrained PRP Lower Bound

Theorem (Informal). Any constrained PRP that allows issuing a constrained key that can evaluate on a non-negligible fraction of the domain is insecure.

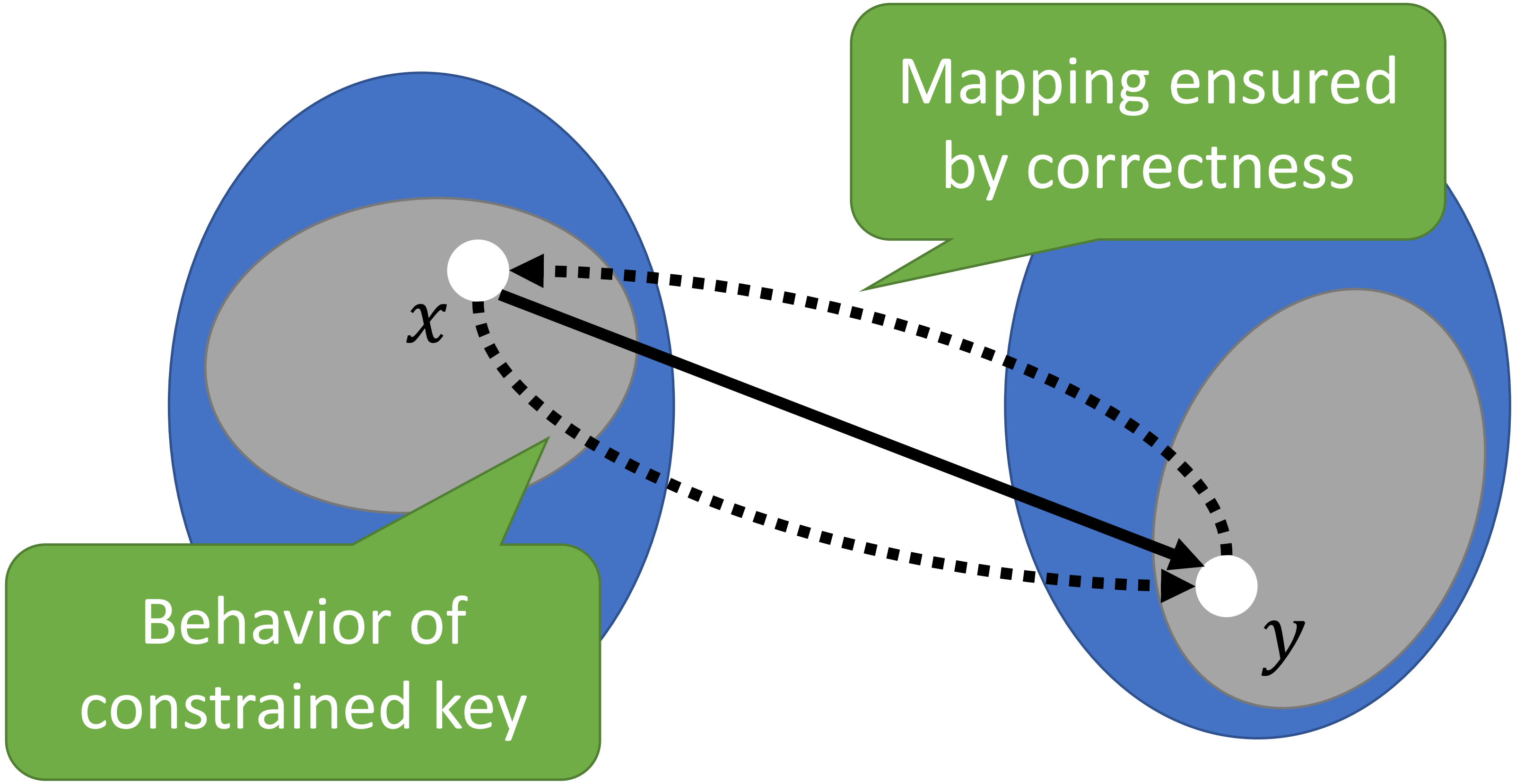




Constrained PRP Lower Bound

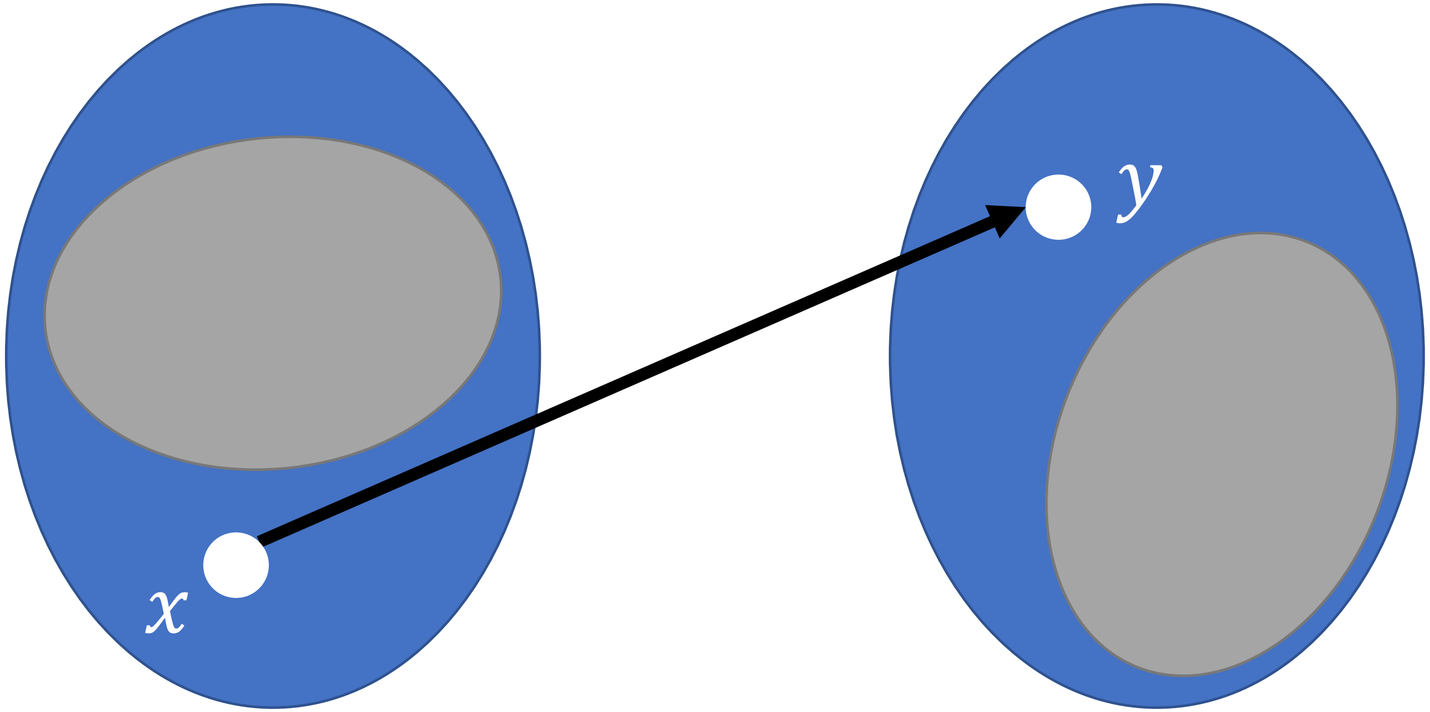
Consider what happens when constrained key is used to invert

If y is the image of an allowable point, then $F\left(k_C, F^{-1}(k_C, y)\right) = y$



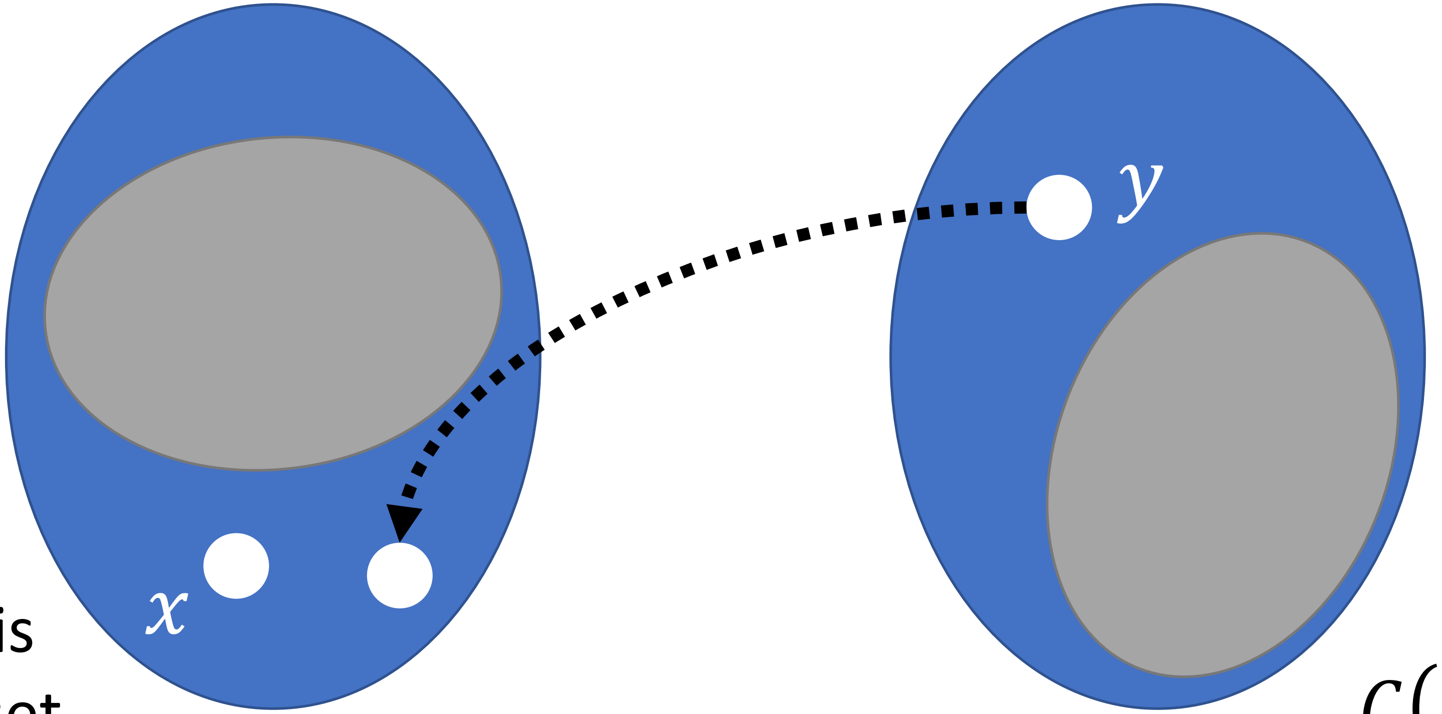
Constrained PRP Lower Bound

Consider what happens when constrained key is used to invert



Constrained PRP Lower Bound

Consider what happens when constrained key is used to invert



Case 1: preimage is outside allowable set

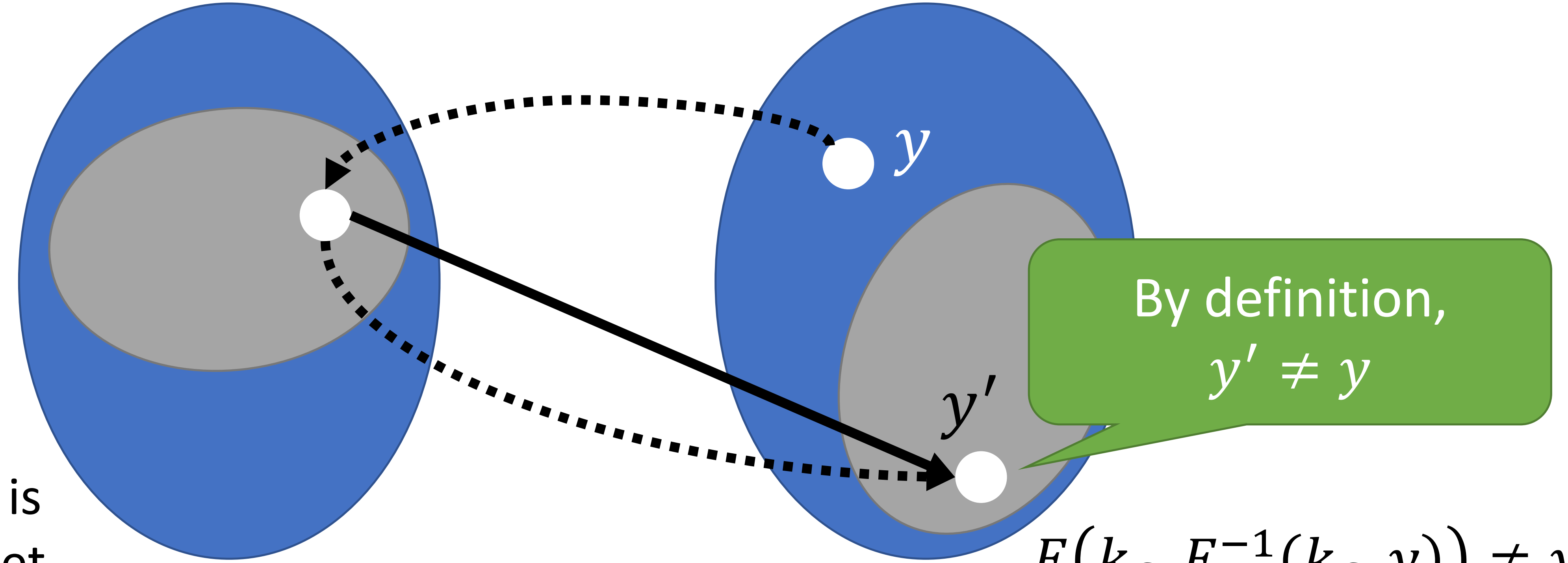
$C\left(F^{-1}(k_C, y)\right) = 0$

Constrained PRP Lower Bound

Consider what happens when constrained key is used to invert

If y is the image of a disallowed point, then either

$$C\left(F^{-1}(k_C, y)\right) = 0 \text{ or } F\left(k_C, F^{-1}(k_C, y)\right) \neq y$$

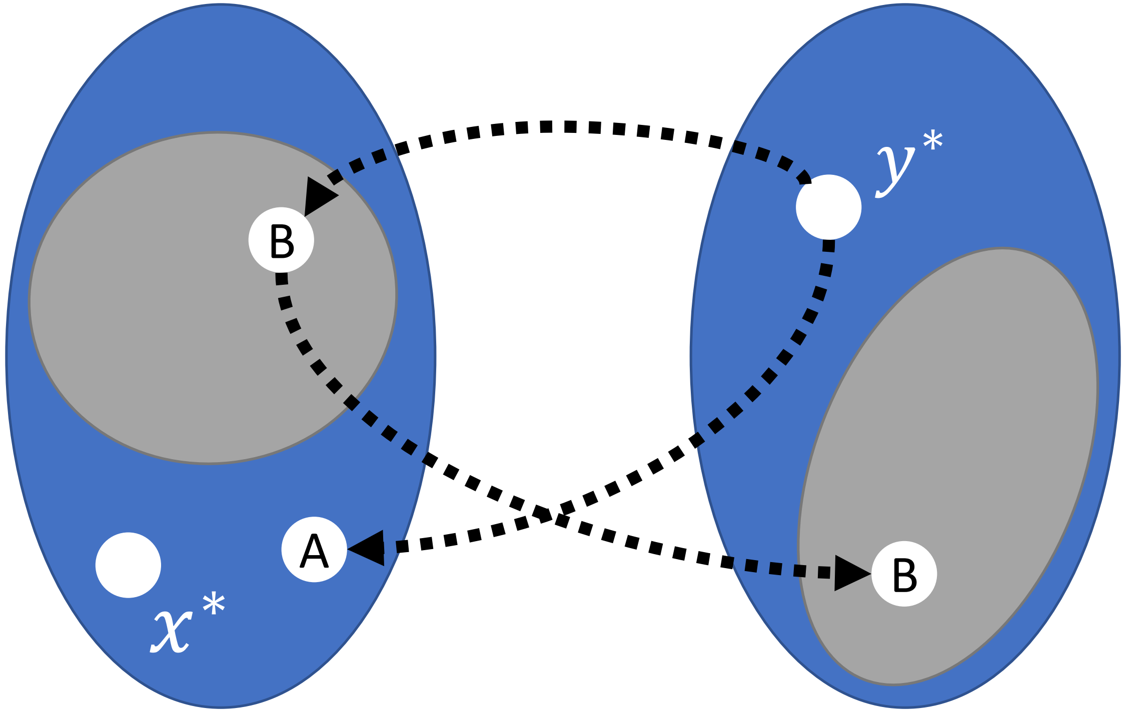


Case 2: preimage is inside allowable set

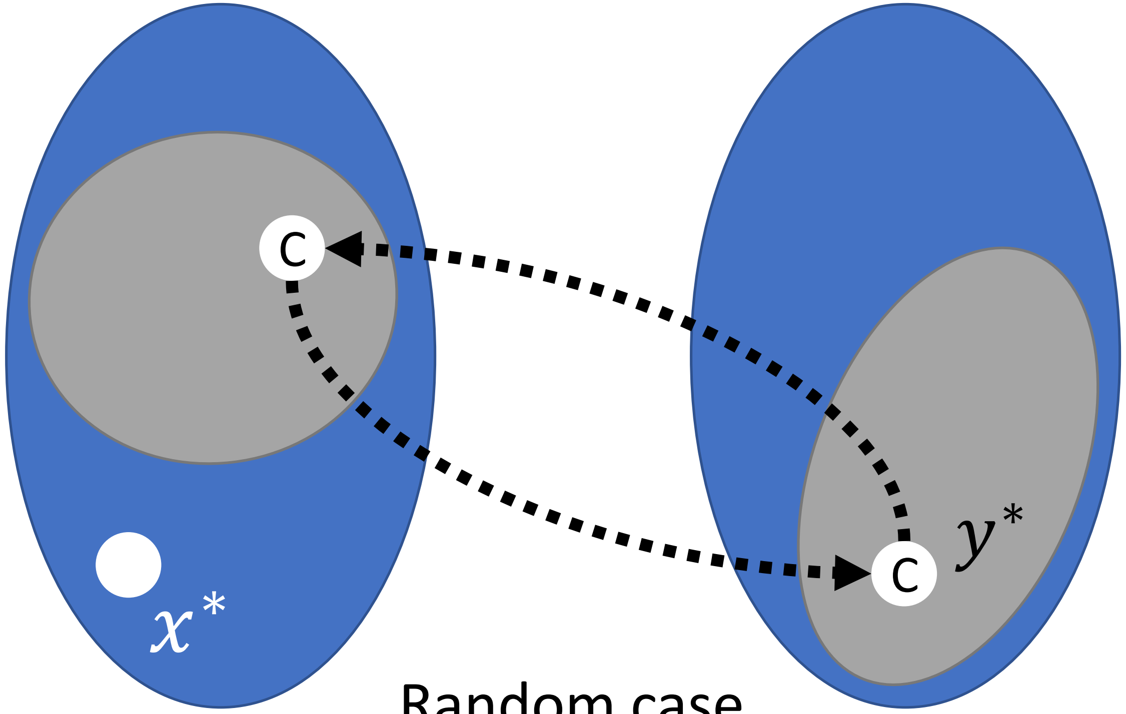
$F\left(k_C, F^{-1}(k_C, y)\right) \neq y$

Constrained PRP Lower Bound

Theorem (Informal). Any constrained PRP that allows issuing a constrained key that can evaluate on a non-negligible fraction of the domain is insecure.



Pseudorandom case



Random case
(with noticeable probability)

Relaxing the Notion

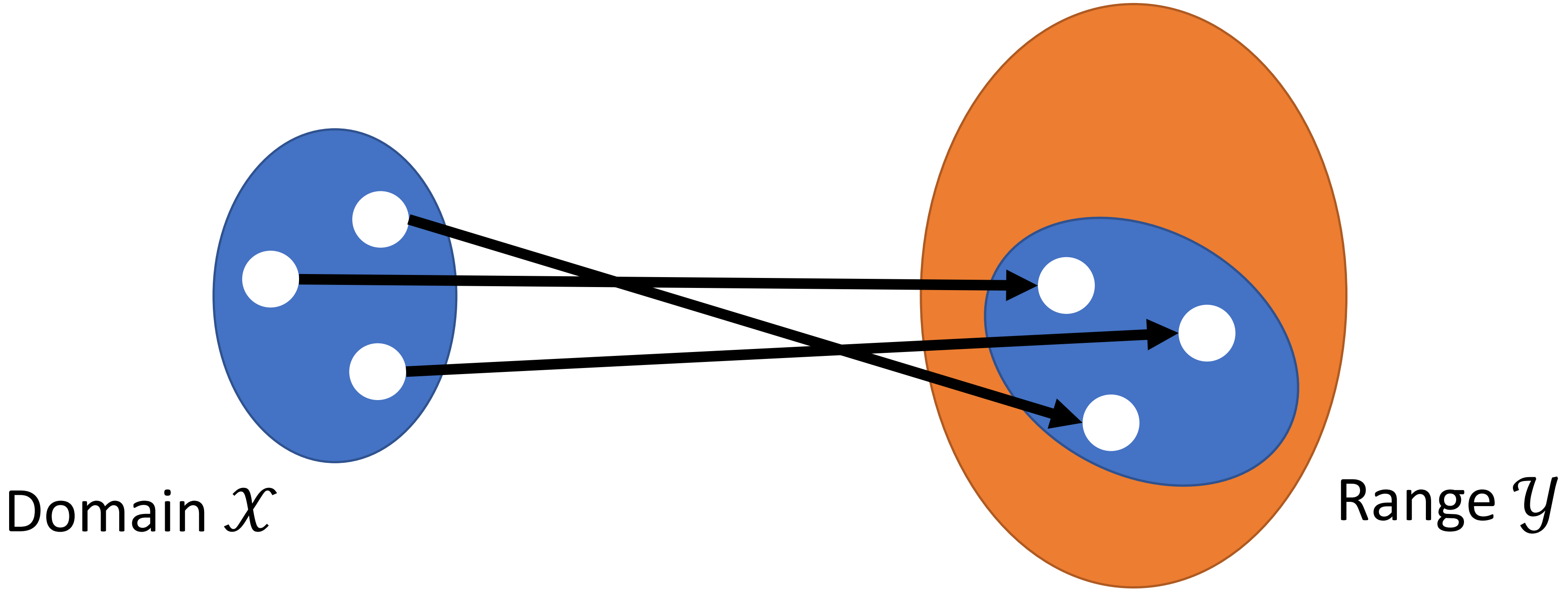
Theorem (Informal). Any constrained PRP that allows issuing a constrained key that can evaluate on a non-negligible fraction of the domain is insecure.

Punctured PRPs do not exist.

Open Question: Do prefix-constrained PRPs (where prefix is $\omega(\log \lambda)$ bits) exist?

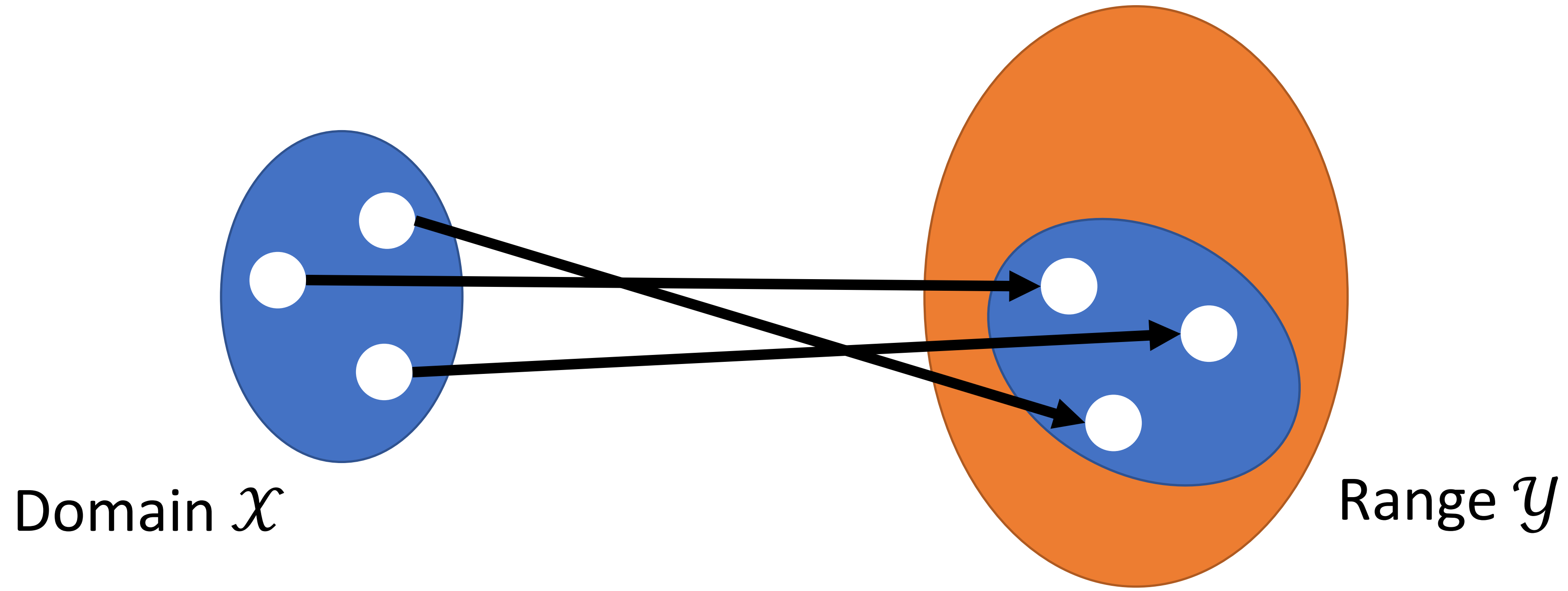
Relaxing the Notion

Theorem (Informal). Any constrained PRP that allows issuing a constrained key that can evaluate on a <u>non-negligible fraction</u> of the domain is insecure.



Relaxation: Allow range to be *much larger* than the domain

Invertible Pseudorandom Functions (IPFs)



An IPF $F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ satisfies the following properties:

- $F(k,\cdot)$ is injective for all $k \in \mathcal{K}$
- There exists an efficiently computable inverse $F^{-1}: \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\bot\}$
- $F^{-1}(k, F(k,x)) = x$ for all $x \in \mathcal{X}$
- $F^{-1}(k,y) = \bot$ for all y not in the range of $F(k,\cdot)$

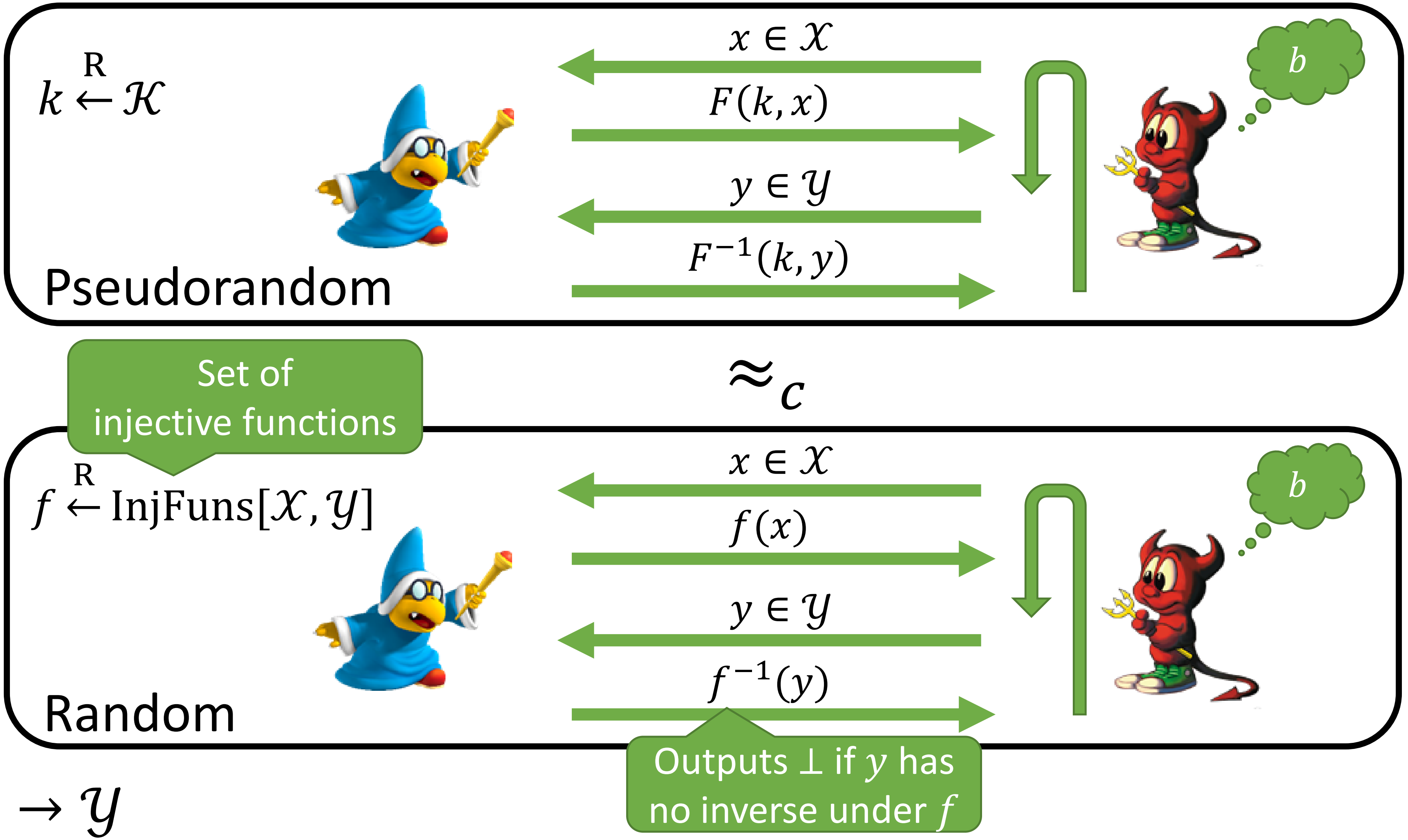
Invertible Pseudorandom Functions (IPFs)

IPFs are closely related to the notion of deterministic authenticated encryption (DAE) [RS06]. IPFs can be used to build DAE, so our constrained IPF constructions imply constrained DAE.

An IPF $F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ satisfies the following properties:

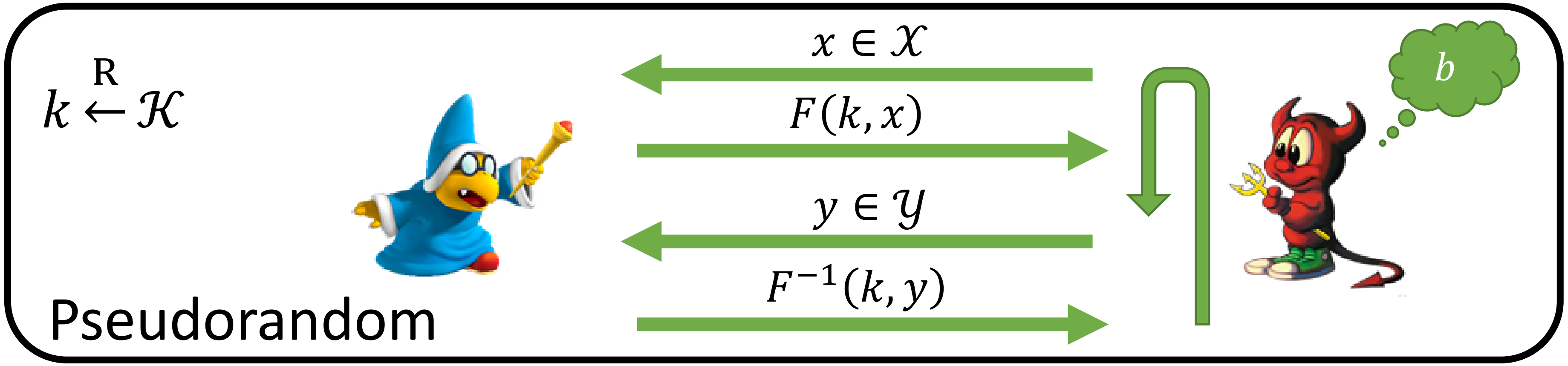
- $F(k, \cdot)$ is injective for all $k \in \mathcal{K}$
- There exists an efficiently computable inverse $F^{-1}: \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$
- $F^{-1}\big(k, F(k, x)\big) = x$ for all $x \in \mathcal{X}$
- $F^{-1}(k, y) = \perp$ for all y not in the range of $F(k, \cdot)$

Invertible Pseudorandom Functions (IPFs)

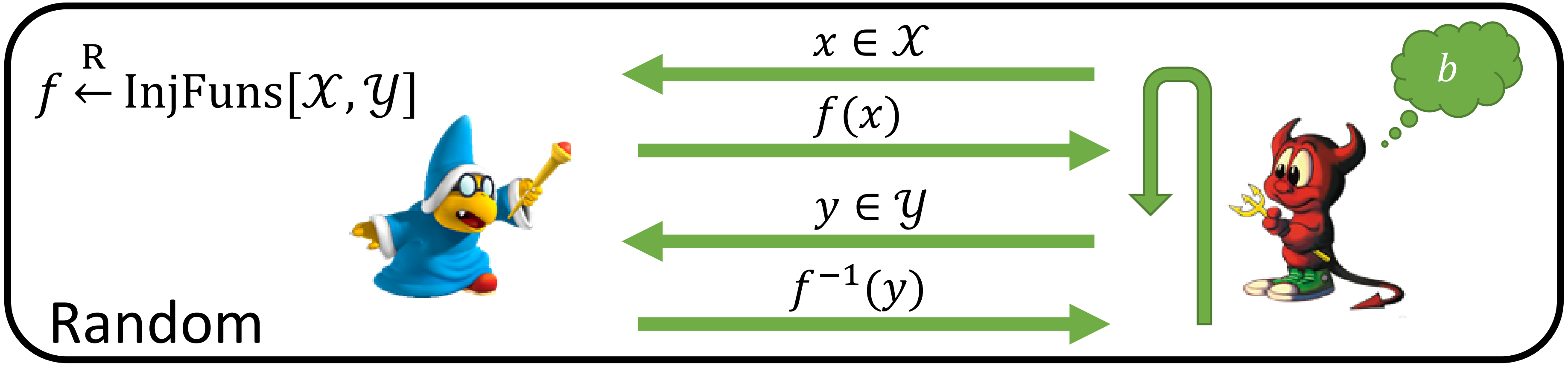


$F: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$

Invertible Pseudorandom Functions (IPFs)



$\approx_c$



When $\mathcal{X} = \mathcal{Y}$, security definition is equivalent to that for a strong PRP

Constrained IPFs

Direct generalization of constrained PRFs

IPF key

Constrained key

Can be used to evaluate at all points $x \in \mathcal{X}$ where $C(x) = 1$ and invert at all points y whenever $y = F(k, x)$ for some x where $C(x) = 1$

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

Constrained IPF Security

$k \xleftarrow{R} \mathcal{K}$

$f \xleftarrow{R} \text{InjFuns}[\mathcal{X}, \mathcal{Y}]$

Challenger

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

Constrain queries

C

$k_C = \text{Constrain}_C(k)$

Evaluation queries

x

$F(k, x)$

Inversion queries

y

$F^{-1}(k, y)$

Challenge queries

x^*

y^*

Adversary

Random: $y^* = f(x^*)$

Pseudorandom: $y^* = F(k, x^*)$

Constrained IPF Security

Admissibility conditions:
- $C(x^*) = 0$
- No evaluation queries on x^*
- No inversion queries on y^*

C ←

$k_C = \text{Constrain}_C(k)$ →

x ←

$F(k, x)$ →

y ←

$F^{-1}(k, y)$ →

x^* ←

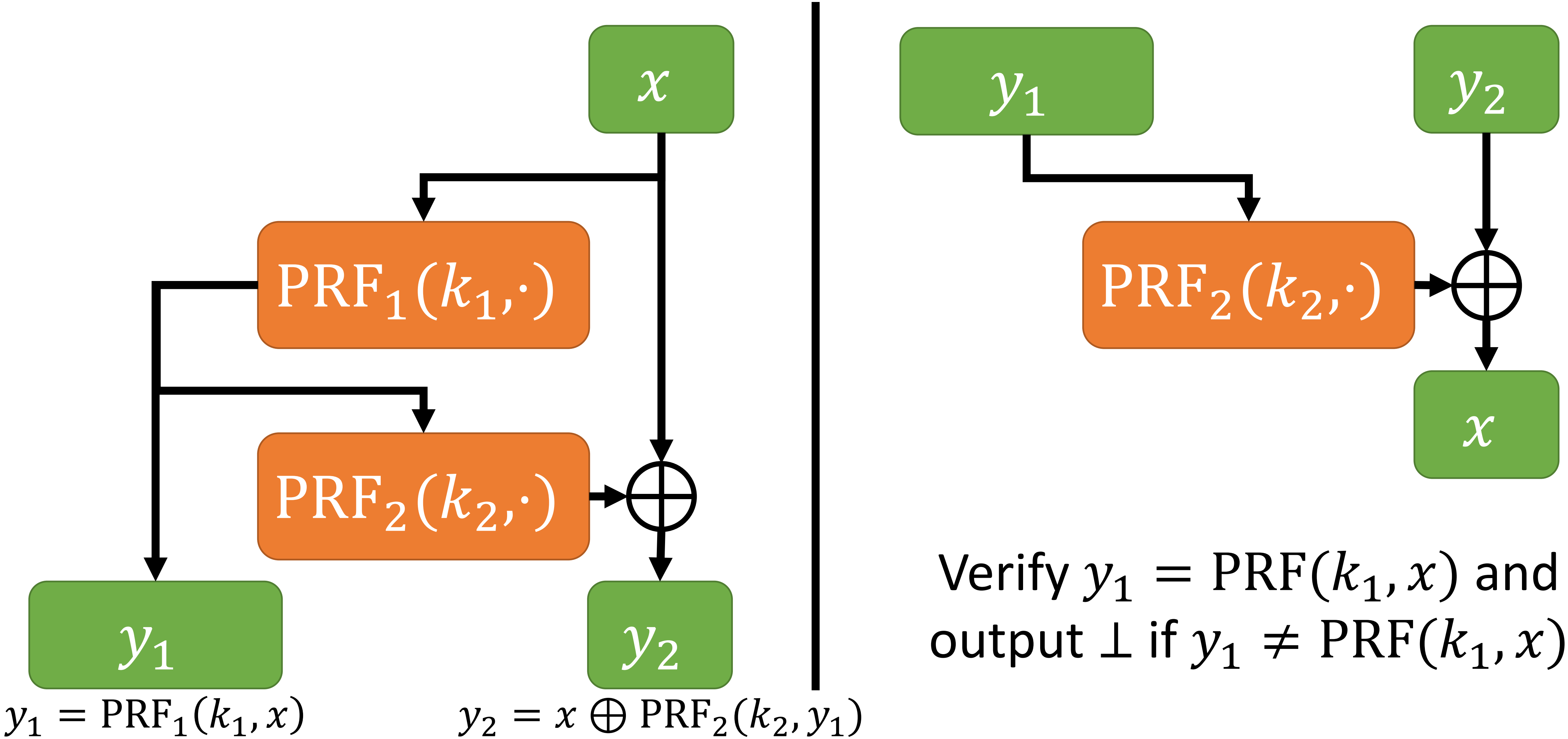
y^* →

Adversary

Random: $y^* = f(x^*)$

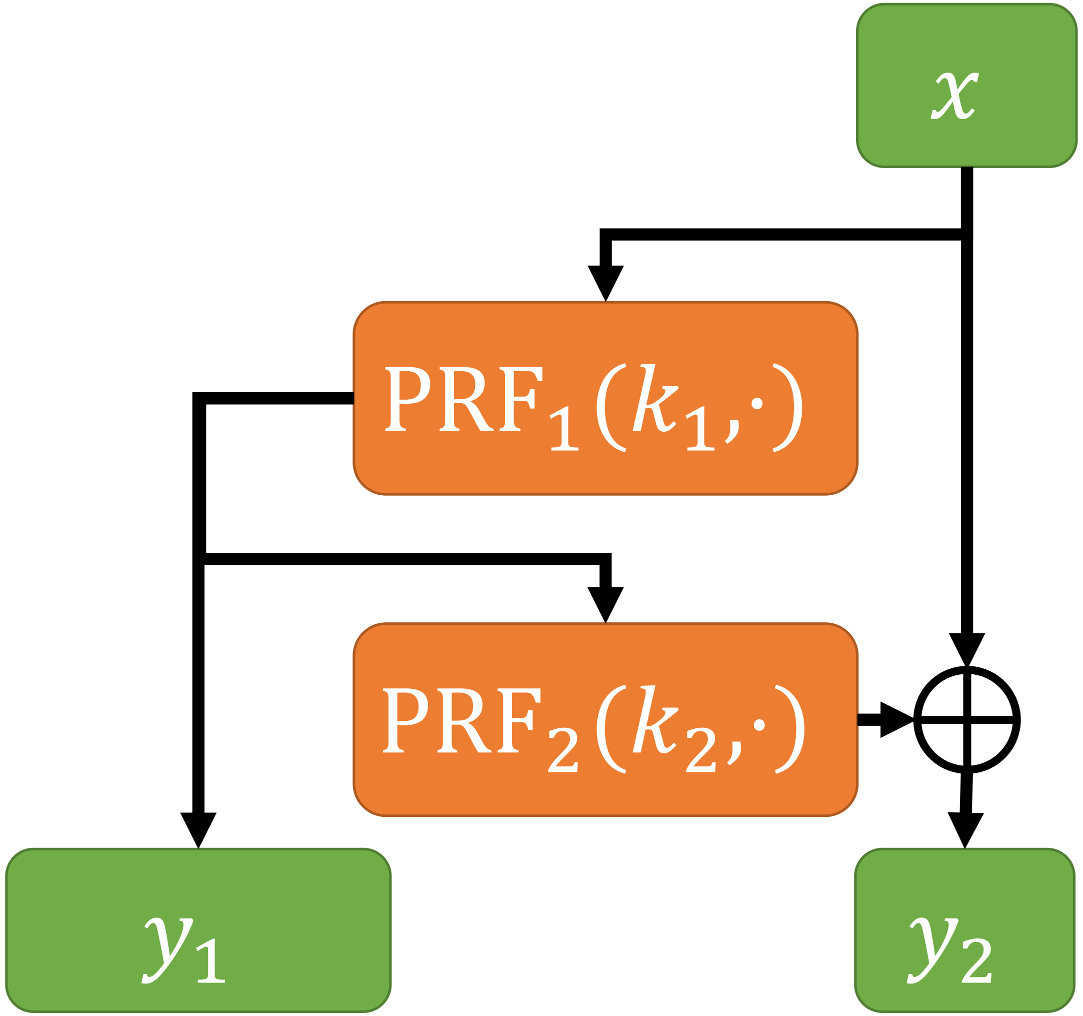
Pseudorandom: $y^* = F(k, x^*)$

A Puncturable IPF



Verify $y_1 = \mathrm{PRF}(k_1, x)$ and output $\perp$ if $y_1 \neq \mathrm{PRF}(k_1, x)$

A Puncturable IPF



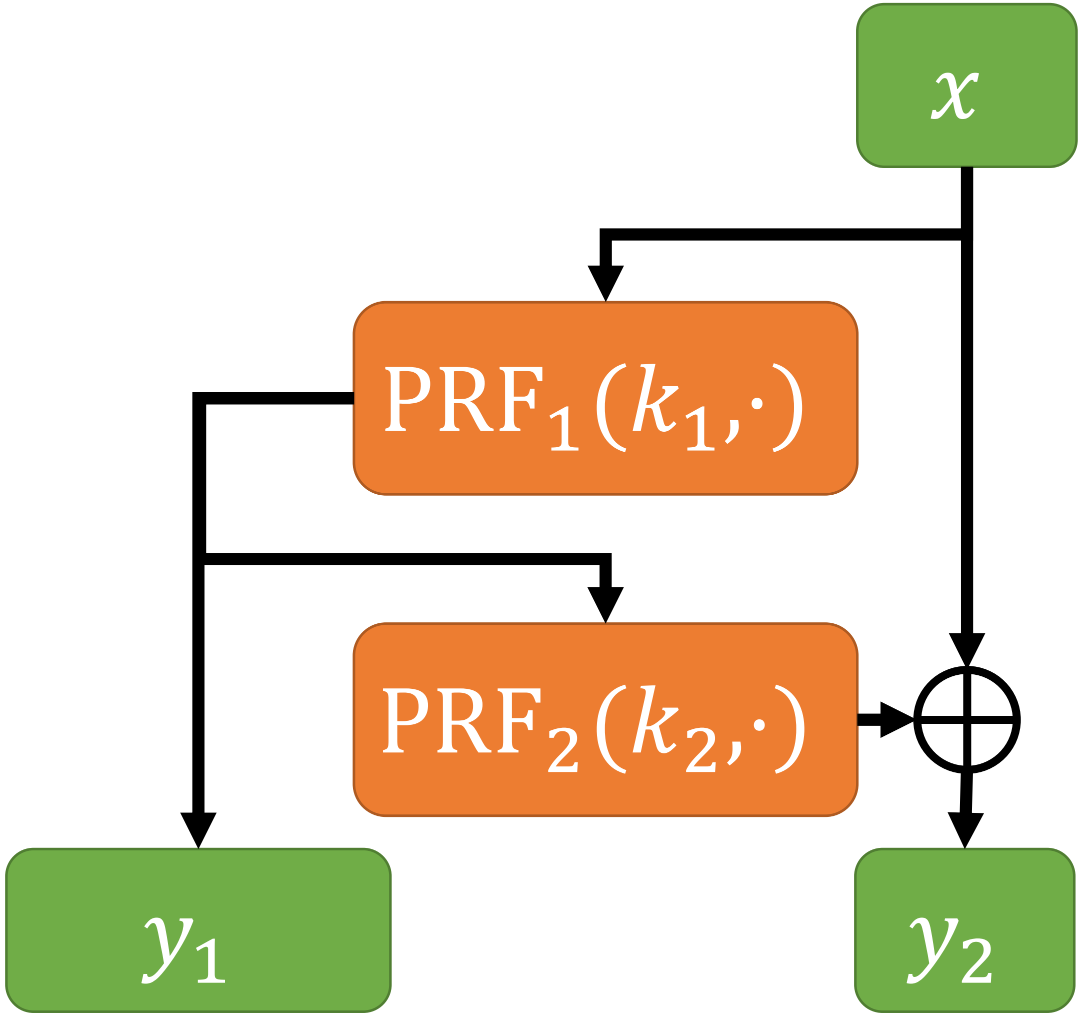
$$y_1 = \mathrm{PRF}_1(k_1, x)$$

$$y_2 = x \oplus \mathrm{PRF}_2(k_2, y_1)$$

Equivalent to DAE construction called synthetic IV (SIV) [RS06]

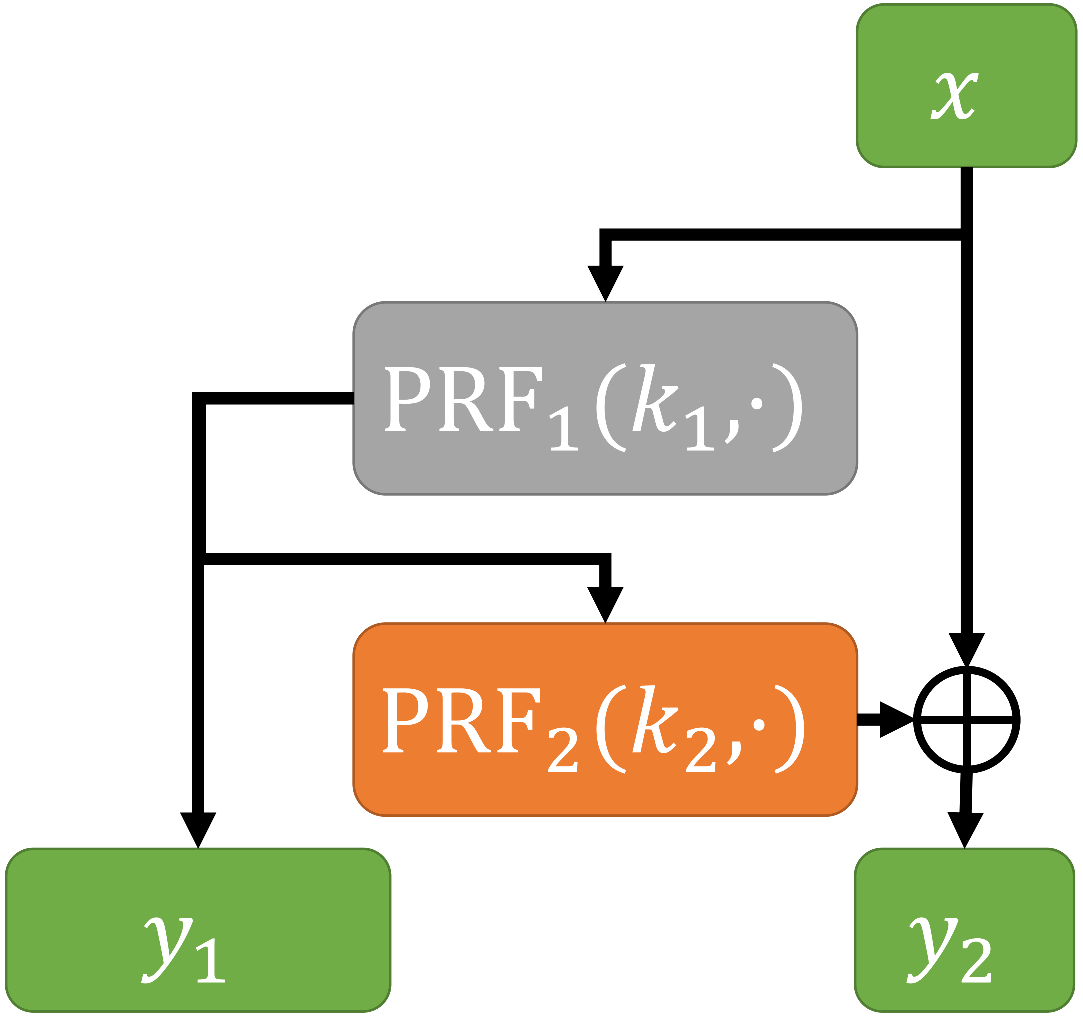
Can also be viewed as an unbalanced Feistel network (with one block set to all 0s)

A Puncturable IPF



How to puncture this construction?

A Puncturable IPF

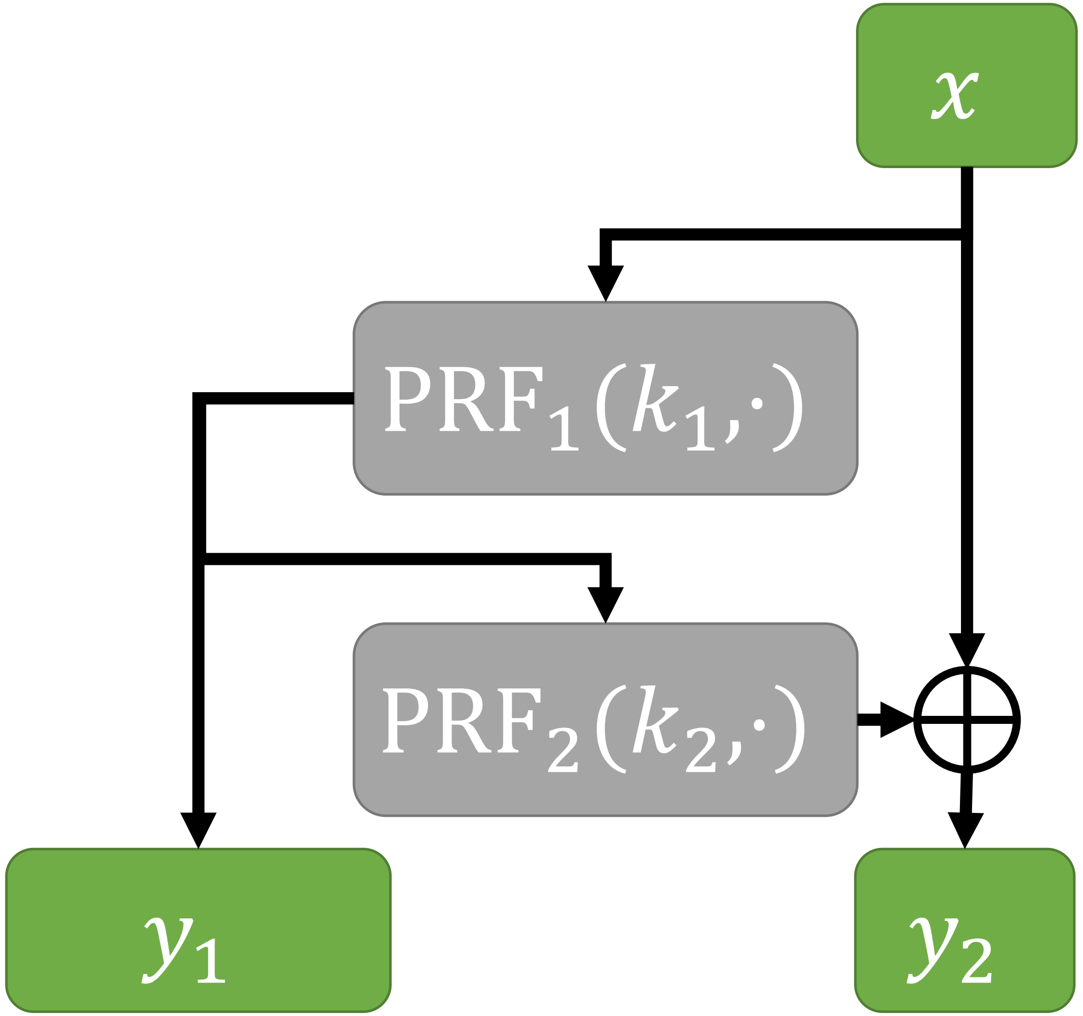


How to puncture this construction?

First attempt: only puncture k_1 at x^*

Given challenge (y_1^*, y_2^*), can test whether $y_2^* \oplus \mathrm{PRF}_2(k_2, y_1^*) = x^*$

A Puncturable IPF



How to puncture this construction?

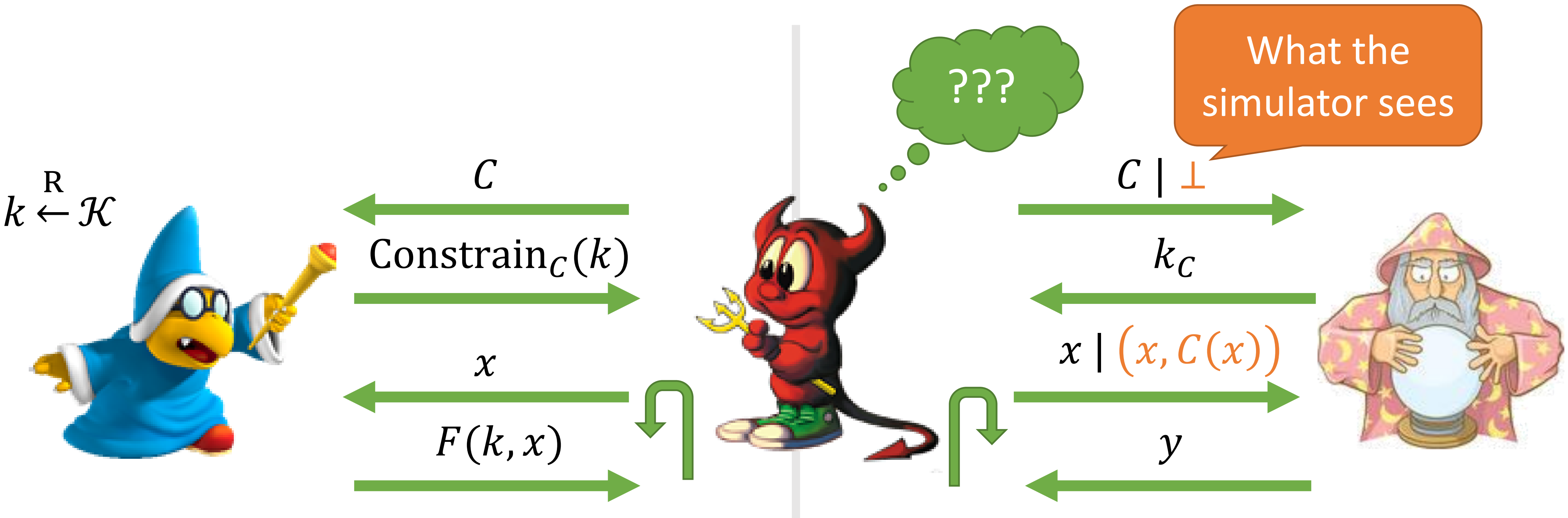
First attempt: only puncture k_1 at x^*

Given challenge (y_1^*, y_2^*), can test whether $y_2^* \oplus \mathrm{PRF}_2(k_2, y_1^*) = x^*$

Second attempt: also puncture k_2 at $y_1^* = \mathrm{PRF}_1(k_1, x^*)$

Punctured key reveals punctured point!

Private Constrained PRFs [BLW17]

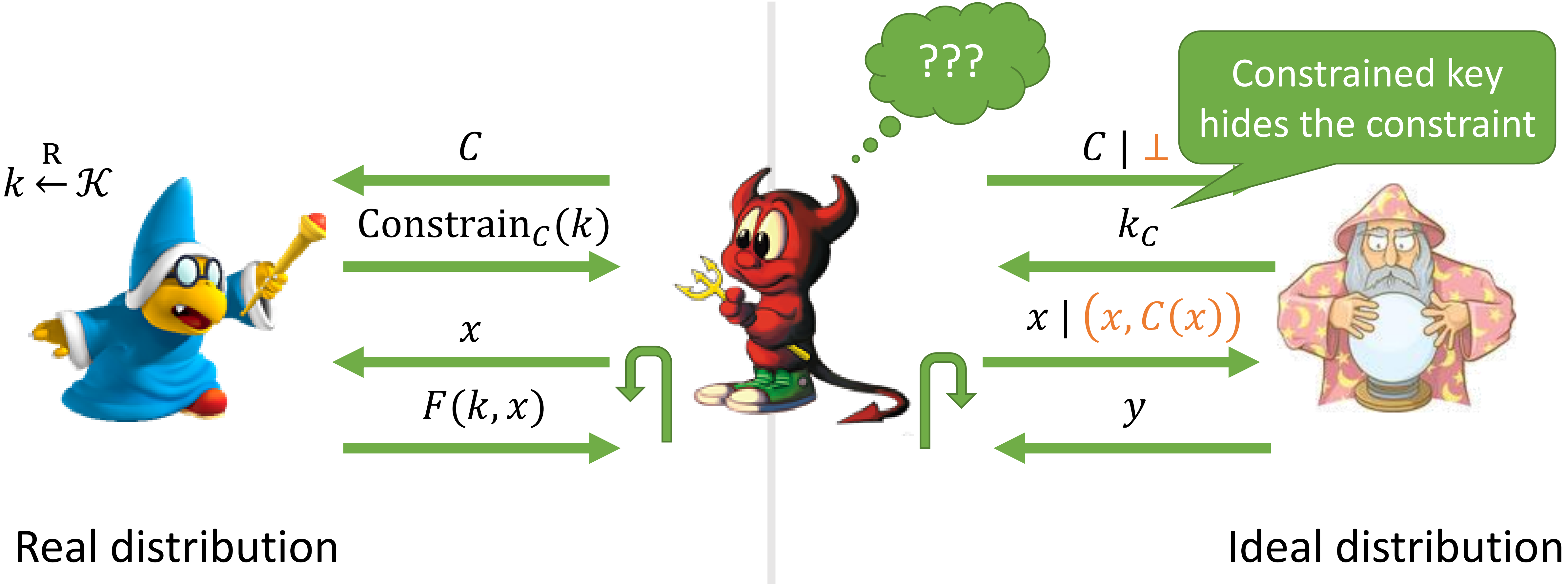


Real distribution

Ideal distribution

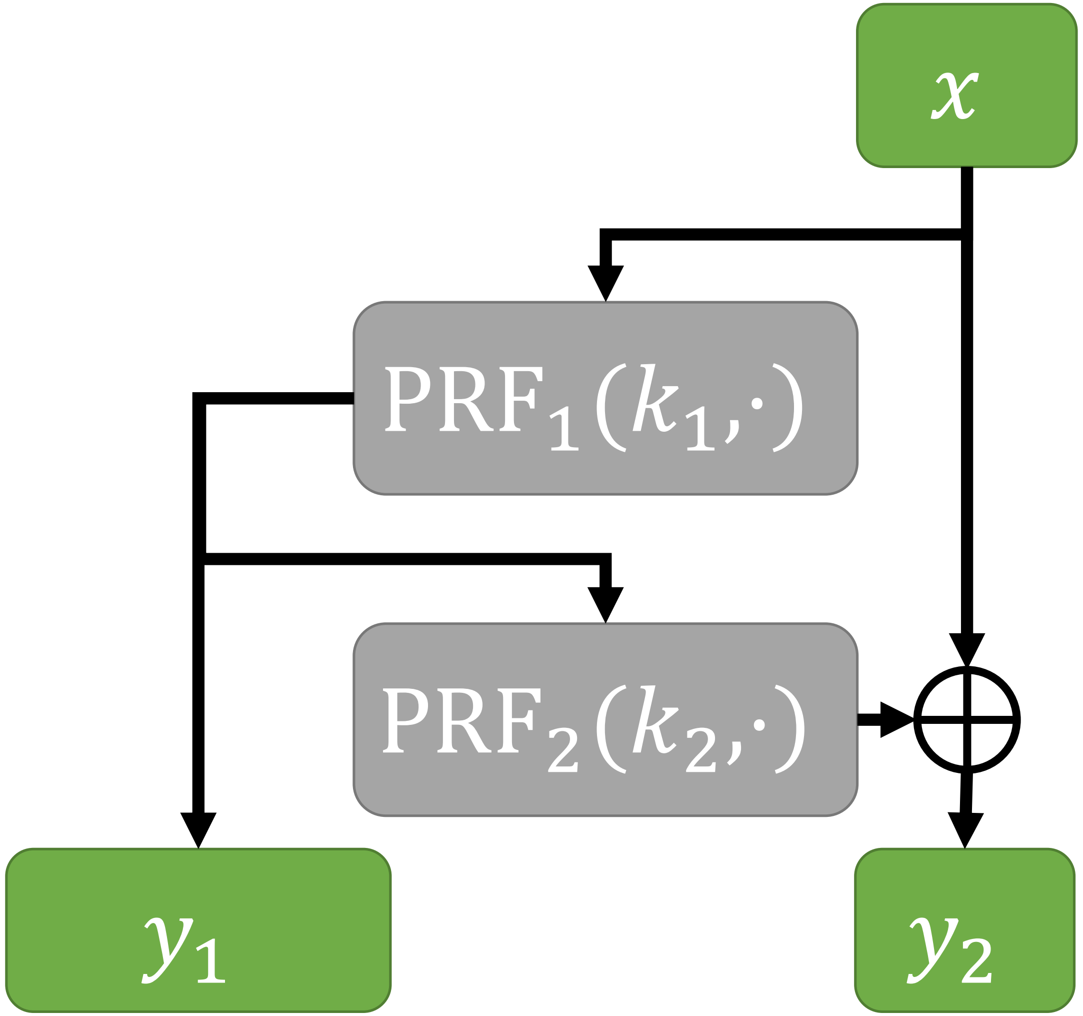
(Selective) single-key privacy, simulation-based security [BKM17, CC17]

Private Constrained PRFs [BLW17]



(Selective) single-key privacy, simulation-based security [BKM17, CC17]

A Puncturable IPF



Master key: $k = (k_1, k_2)$

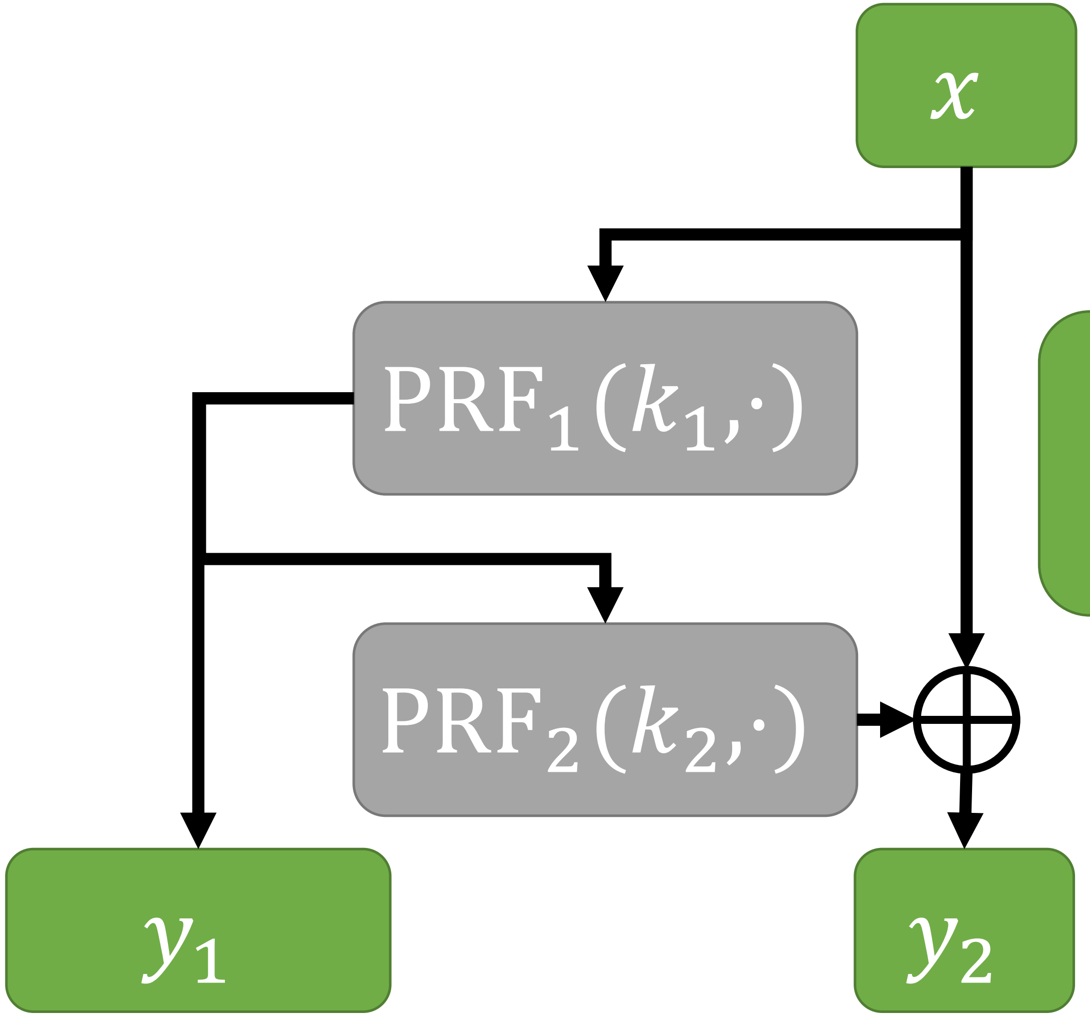
Punctured key (punctured at x^*):

- k_1 punctured at x^*
- k_2 *privately* punctured at $\mathrm{PRF}_1(k_1, x^*)$

$$y_1^* = \mathrm{PRF}_1(k_1, x^*)$$

$$y_2^* = x^* \oplus \mathrm{PRF}_2(k_2, y_1^*)$$

A Puncturable IPF



Master key: $k = (k_1, k_2)$

Punctured key (punctured at x^*):

- punctured at x^*
- punctured at

Indistinguishable from uniform by constrained security of PRF_1

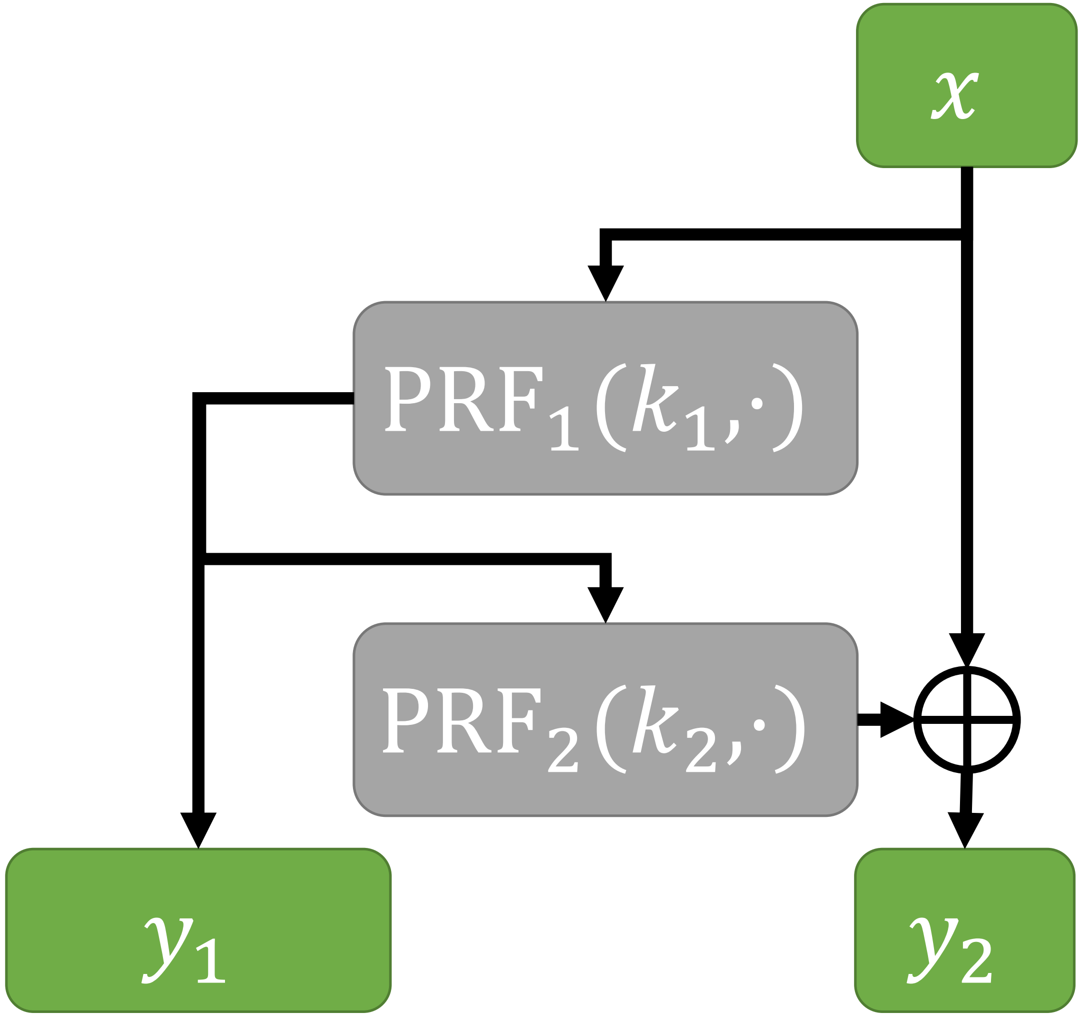
Hides y_1^*

$$y_1^* = \mathrm{PRF}_1(k_1, x^*)$$

$$y_2^* = x^* \oplus \mathrm{PRF}_2(k_2, y_1^*)$$

Indistinguishable from uniform by constrained security of PRF_2

A Puncturable IPF



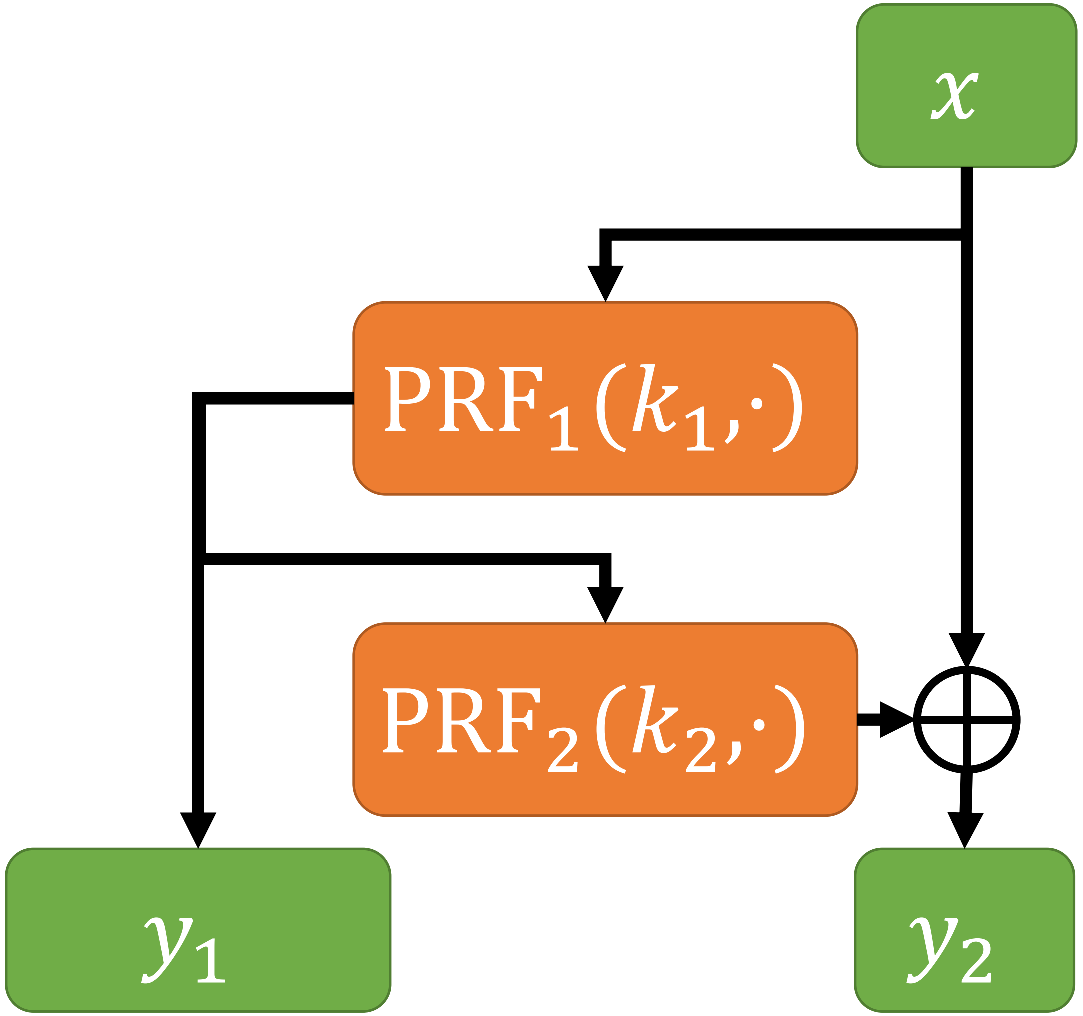
Master key: $k = (k_1, k_2)$

Punctured key (punctured at x^*):

- k_1 punctured at x^*
- k_2 *privately* punctured at $\mathrm{PRF}_1(k_1, x^*)$

Can be instantiated from standard lattice assumptions [BKM17, CC17, BTVW17]

Circuit-Constrained IPFs



Master key: $k = (k_1, k_2)$

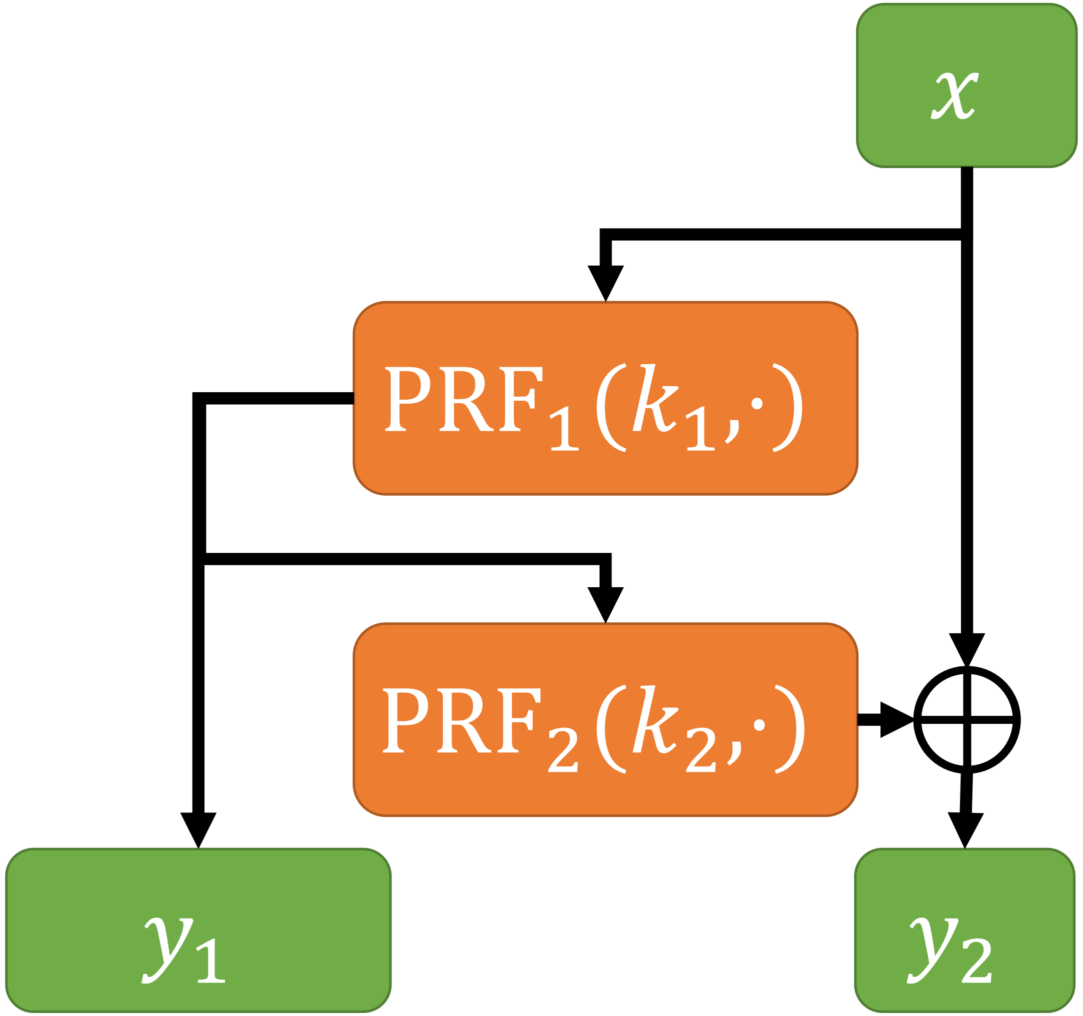
For puncturing at x^*:

- Puncture k_1 at x^*
- Puncture k_2 at $\mathrm{PRF}_1(k_1, x^*)$

To constrain to circuit C:

- Constrain k_1 to C
- **Difficulty:** Need to constrain k_2 on a *pseudorandom* set (the image of $\mathrm{PRF}_1(k_1,\cdot)$ on the points allowed by C)

Circuit-Constrained IPFs



Master key: $k = (k_1, k_2)$

For puncturing at x^*:

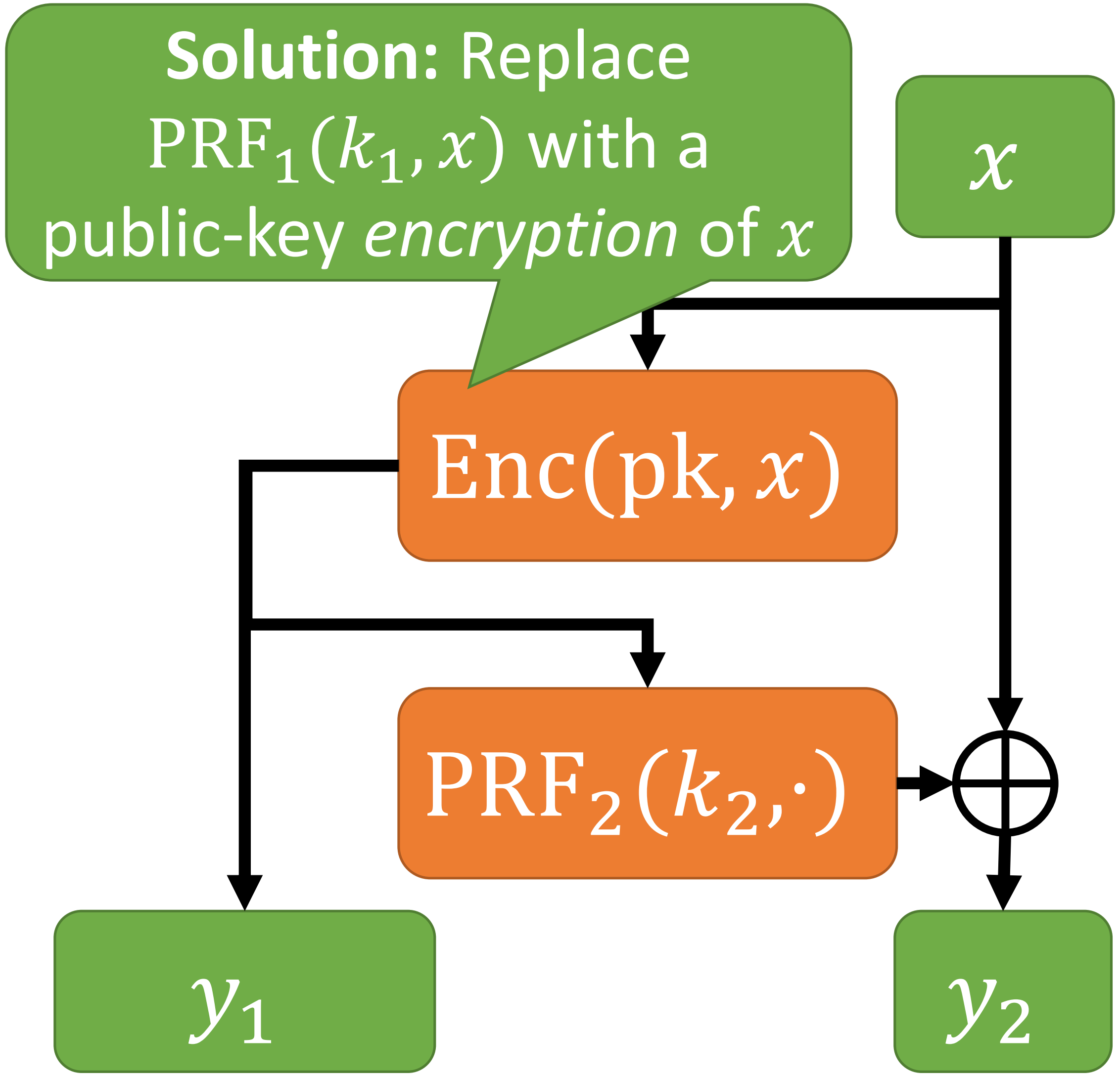
- Puncture k_1 at x^*
-

This set does not have a simple description unless PRF_1 is efficiently invertible

To co

-
- **Difficulty:** Need to constrain k_2 on a *pseudorandom* set (the image of $\mathrm{PRF}_1(k_1,\cdot)$ on the points allowed by C)

Circuit-Constrained IPFs

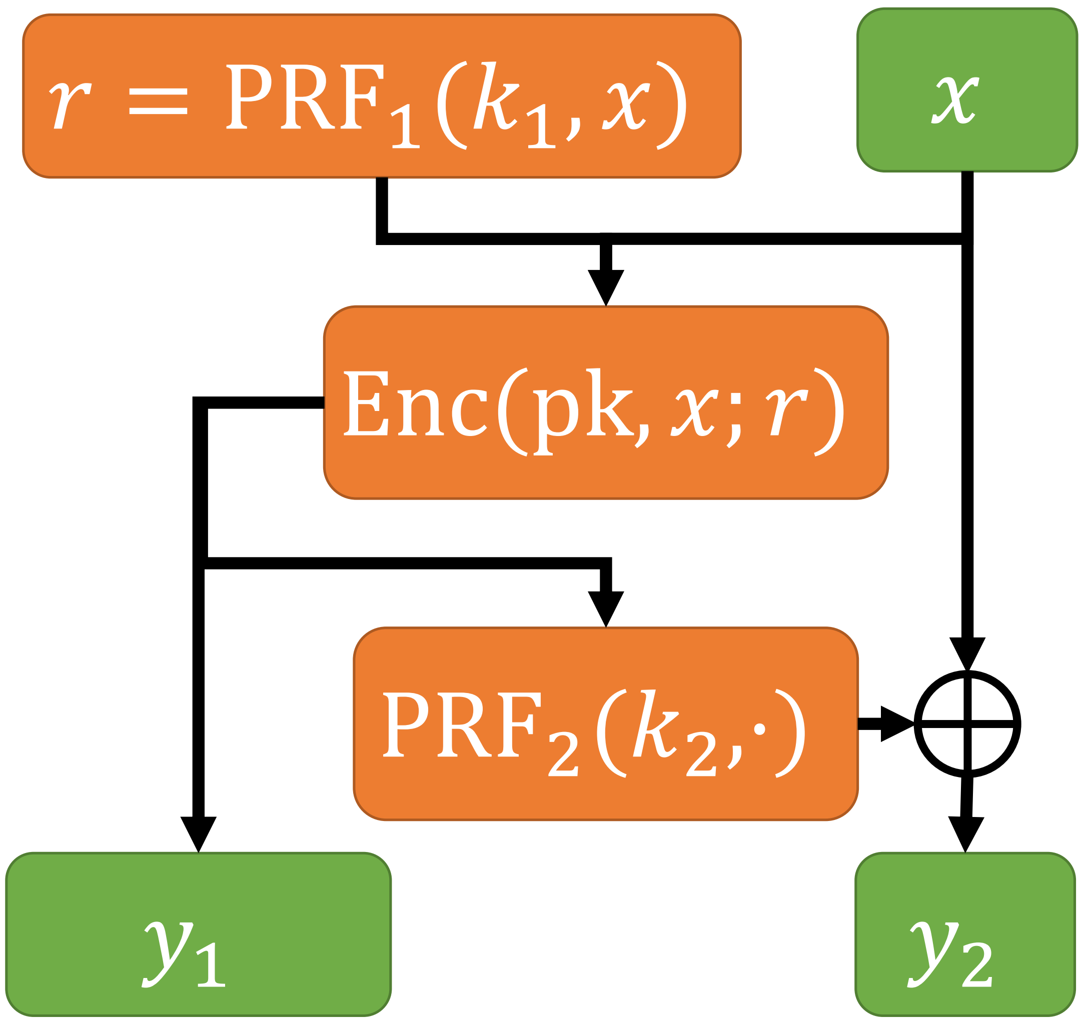


Decryption key can be used to recover x from y_1 and for checking constraint satisfiability

Two problems:

- IPFs are deterministic, but encryption is randomized
- Need a way to constrain the encryption scheme

Circuit-Constrained IPFs



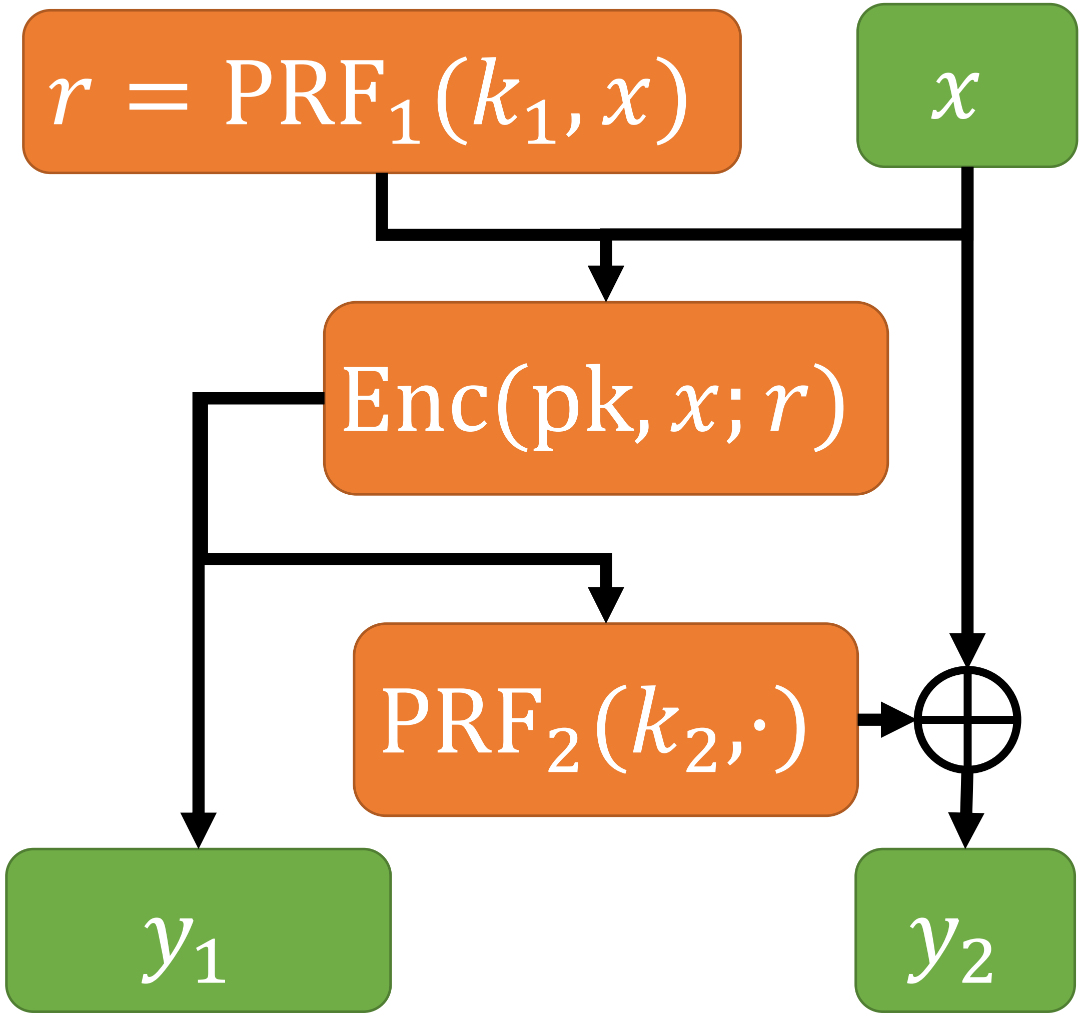
Decryption key can be used to recover x from y_1 and for checking constraint satisfiability

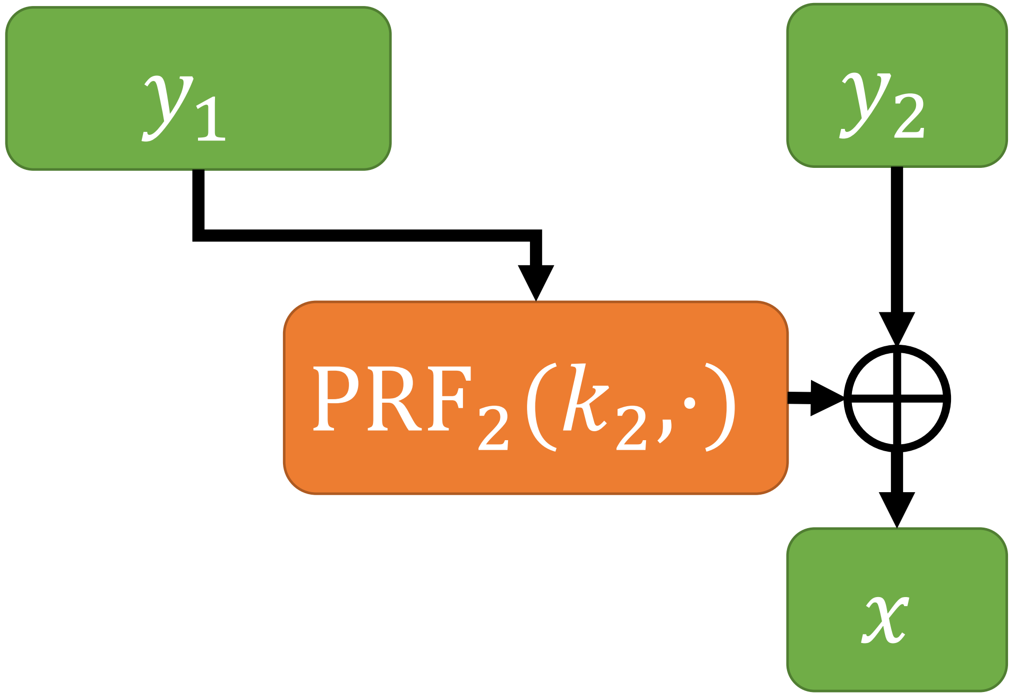
Two problems:

- IPFs are deterministic, but encryption is randomized
- Need a way to constrain the encryption scheme

Solution: derive encryption randomness from constrained PRF

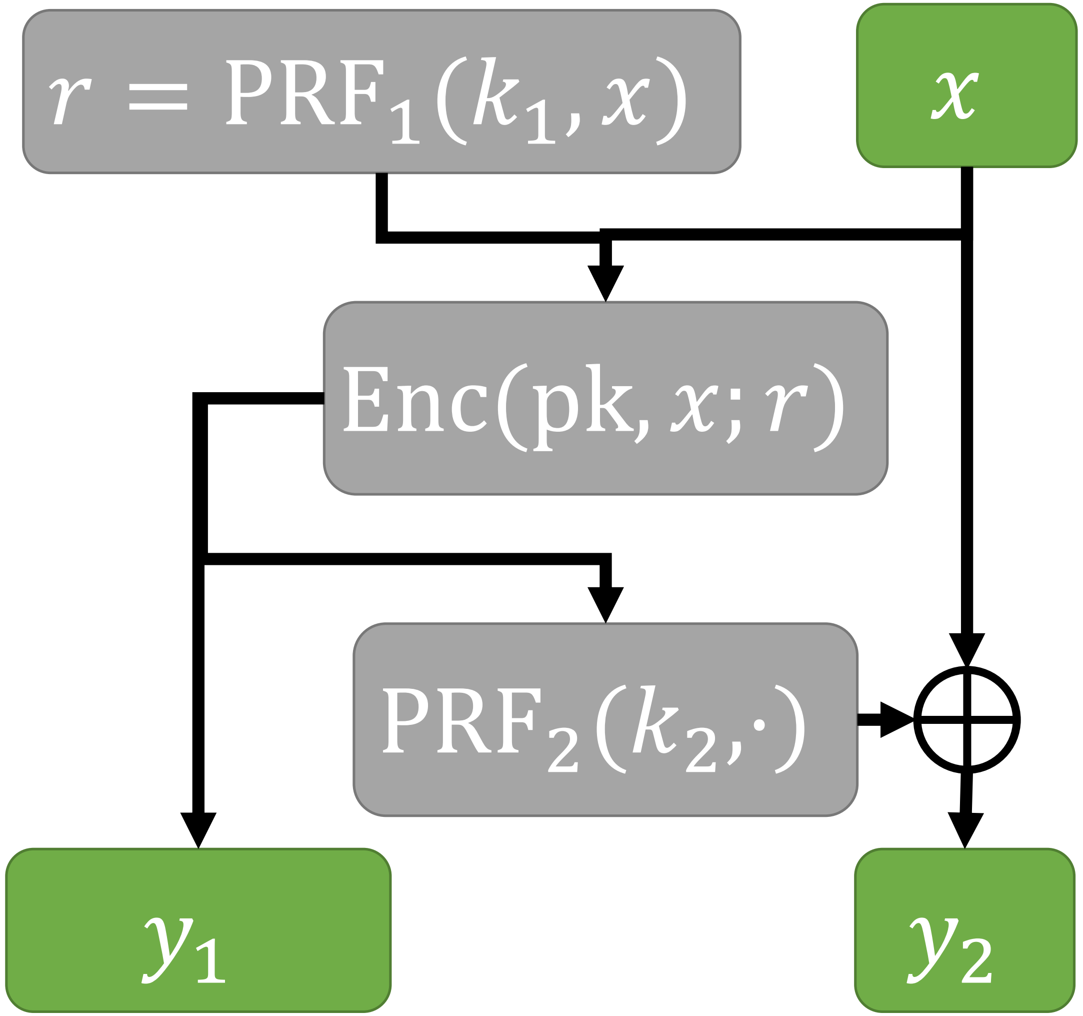
Circuit-Constrained IPFs





Verify $y_1 = \mathrm{Enc}(\mathrm{pk}, x; r)$ where $r = \mathrm{PRF}_1(k_1, x)$ and output $\perp$ if $y_1 \neq \mathrm{Enc}(\mathrm{pk}, x; r)$

Circuit-Constrained IPFs



Master key: $k = (\mathrm{pk}, \mathrm{sk}, k_1, k_2)$

Constrained key for a circuit C:

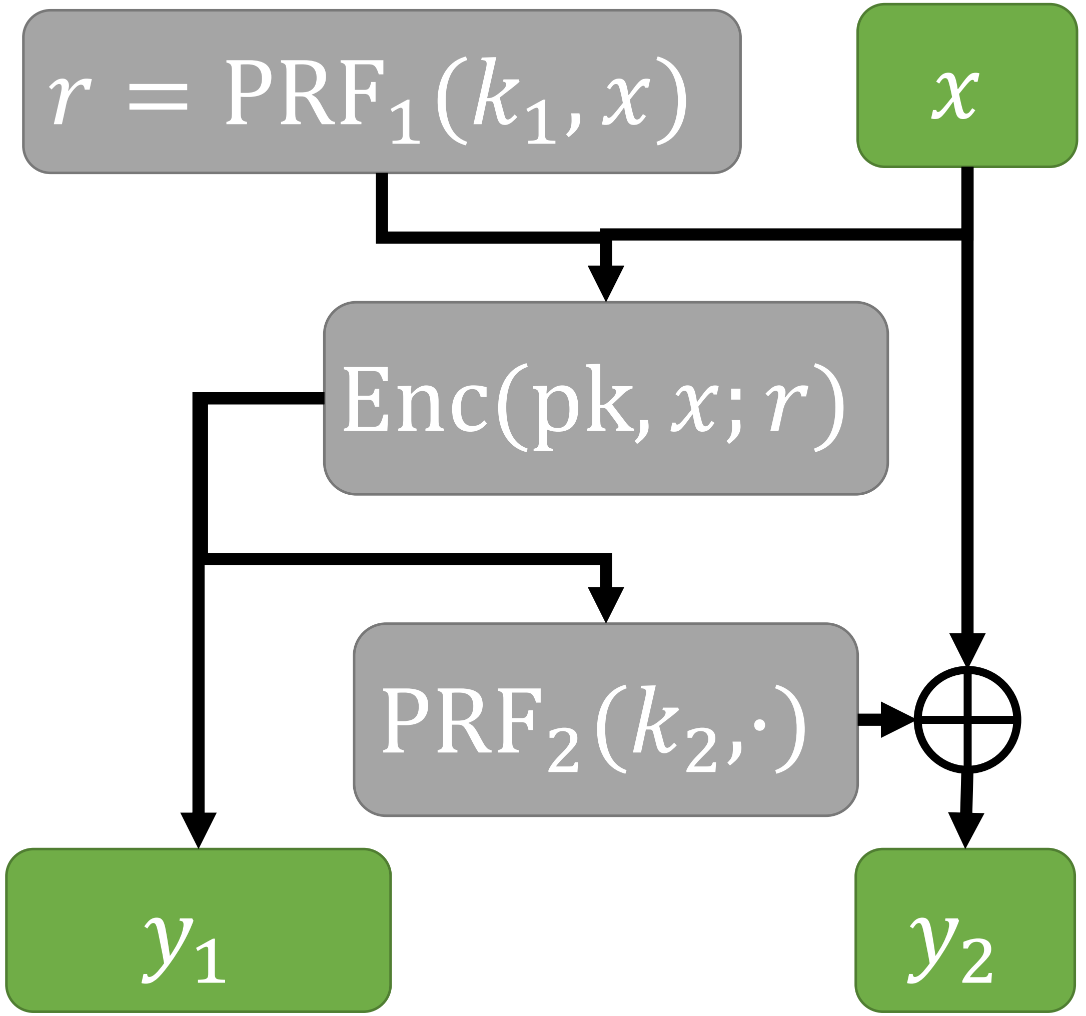
- public key pk
- k_1 constrained to C
- k_2 *privately* constrained to following circuit:

Hard-wired: sk and C

On input ct:

- Let $x \leftarrow \mathrm{Dec}(\mathrm{sk}, \mathrm{ct})$
- Output 1 if $x \neq \perp$ and $C(x) = 1$
- Output 0 otherwise

Circuit-Constrained IPFs



Master key: $k = (\mathrm{pk}, \mathrm{sk}, k_1, k_2)$

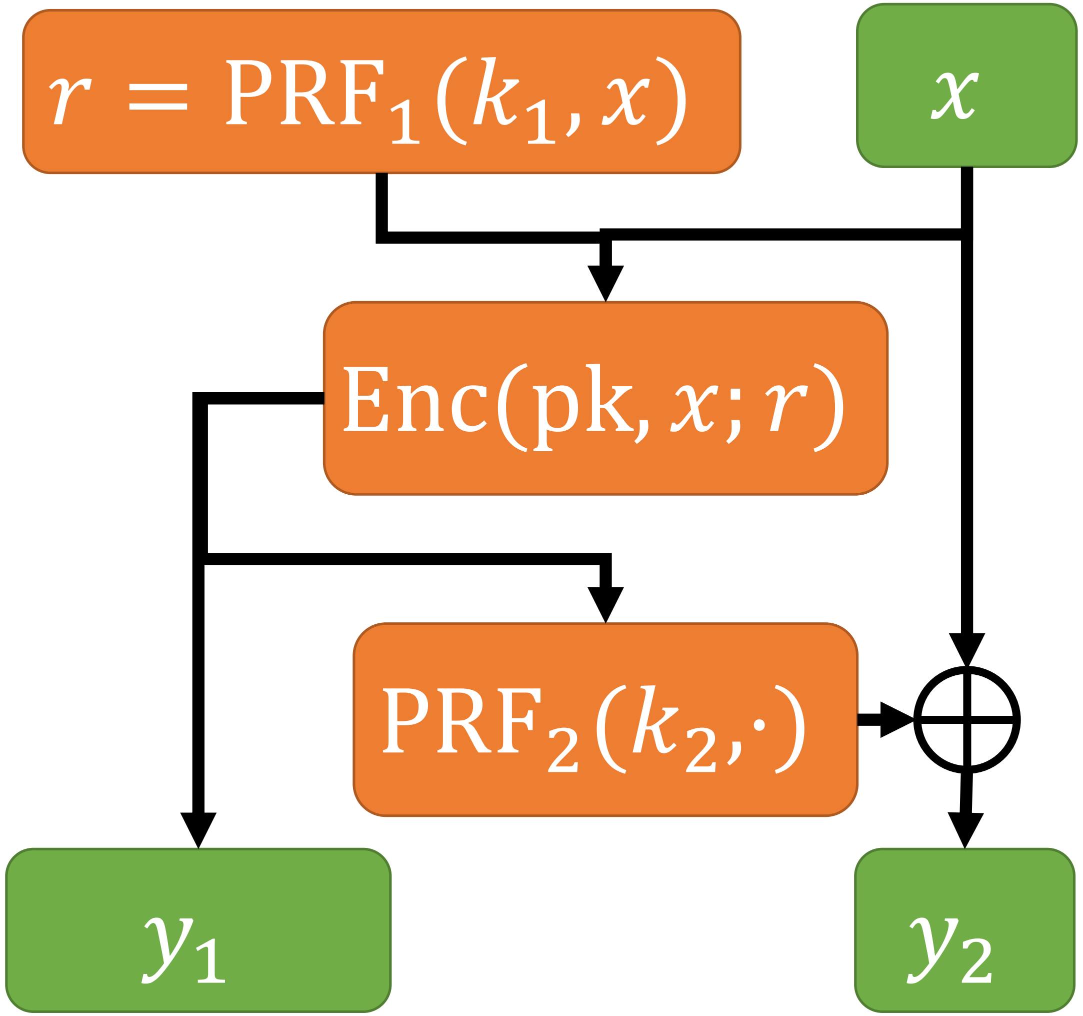
Privacy is essential to hide the secret key (the inversion trapdoor)

Hard-wired: sk and C

On input ct:

- Let $x \leftarrow \mathrm{Dec}(\mathrm{sk}, \mathrm{ct})$
- Output 1 if $x \neq \perp$ and $C(x) = 1$
- Output 0 otherwise

Circuit-Constrained IPFs



Construction is a (single-key) secure circuit-constrained IPF if

- PRF_1 is a circuit-constrained PRF
- PRF_2 is a private circuit-constrained PRF
- $(\mathrm{Enc}, \mathrm{Dec})$ is a CCA-secure public-key encryption scheme

All primitives can be instantiated from standard lattice assumptions

[See paper for security analysis]

Conclusions

Can we constrain other cryptographic primitives, such as pseudorandom permutations (PRPs)?

- Constrained PRPs for many natural classes of constraints *do not exist*
- Circuit-constrained *invertible pseudorandom functions* (IPFs) where the range is superpolynomially larger than the domain can be constructed from lattices

Open Problems

Can we construct constrained **PRPs** for sufficiently restricting constraint classes (e.g., prefix-constrained PRPs)?

Can we construct a multi-key circuit-constrained IPF from standard assumptions?

Thank you!

https://eprint.iacr.org/2017/477