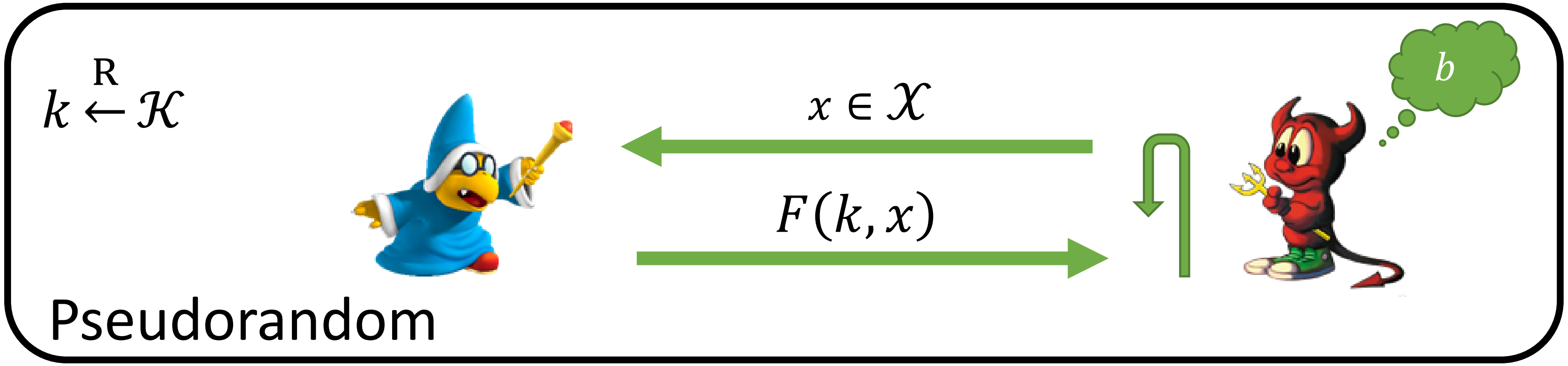
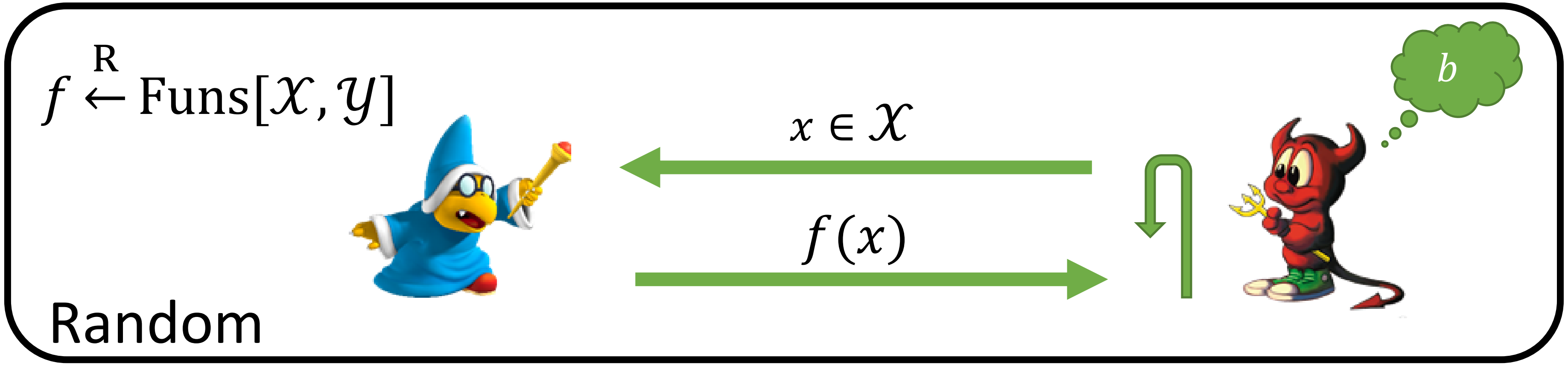
Constrained Keys for Invertible Pseudorandom Functions

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

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

Constrained PRFs [BW13, BGI13, KPTZ13]

Constrained PRF: PRF with additional "constrain" functionality

PRF key

→ Constrain_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]

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

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]

Constrained PRFs [BW13, BGI13, KPTZ13]

Many applications:

- Punctured programming paradigm [SW14]
- Identity-based key exchange, broadcast encryption [BW13]

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]
- (Single-key) circuit-constrained PRFs from LWE [BV15]

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

Our Results

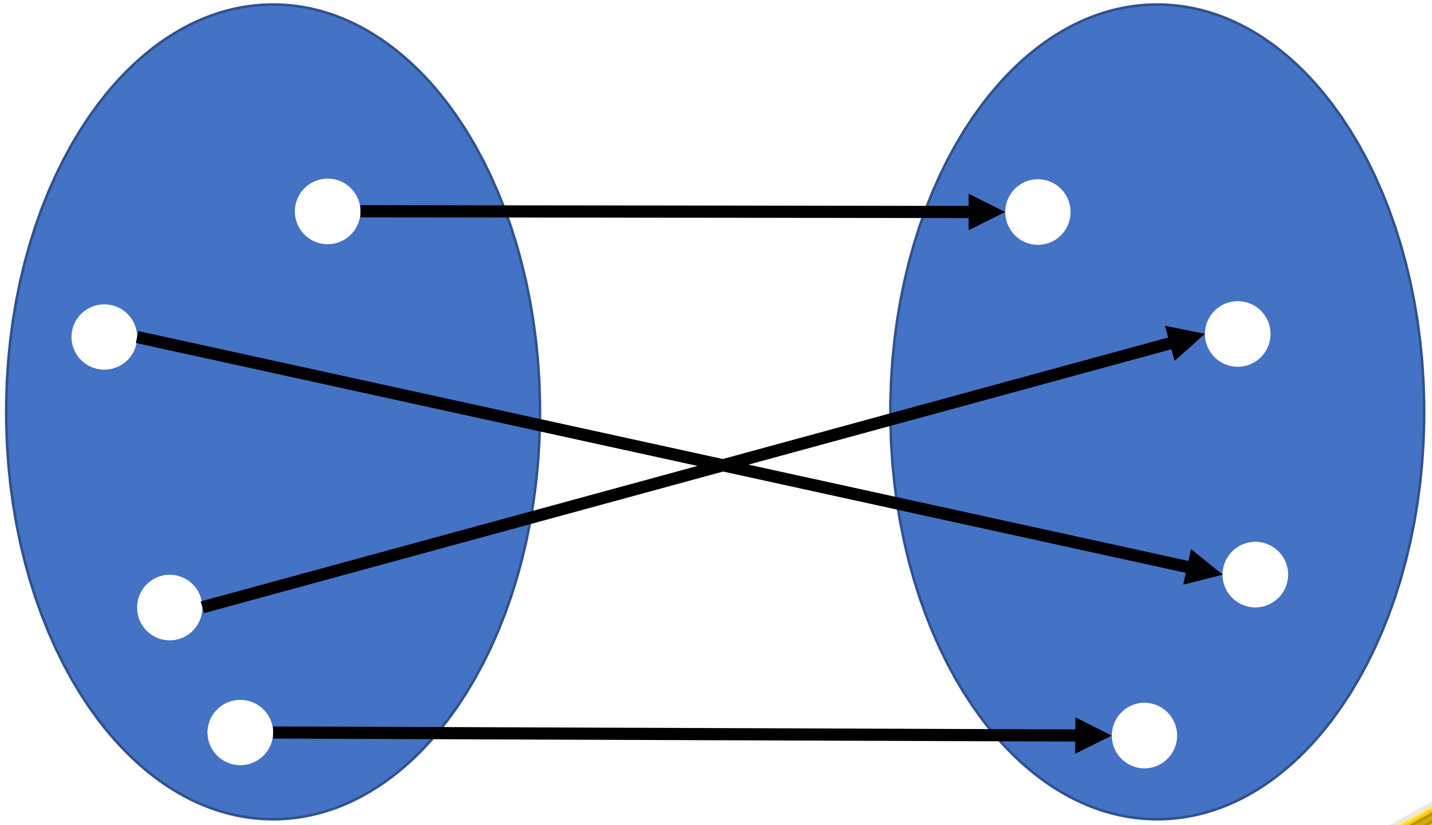
- Constrained PRPs for many natural classes of constraints *do not exist*

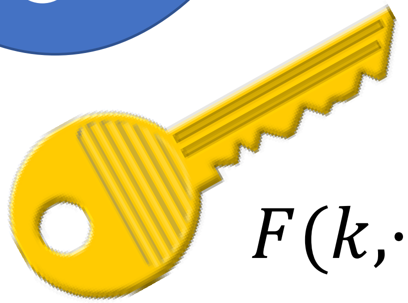
Our Results

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

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

Pseudorandom Permutations (PRPs)

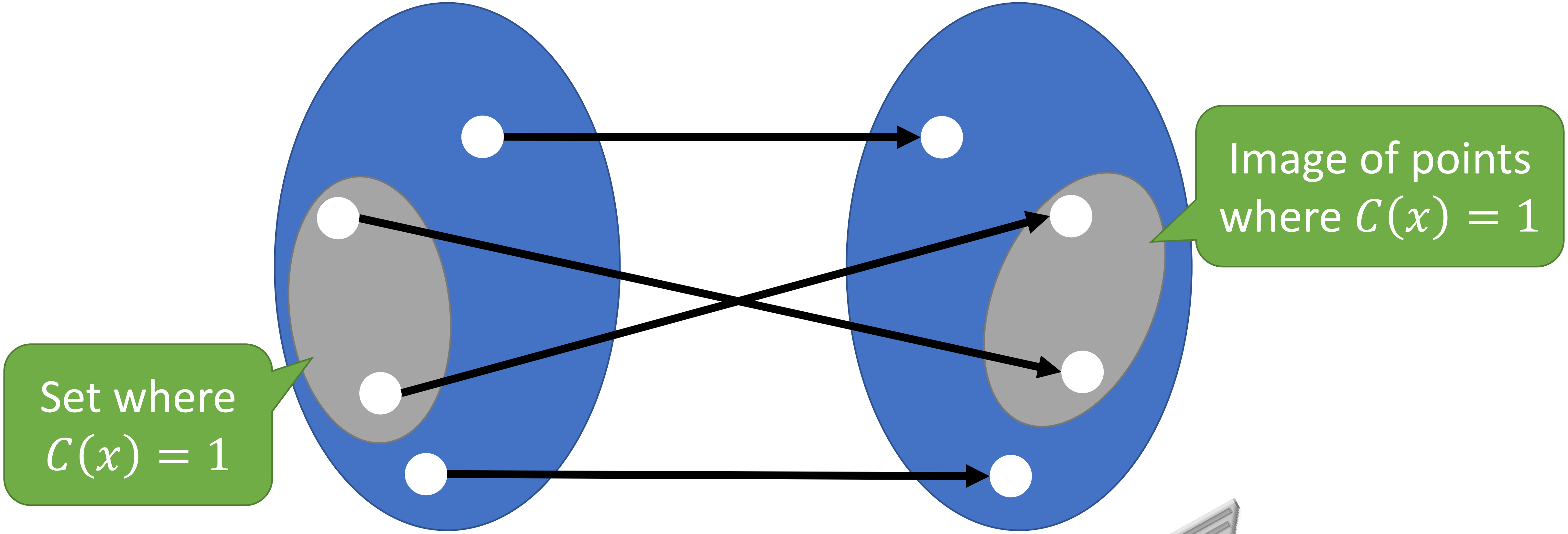


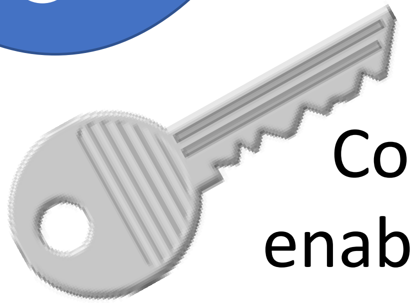


$F(k,\cdot)$ implements a permutation over $\mathcal{X}$

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

Constrained PRPs

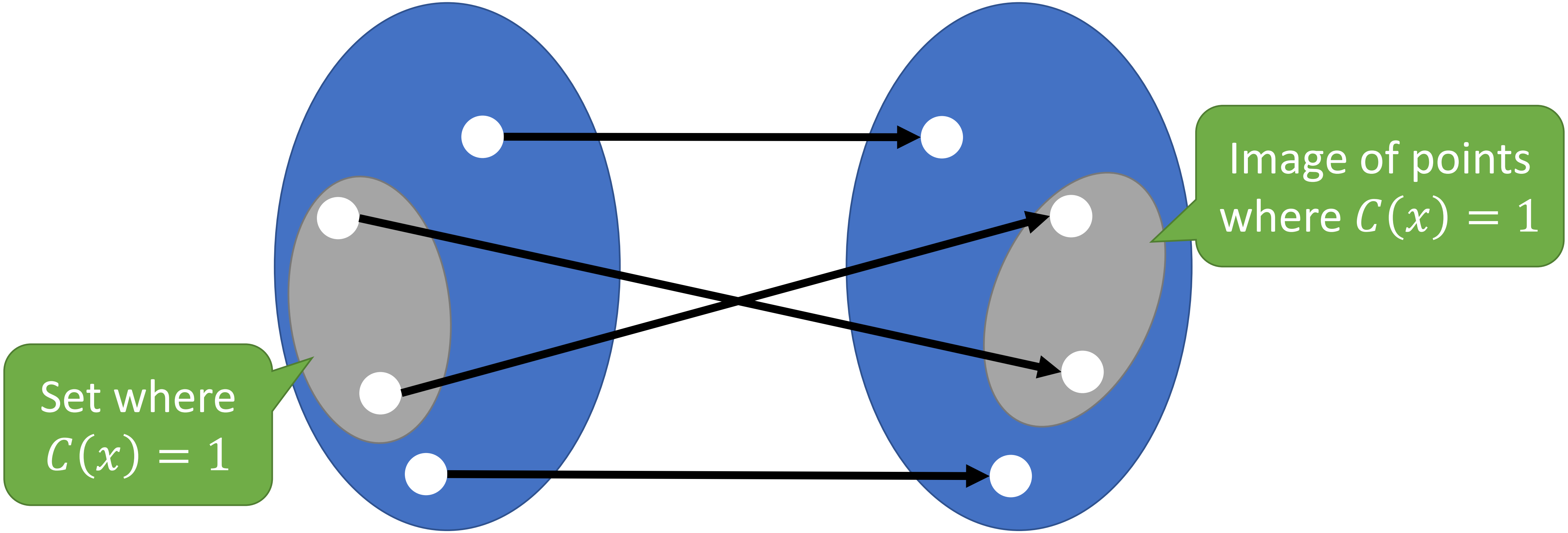




Constrained key
enables forward <u>and</u>
backward evaluation

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

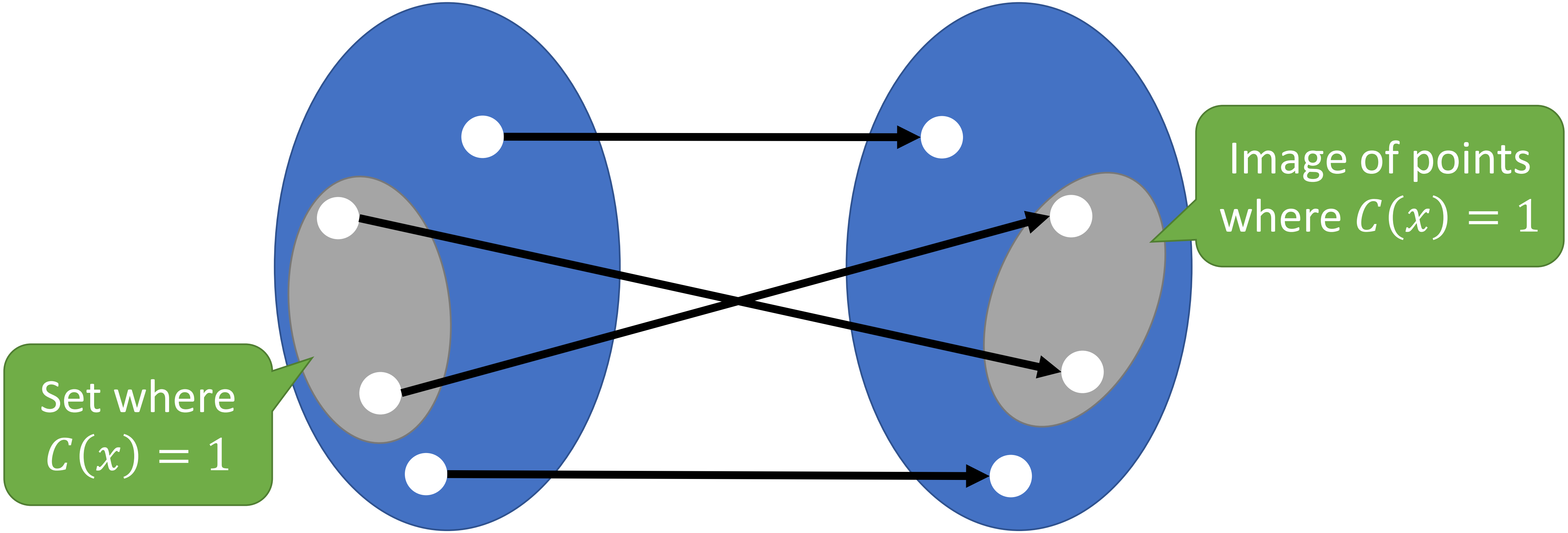
Constrained PRPs



Correctness:

- Forward evaluation when $C(x) = 1$

Constrained PRPs



Correctness:

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

Difficulties in Constraining PRPs

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

[See paper for details]

Difficulties in Constraining PRPs

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.

Puncturable PRPs do not exist.

[See paper for details]

Difficulties in Constraining PRPs

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.

Puncturable PRPs do not exist.

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

[See paper for details]

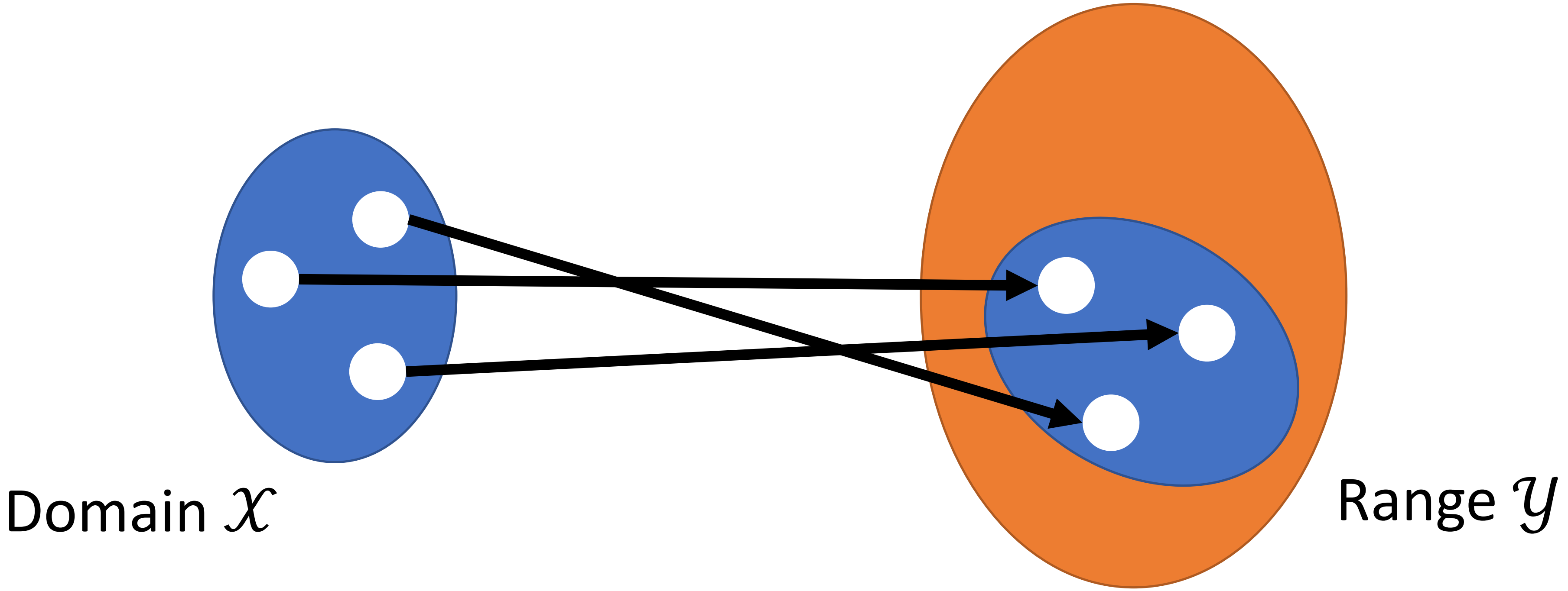
Relaxing the Notion

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

Lower bound critically relies on the set of points that satisfy the constraint being a non-negligible fraction of the range of the PRP

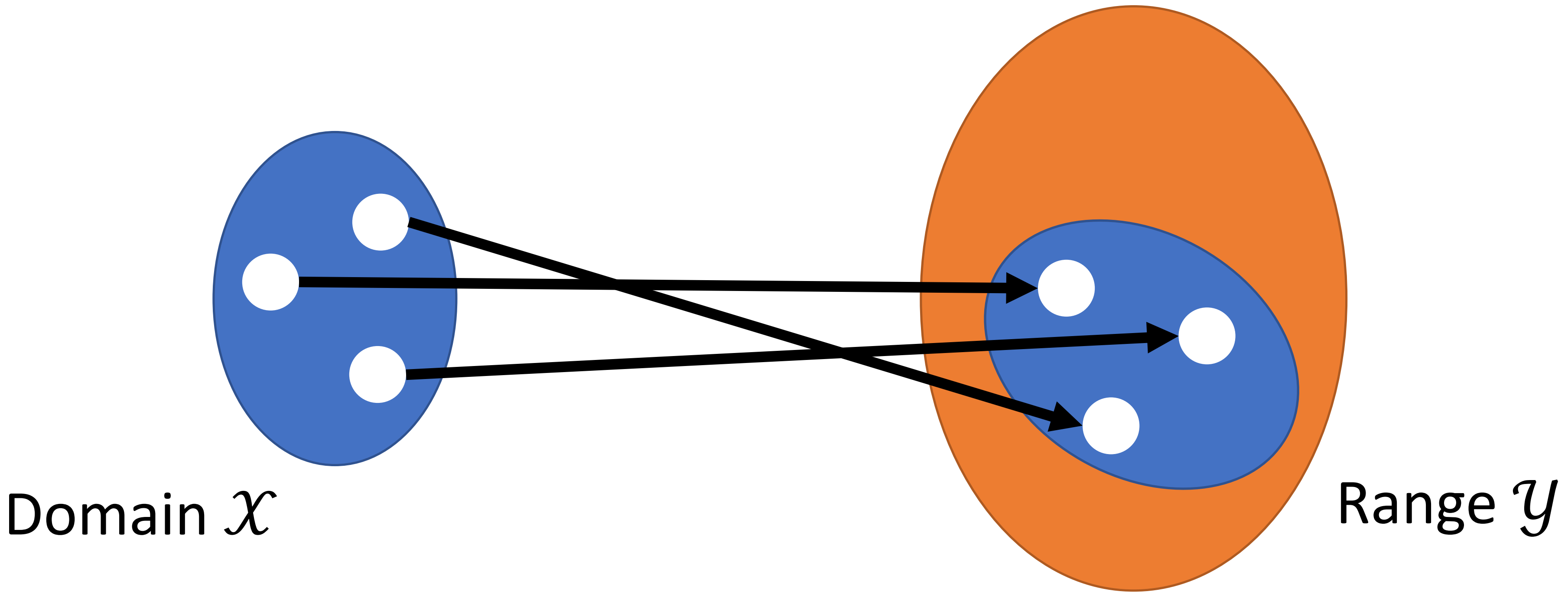
Relaxing the Notion

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



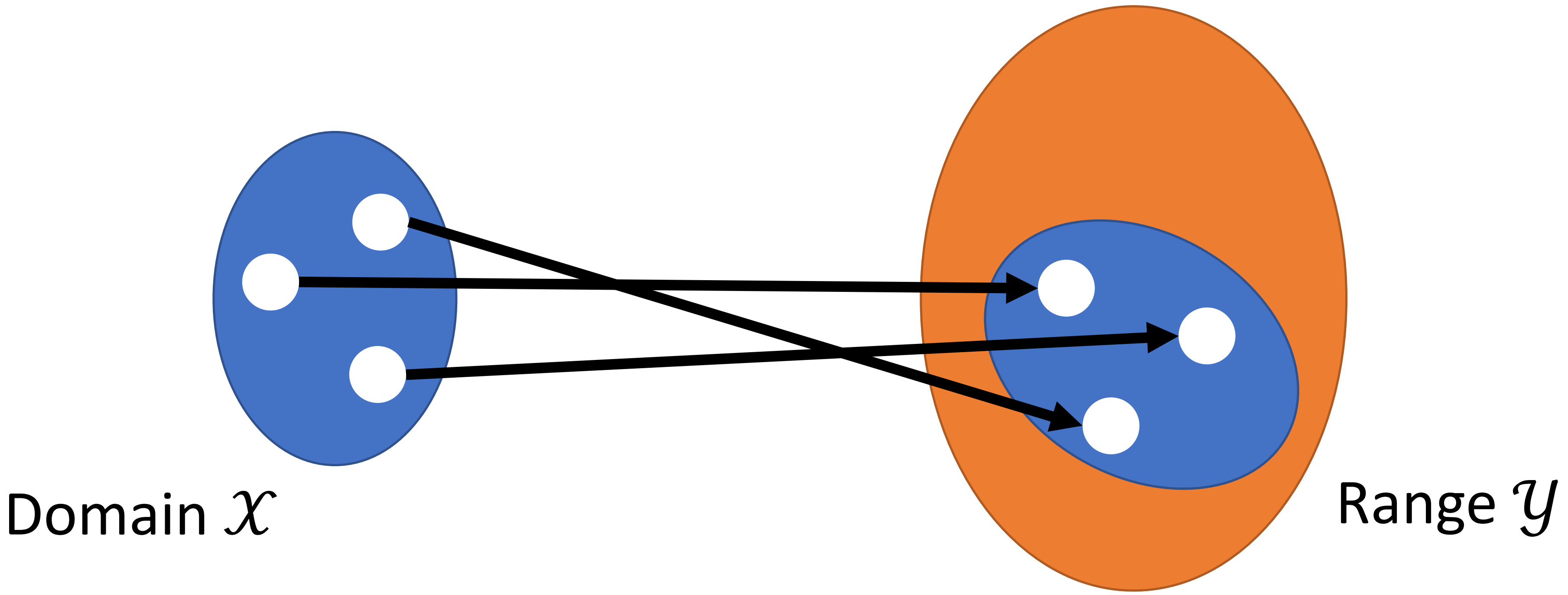
Relaxation: Allow range to be *much larger* than the domain

Invertible Pseudorandom Functions (IPFs)



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

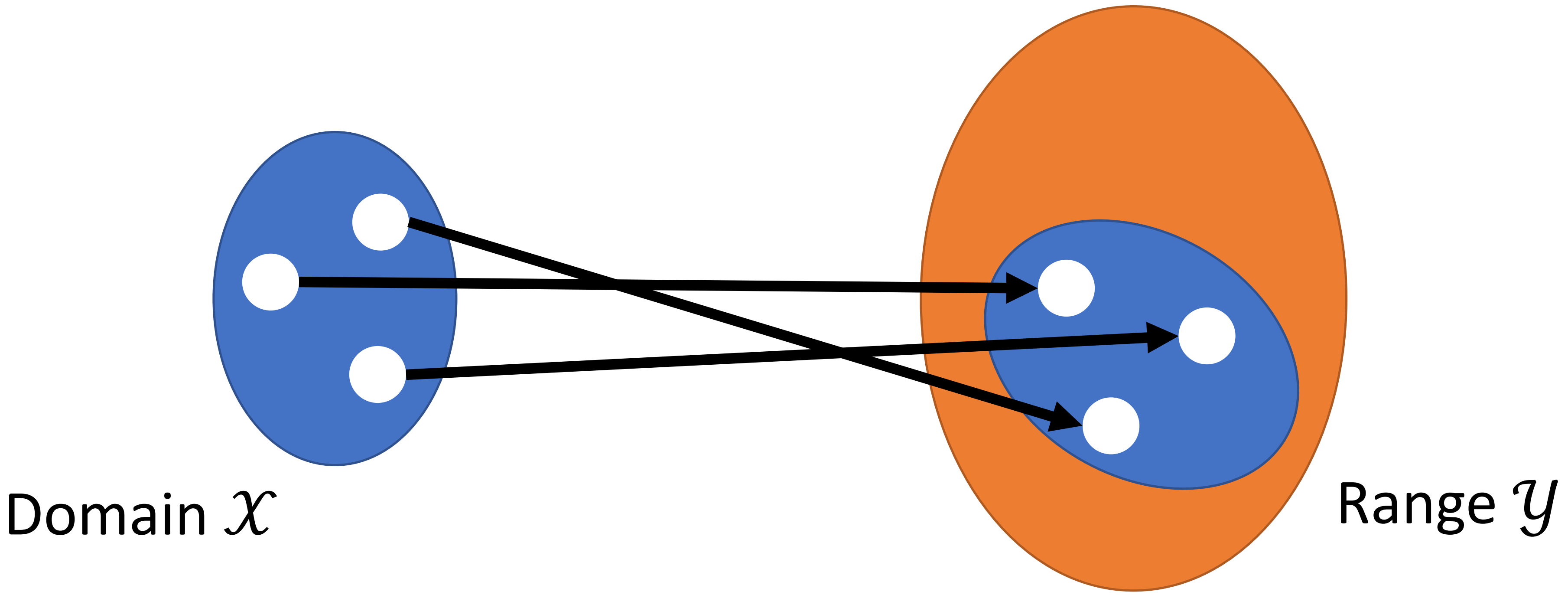
Invertible Pseudorandom Functions (IPFs)



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

- $F(k, \cdot)$ is injective for all $k \in \mathcal{K}$

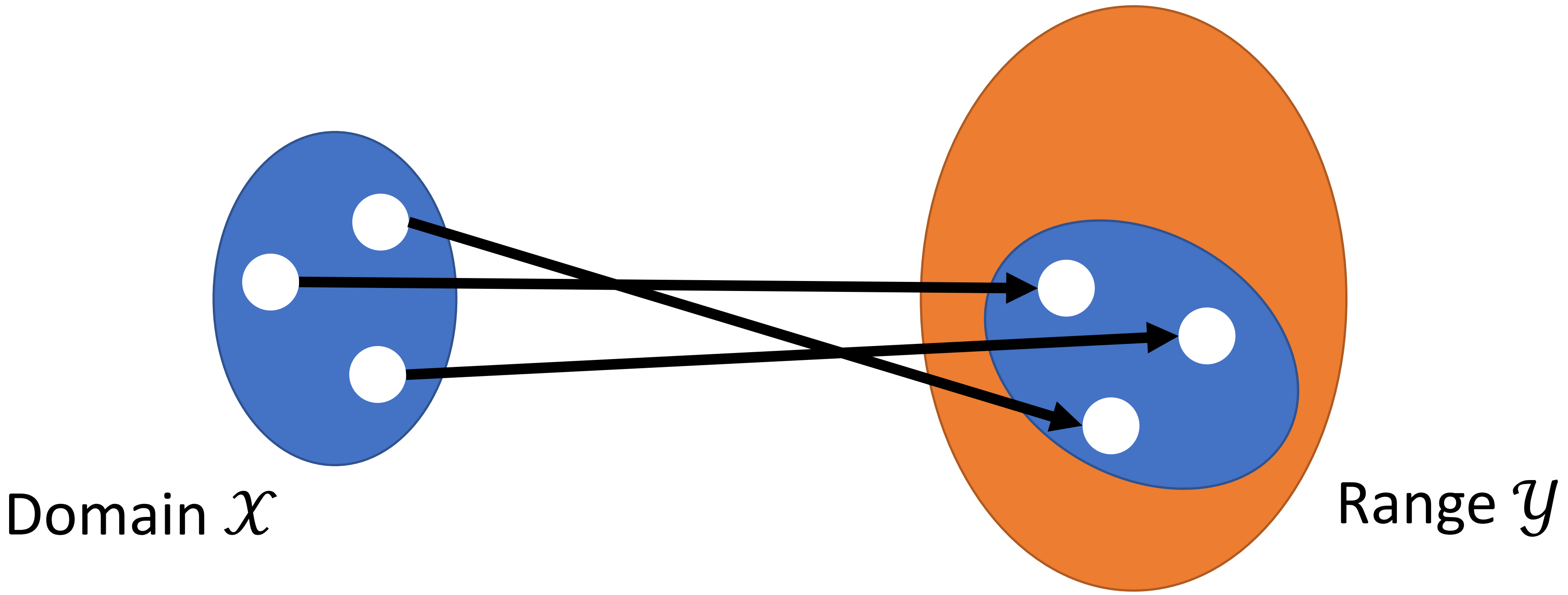
Invertible Pseudorandom Functions (IPFs)



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 $F^{-1}: \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$

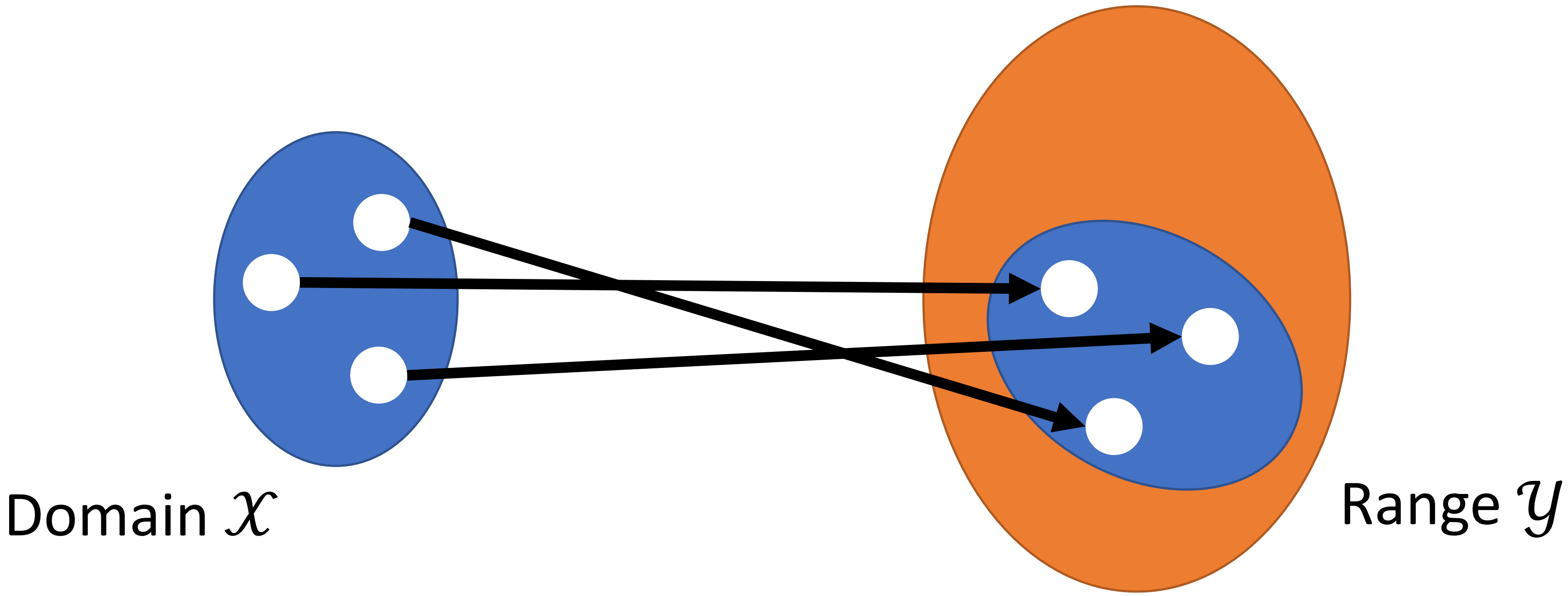
Invertible Pseudorandom Functions (IPFs)



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 $F^{-1}: \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$
- $F^{-1}(k, F(k, x)) = x$ for all $x \in \mathcal{X}$

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 $F^{-1}: \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$
- $F^{-1}\big(k, F(k, x)\big) = x$ 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)

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

- $F(k,\cdot)$ is injective for all $k \in \mathcal{K}$
- There exists an efficiently computable inverse $F^{-1}\colon \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$
- $F^{-1}\big(k, F(k,x)\big) = x$ 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)

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

$x \in \mathcal{X}$

$F(k, x)$

$y \in \mathcal{Y}$

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

Pseudorandom

$\approx_c$

Set of injective functions

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

$x \in \mathcal{X}$

$f(x)$

$y \in \mathcal{Y}$

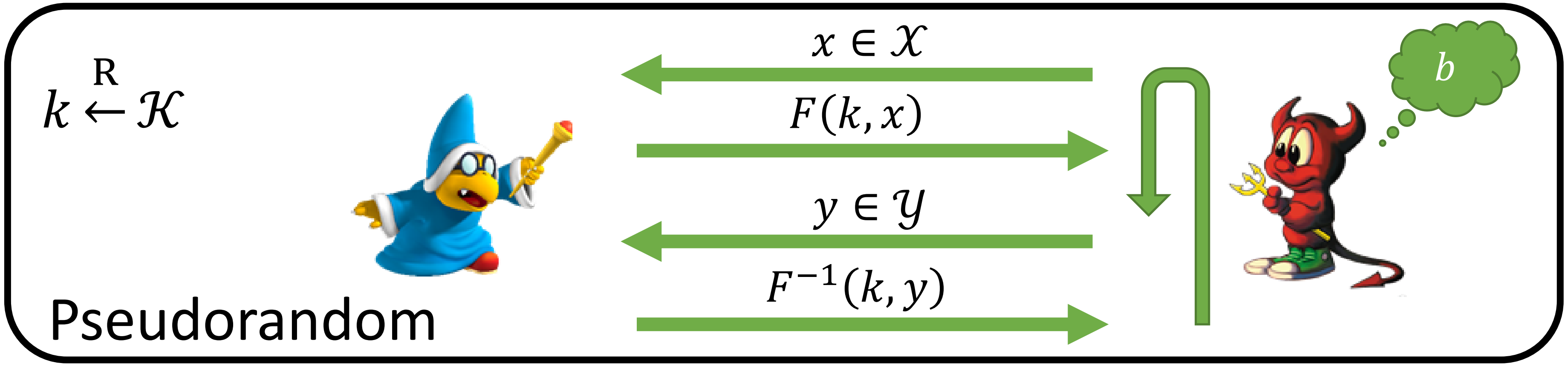
$f^{-1}(y)$

Random

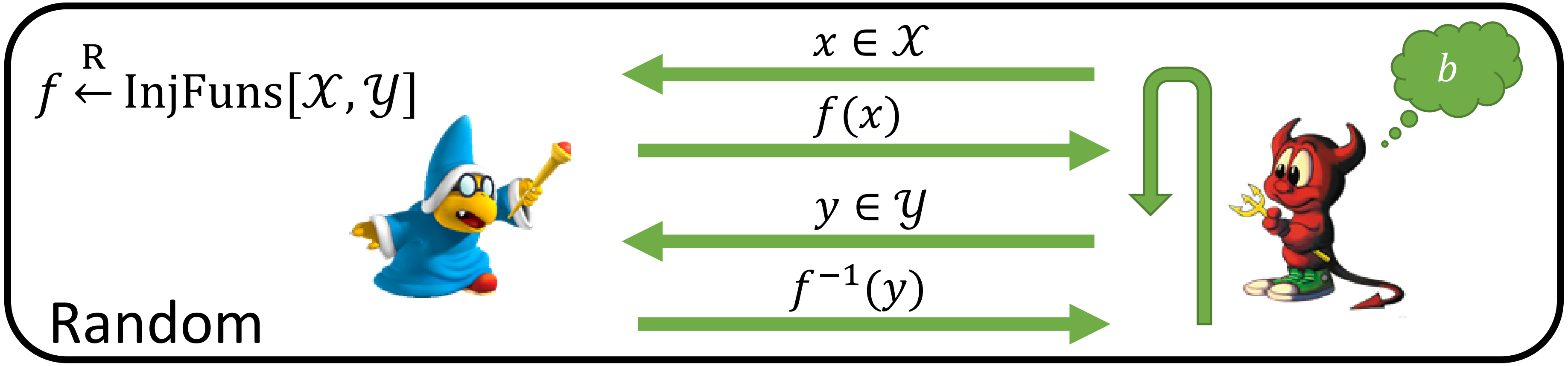
Outputs $\perp$ if y has no inverse under f

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

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

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

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 $C(x) = 1$

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

A Puncturable IPF

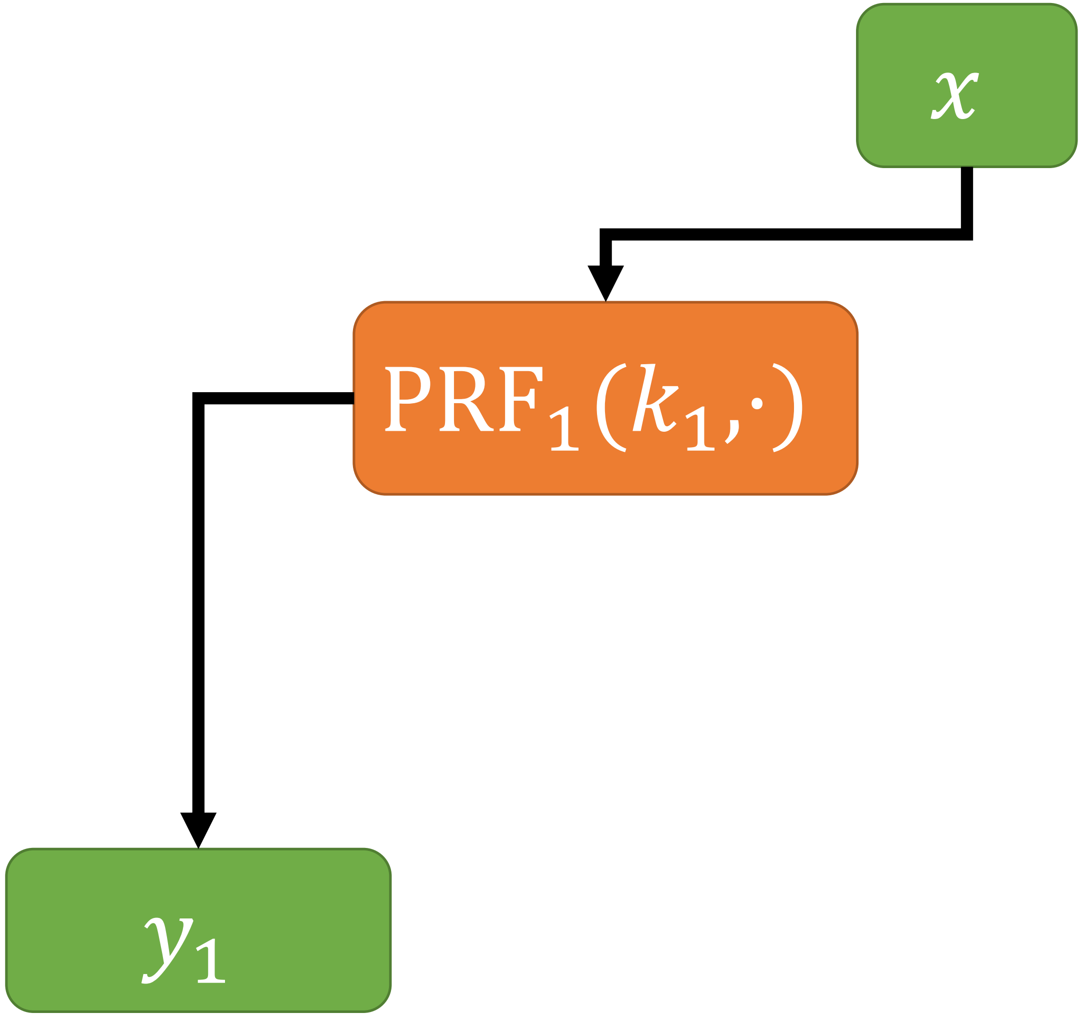
Starting point: DAE construction called synthetic IV (SIV) [RS06]

A Puncturable IPF

$\boxed{x}$

Starting point: DAE construction called synthetic IV (SIV) [RS06]

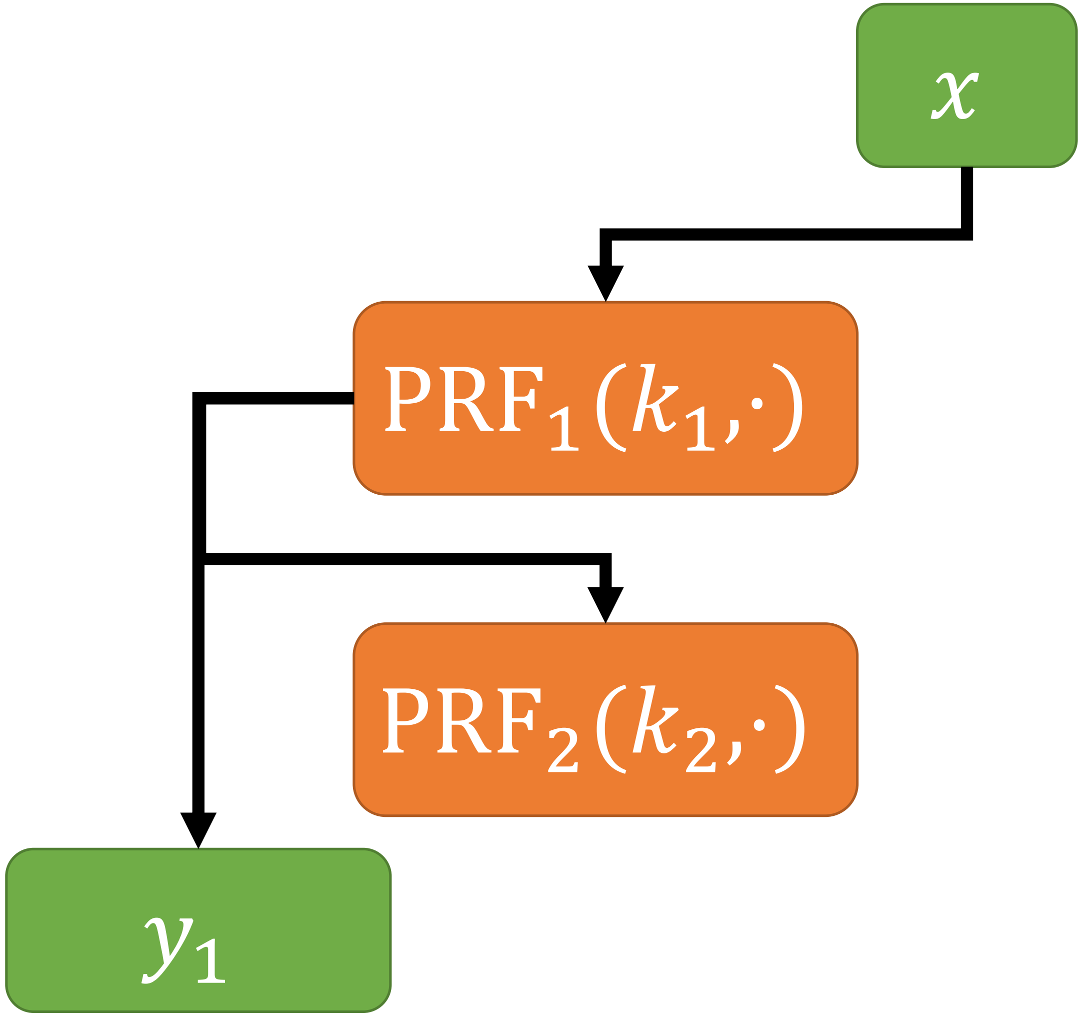
A Puncturable IPF



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

Starting point: DAE construction called synthetic IV (SIV) [RS06]

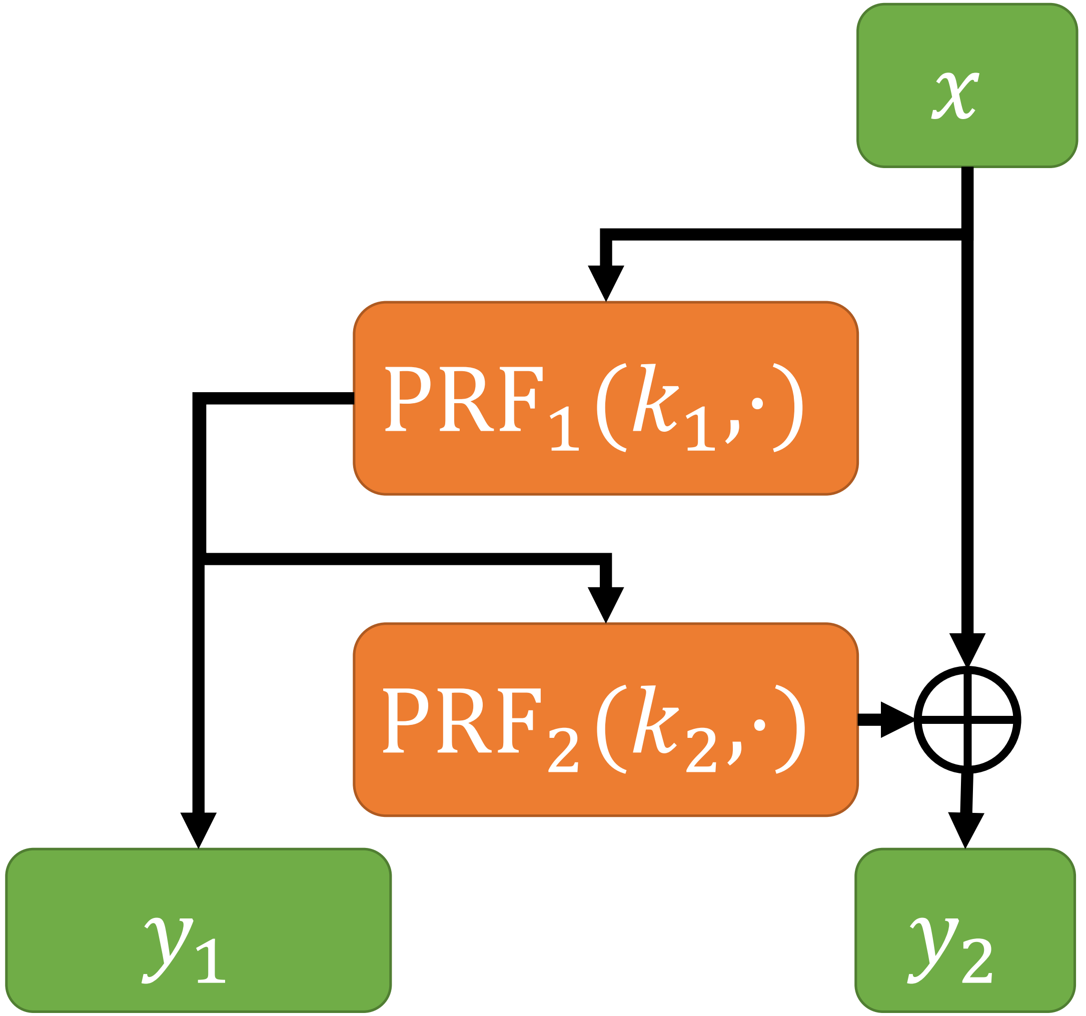
A Puncturable IPF



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

Starting point: DAE construction called synthetic IV (SIV) [RS06]

A Puncturable IPF

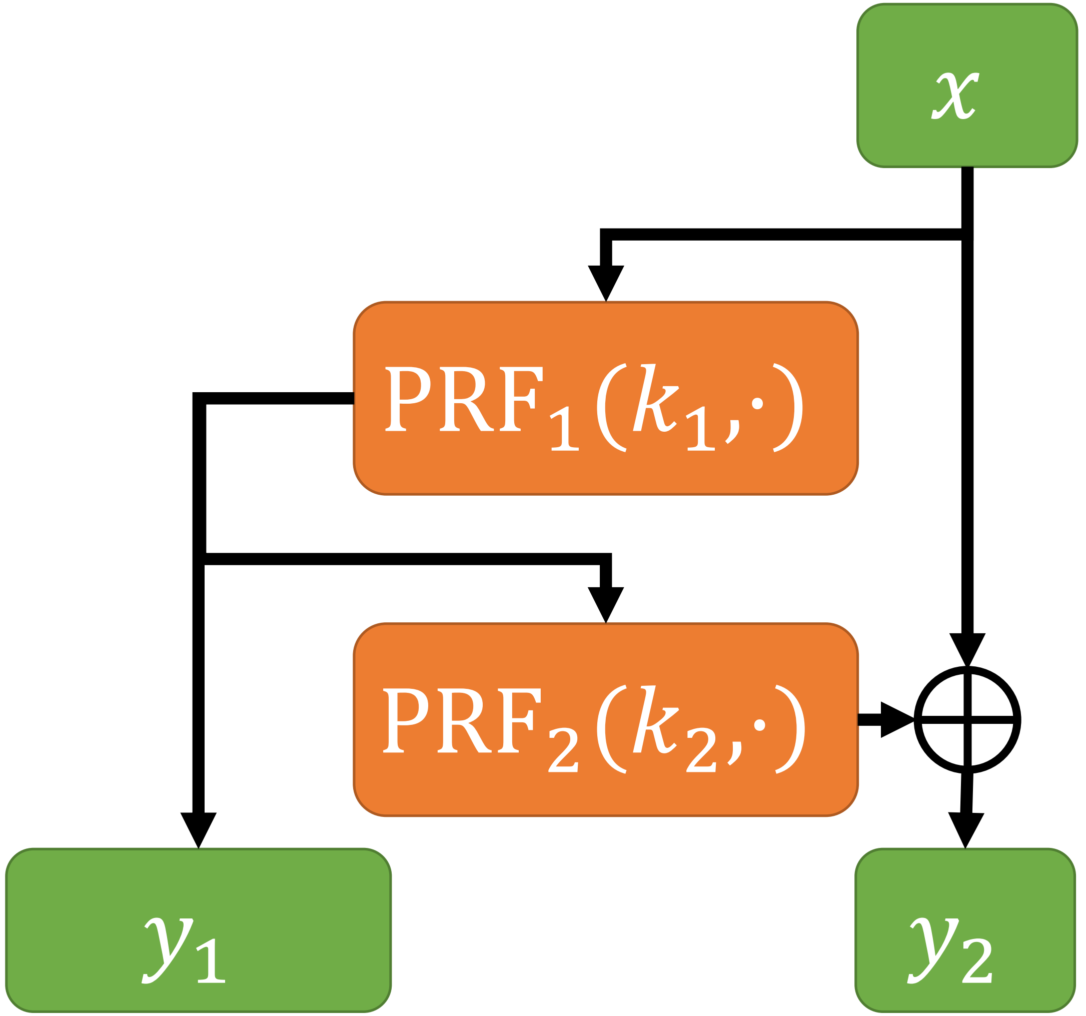


$y_1 = \mathrm{PRF}_1(k_1, x)$

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

Starting point: DAE construction called synthetic IV (SIV) [RS06]

A Puncturable IPF



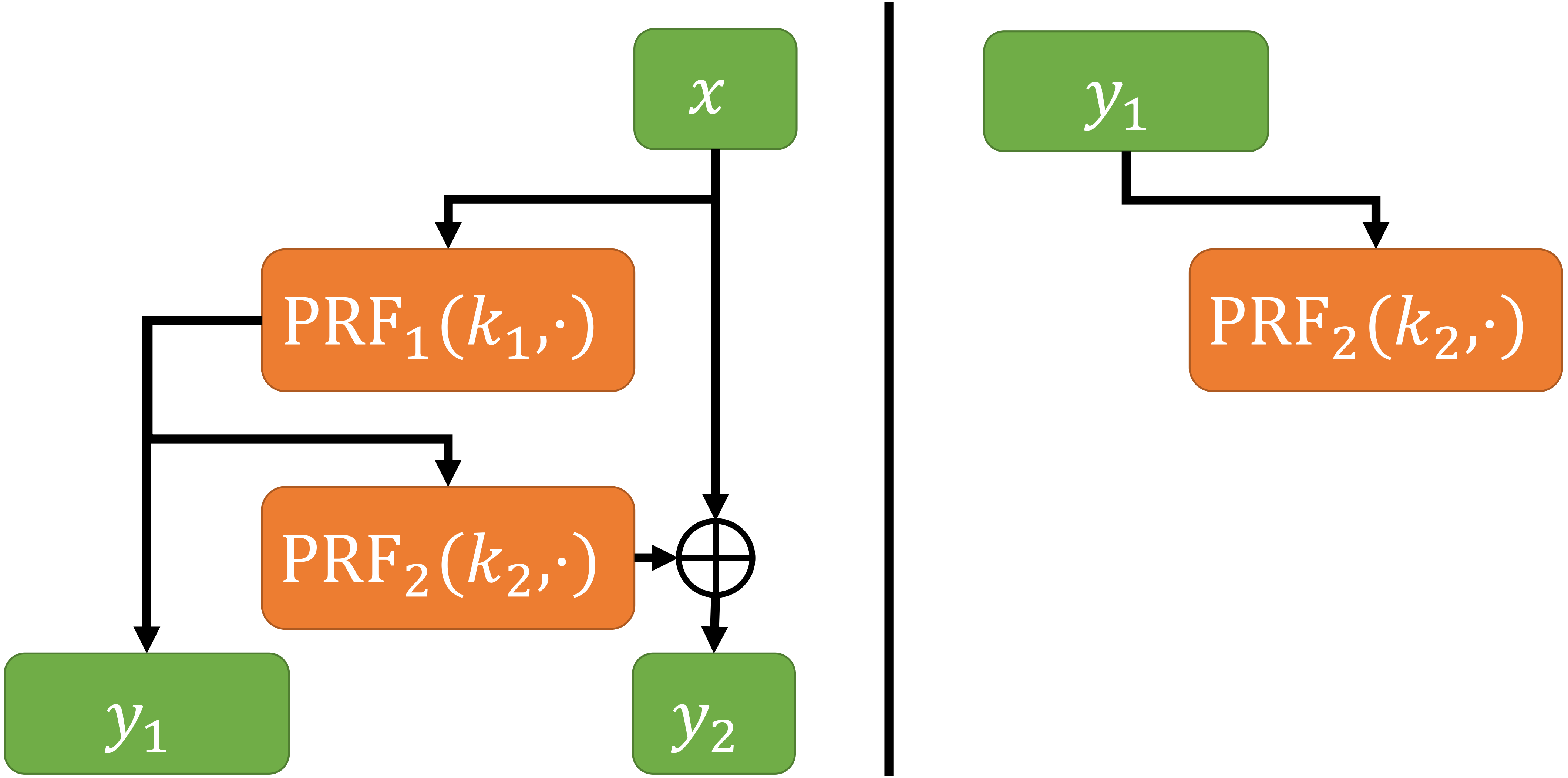
$y_1 = \mathrm{PRF}_1(k_1, x)$

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

Starting point: DAE construction called synthetic IV (SIV) [RS06]

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

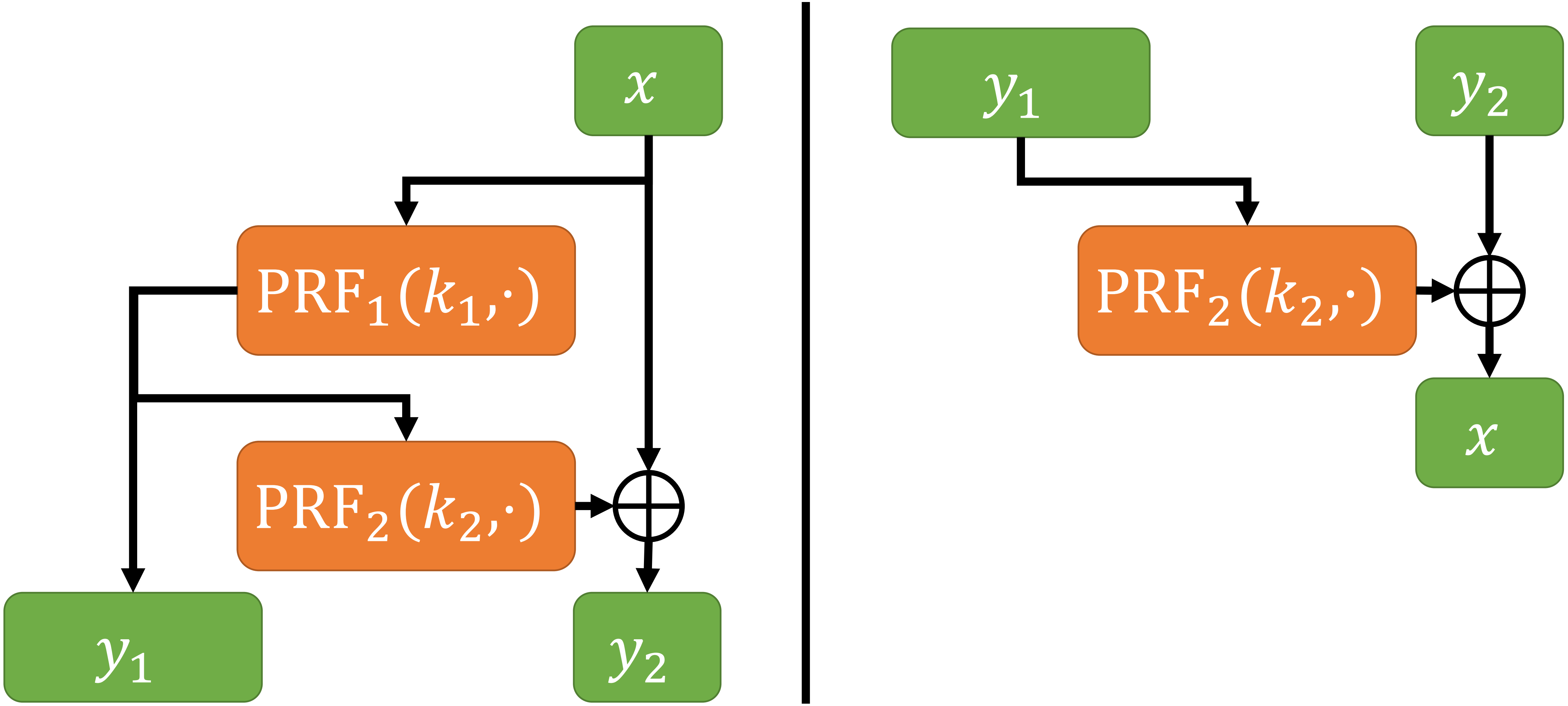
A Puncturable IPF



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

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

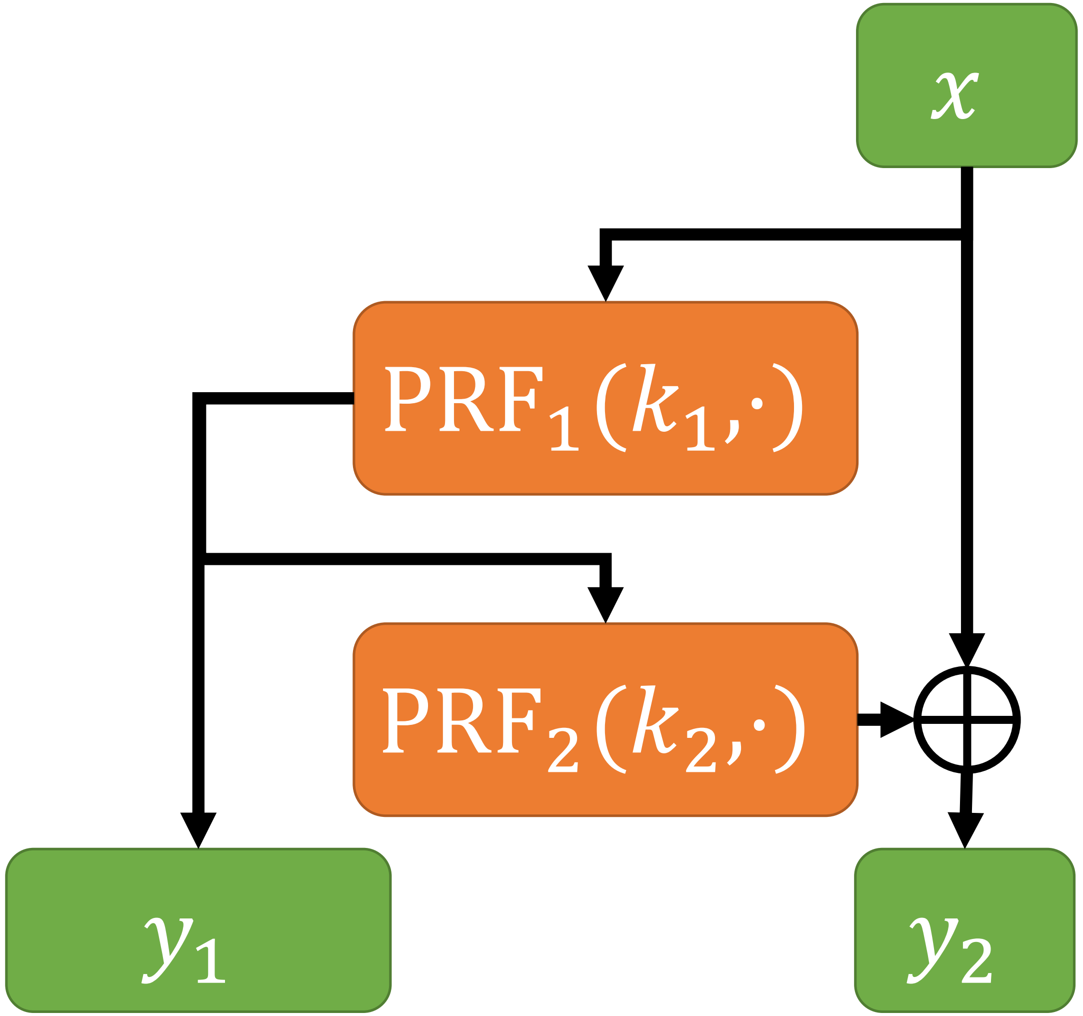
A Puncturable IPF



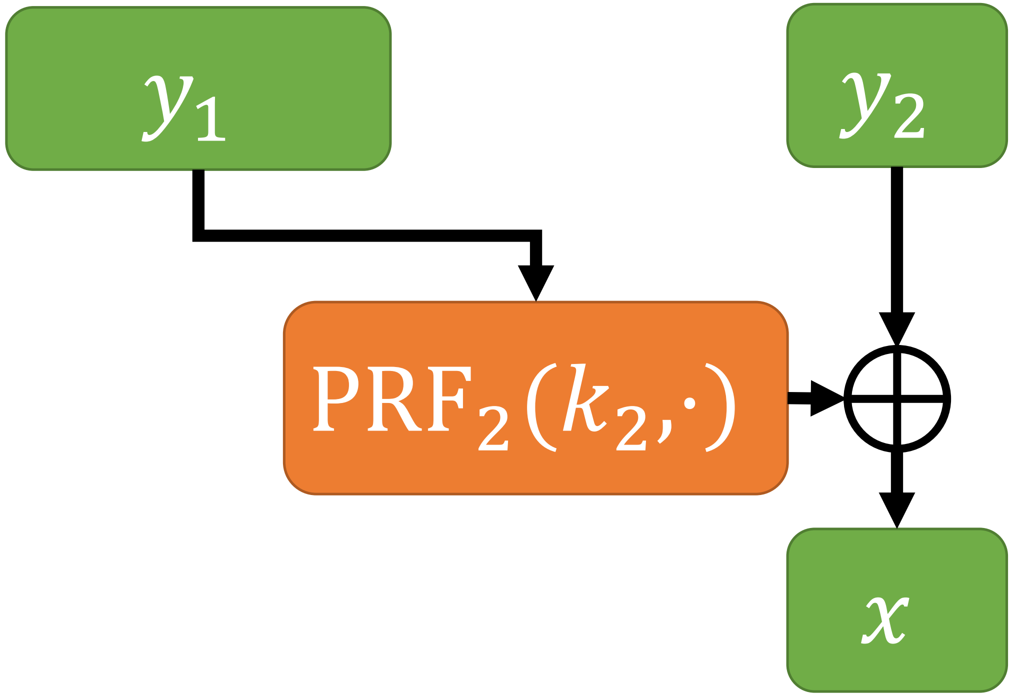
$y_1 = \mathrm{PRF}_1(k_1, x)$

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

A Puncturable IPF

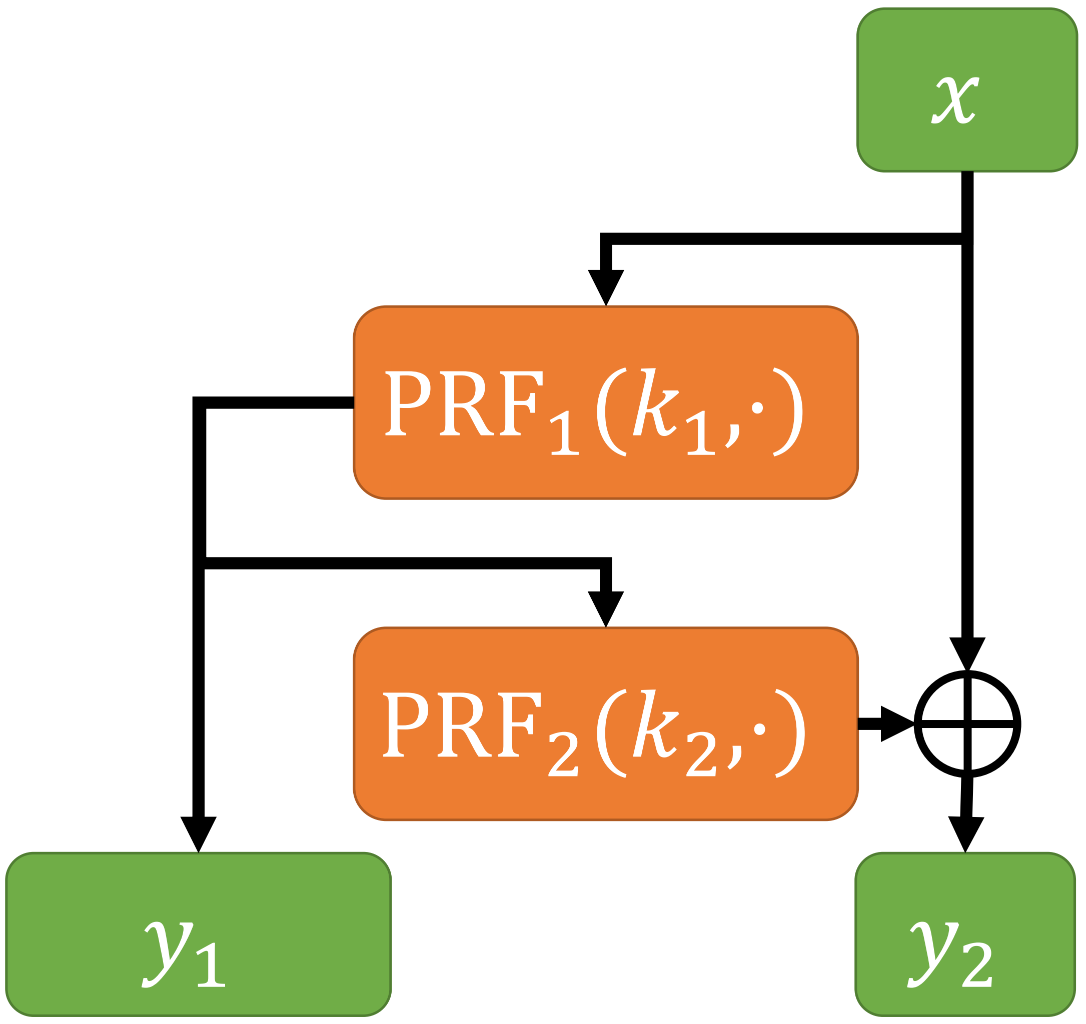


$y_1 = \mathrm{PRF}_1(k_1, x)$ $\quad$ $y_2 = x \oplus \mathrm{PRF}_2(k_2, y_1)$



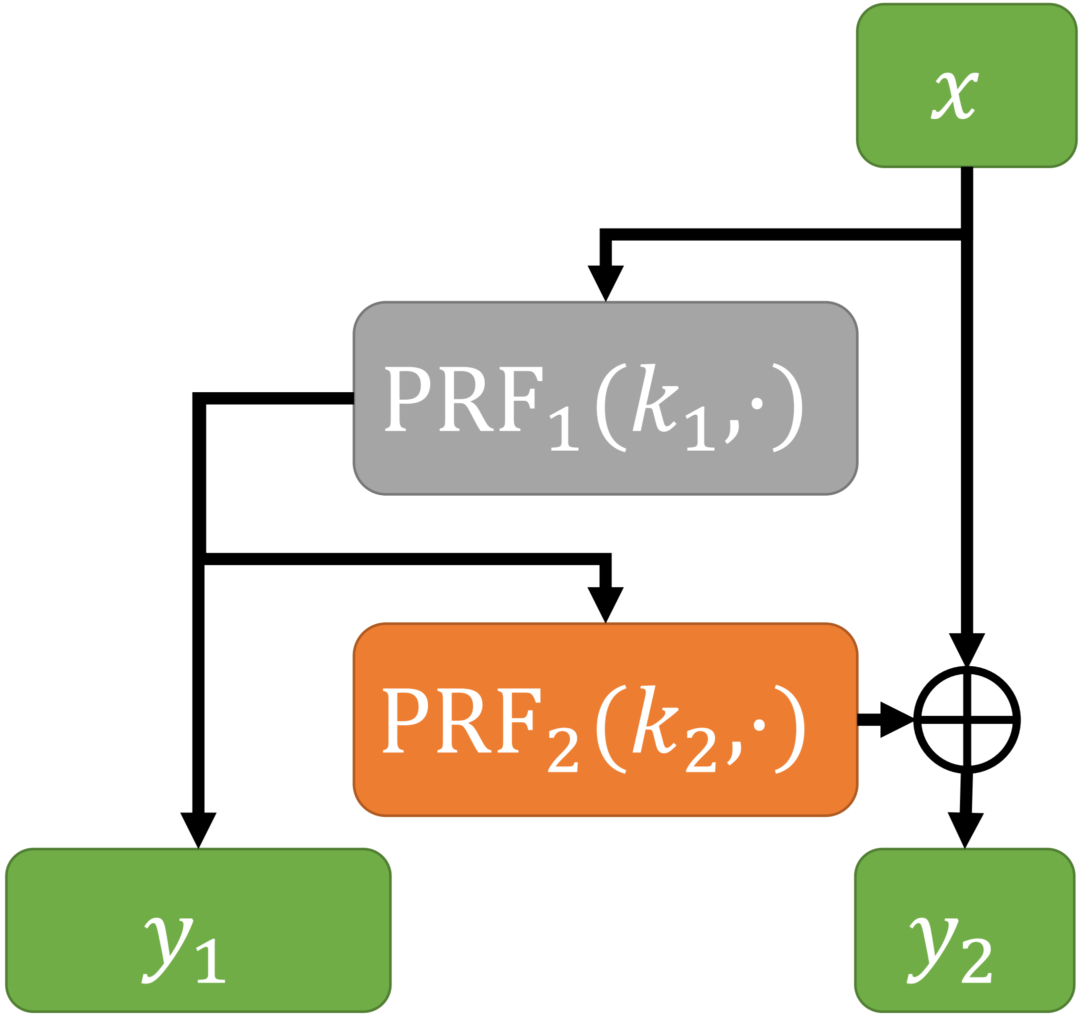
Verify $y_1 = \mathrm{PRF}(k_1, x)$ and output $\perp$ if $y_1 \neq \mathrm{PRF}(k_1, x)$

A Puncturable IPF



How to puncture this construction?

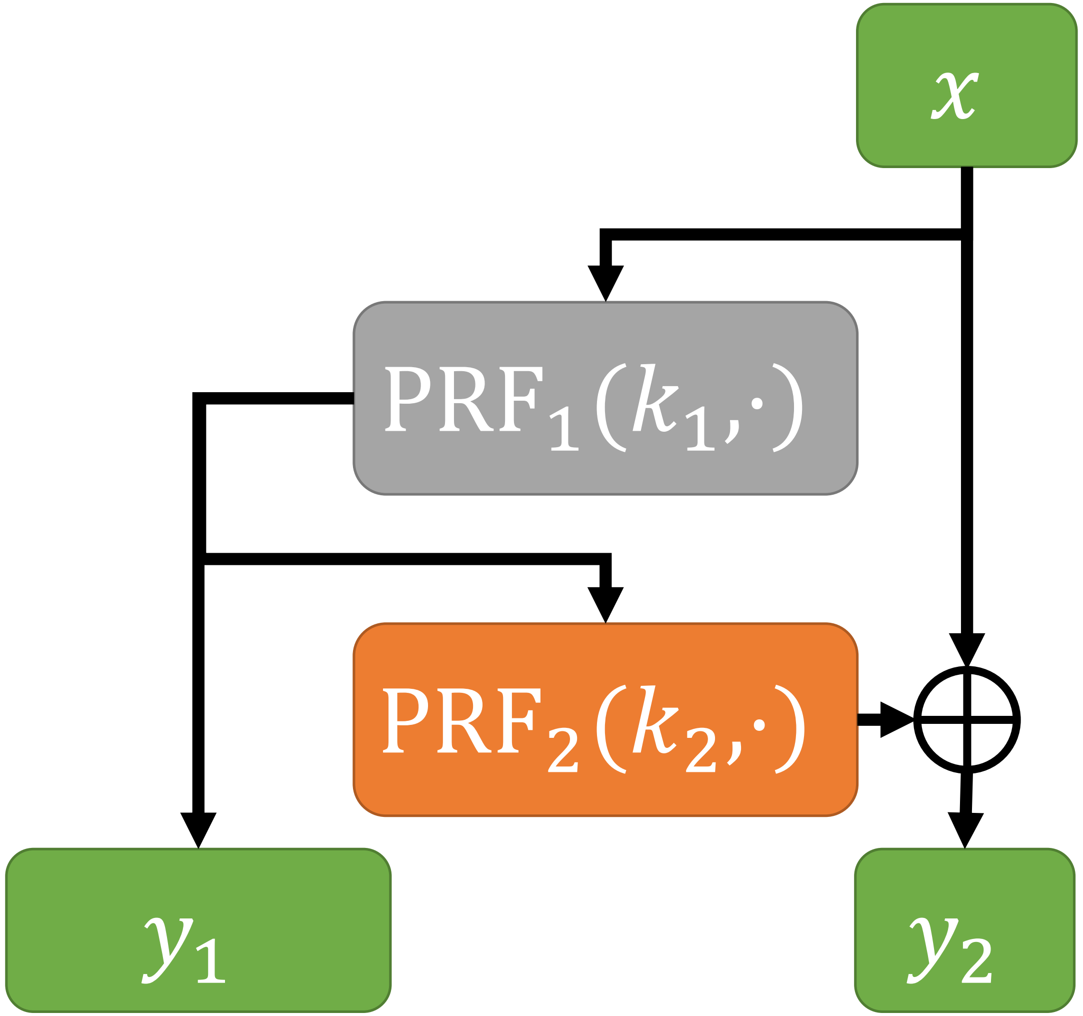
A Puncturable IPF



How to puncture this construction?

First attempt: only puncture k_1 at x^*

A Puncturable IPF

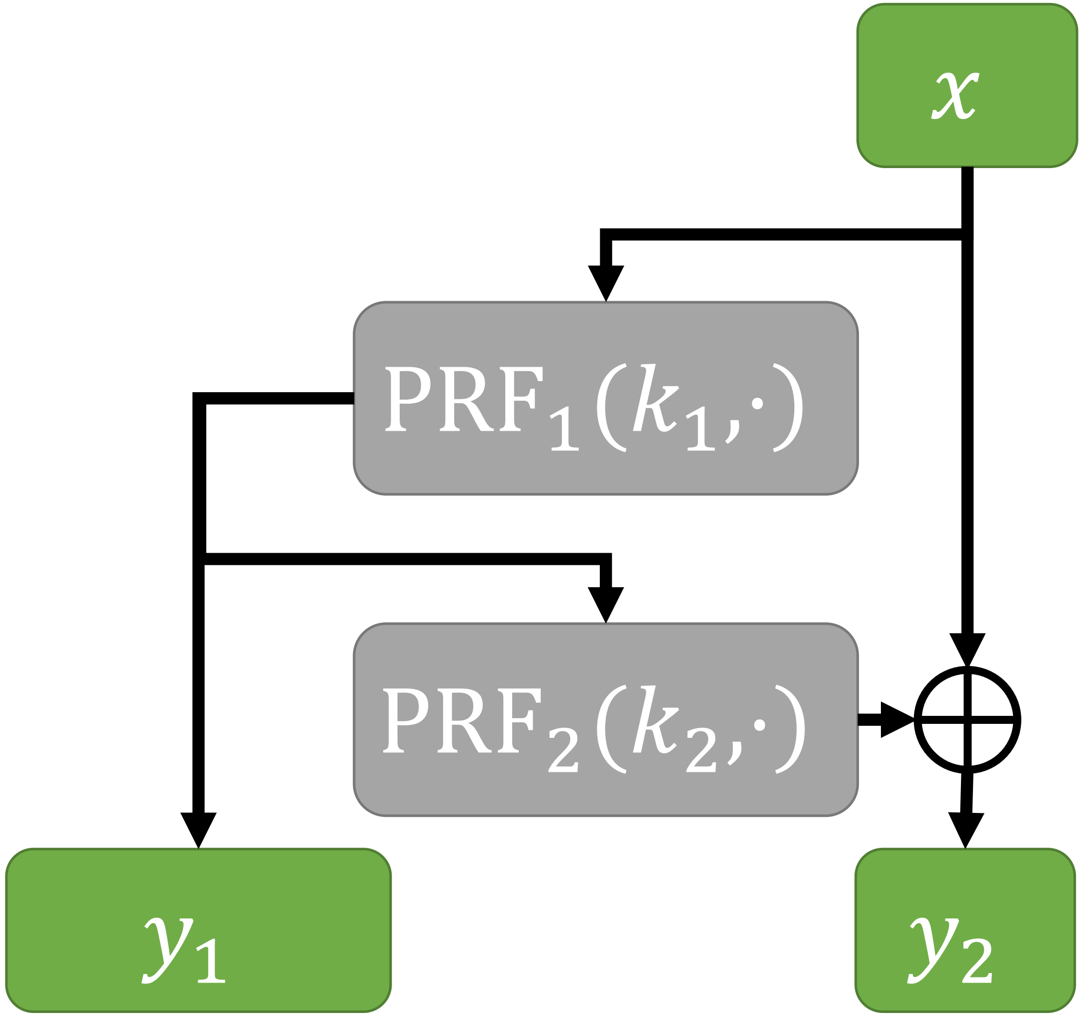


How to puncture this construction?

First attempt: only puncture k_1 at x^*

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

A Puncturable IPF



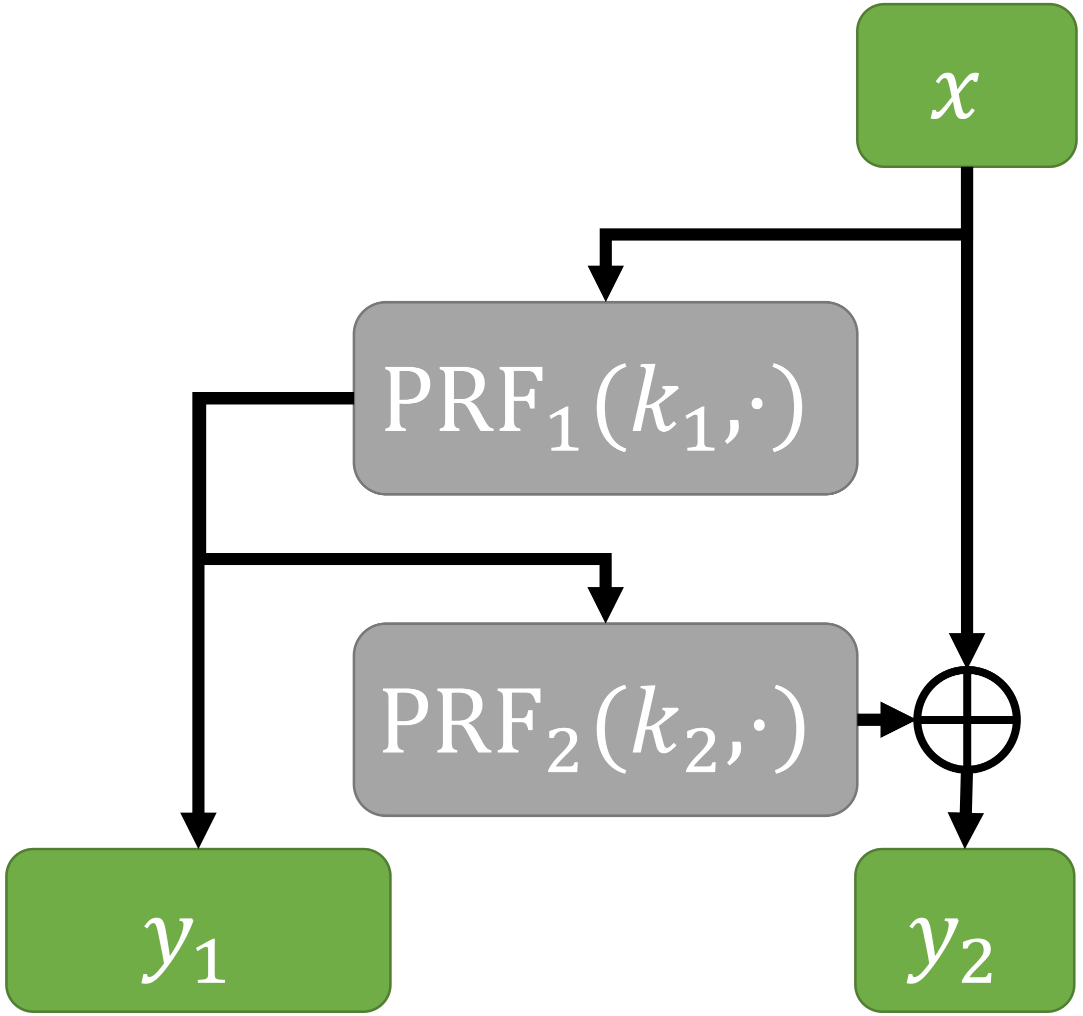
How to puncture this construction?

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

A Puncturable IPF



How to puncture this construction?

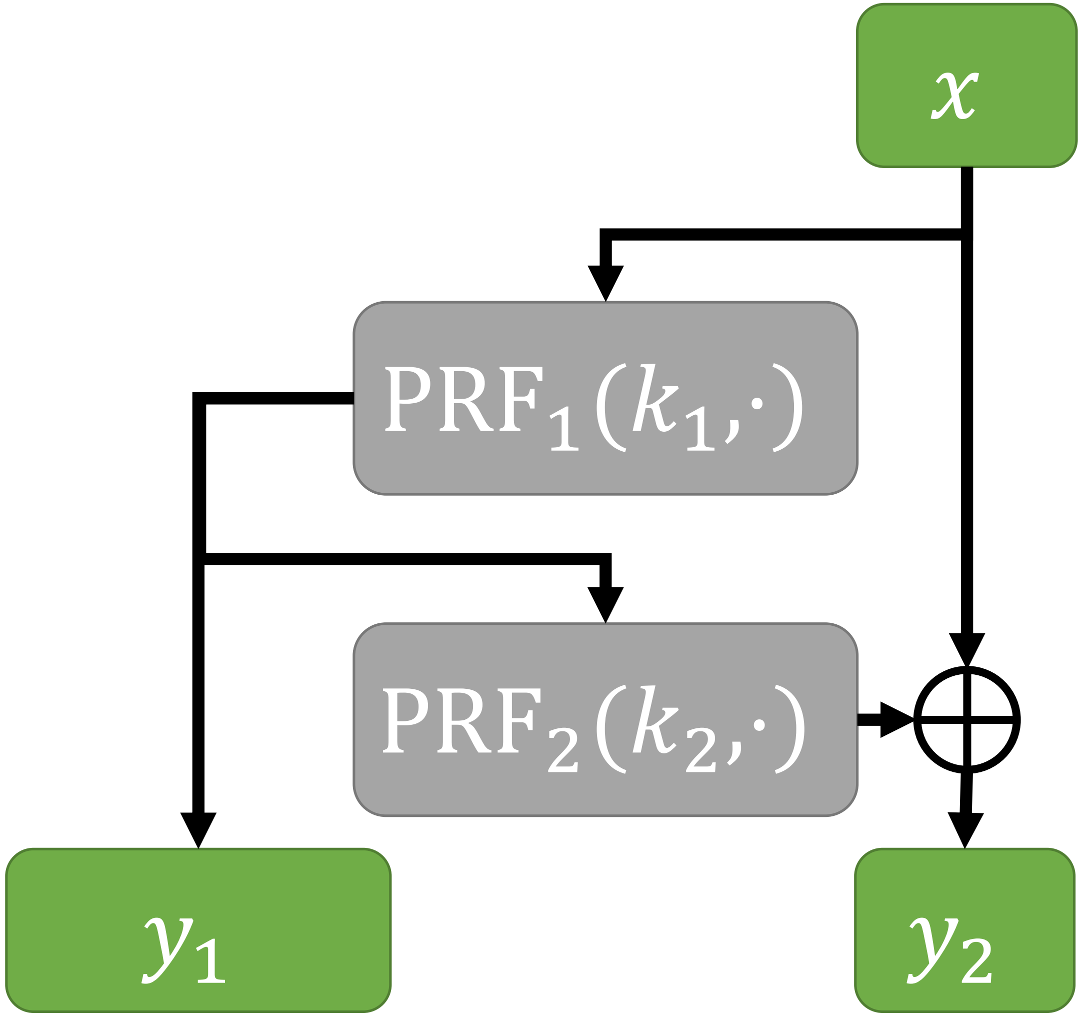
First attempt: only puncture k_1 at x^*

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!

A Puncturable IPF



How to puncture this construction?

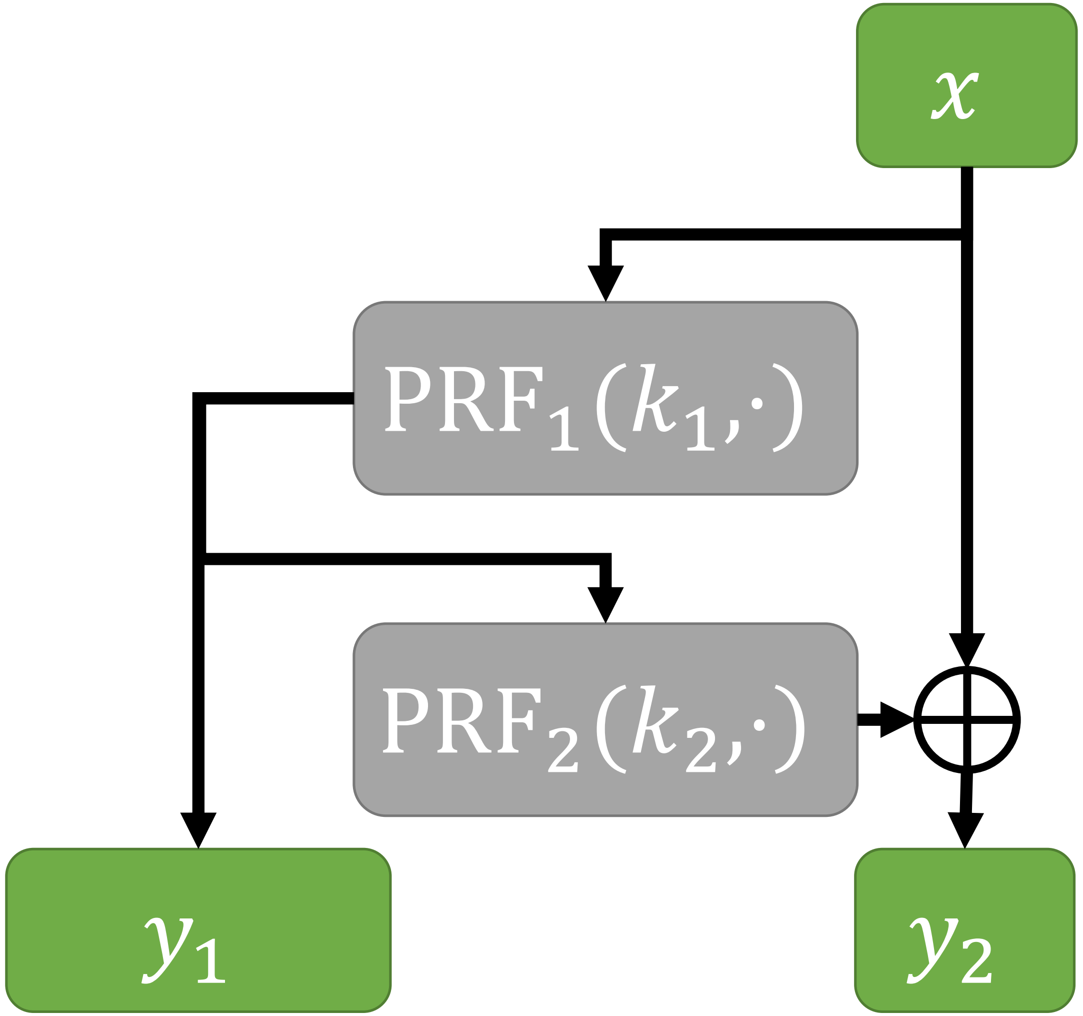
First attempt: only puncture k_1 at x^*

Solution: use <u>private</u> puncturing

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

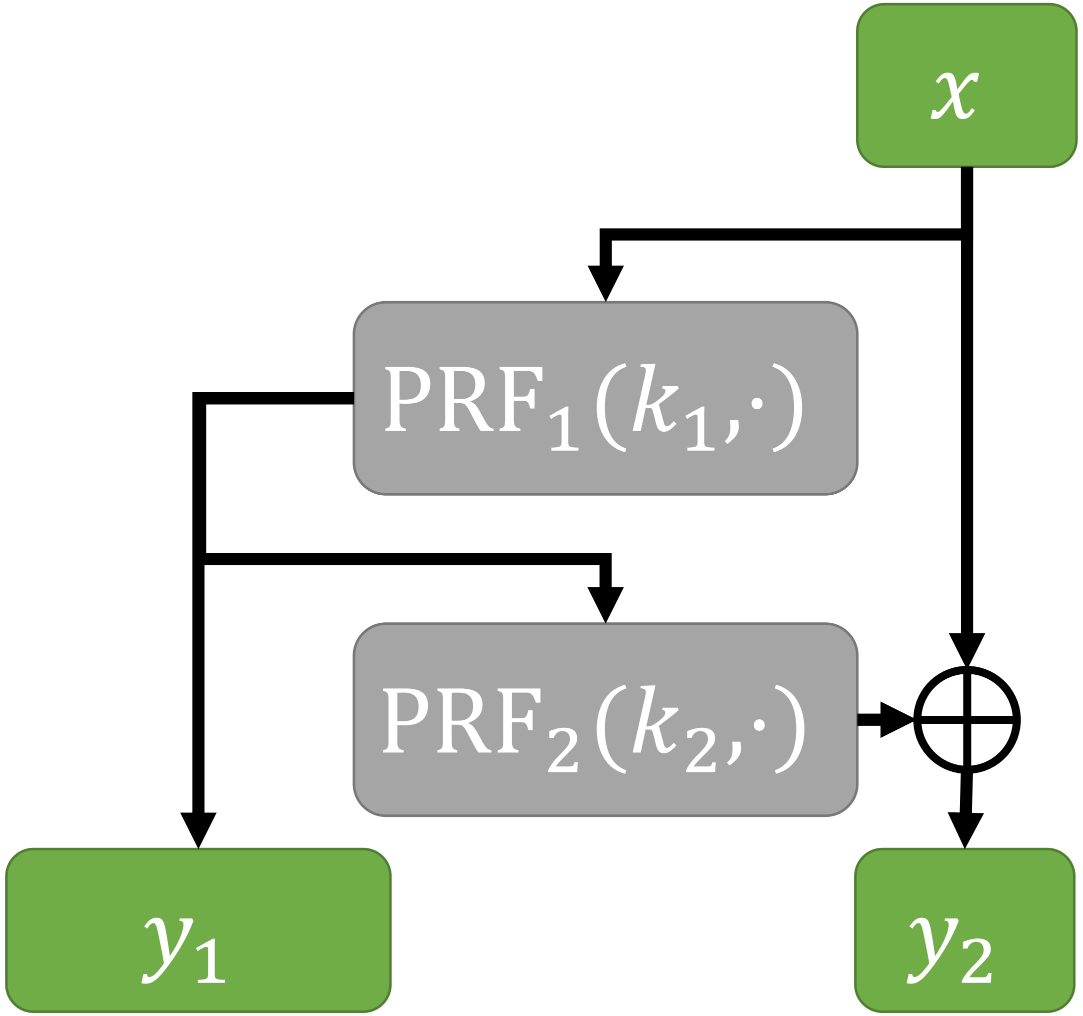
Punctured key reveals punctured point!

A Puncturable IPF



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

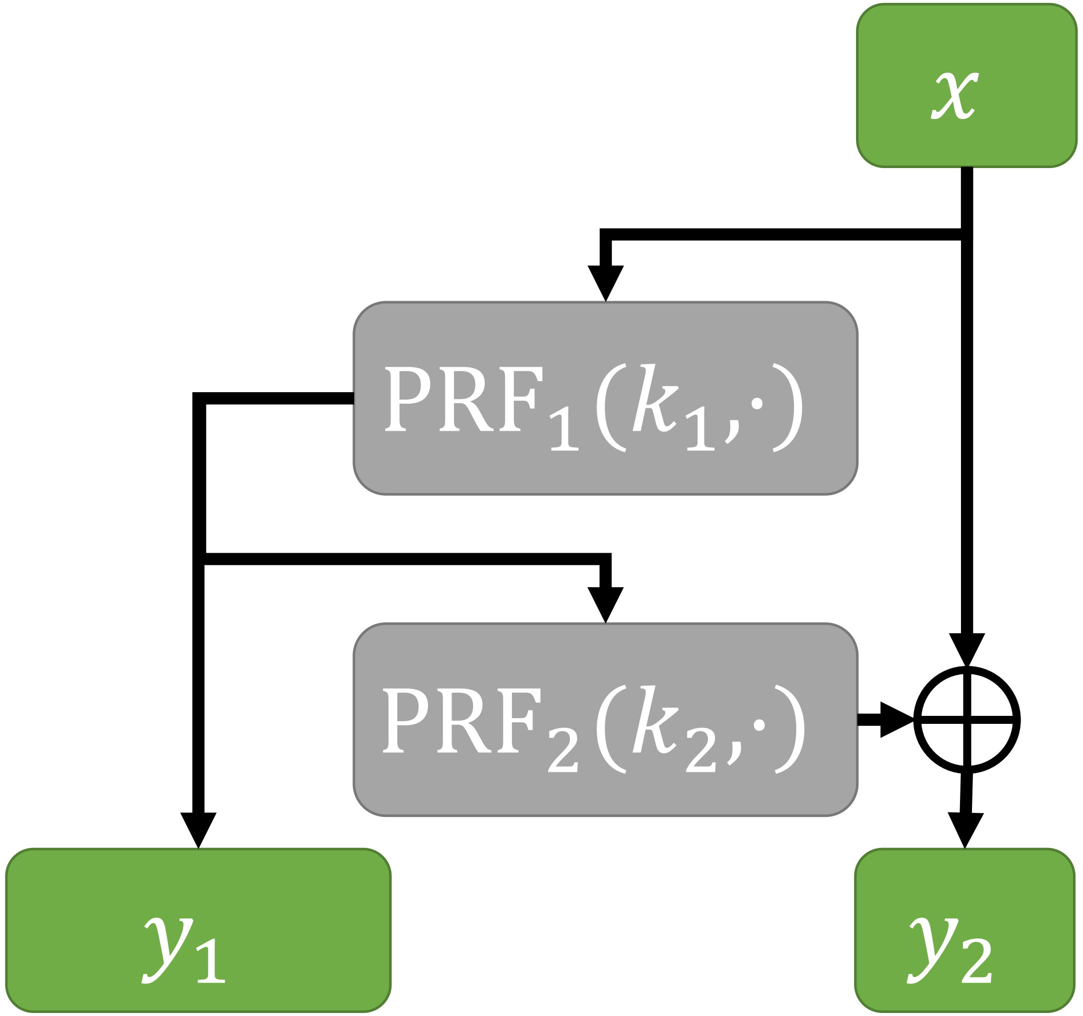
A Puncturable IPF



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

Punctured key (punctured at x^*):

A Puncturable IPF

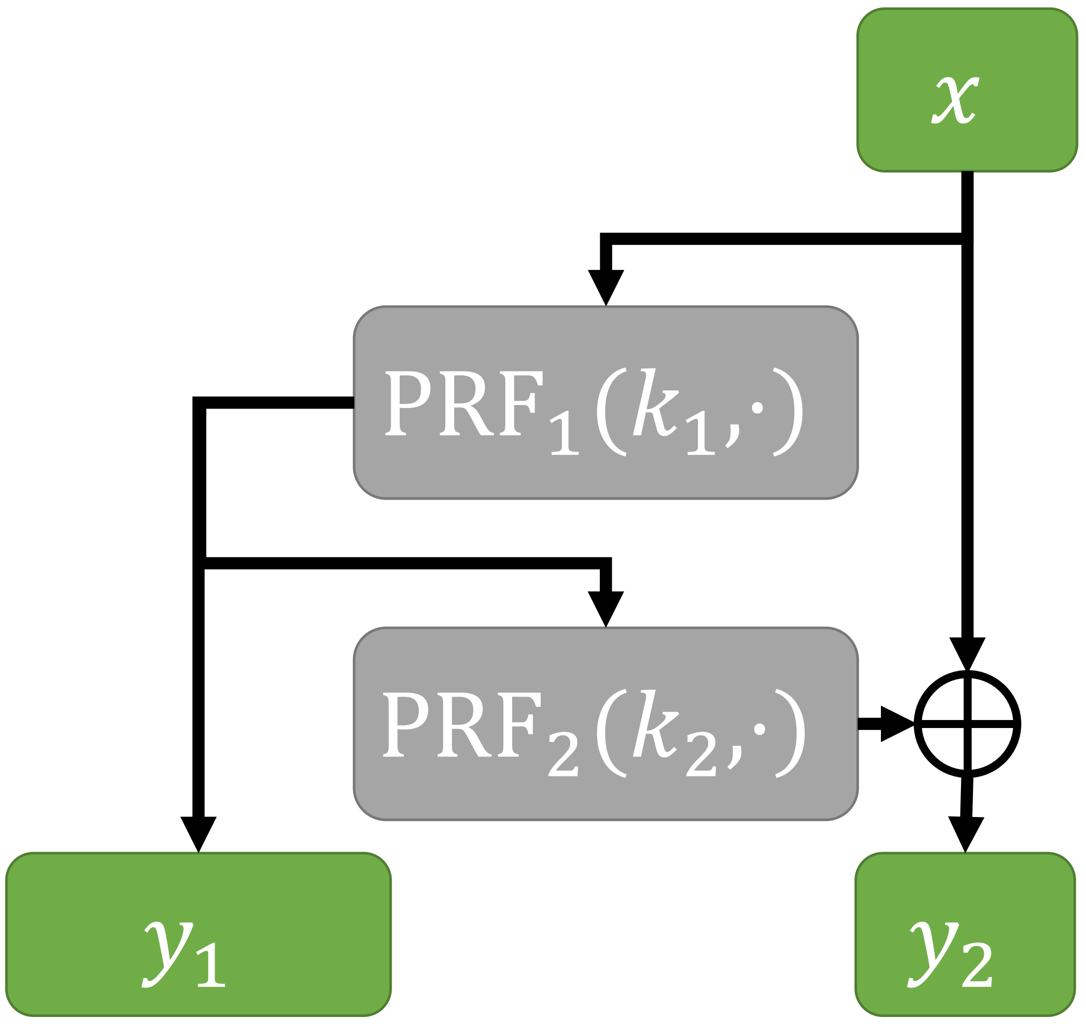


Master key: $k = (k_1, k_2)$

Punctured key (punctured at x^*):

- k_1 punctured at x^*

A Puncturable IPF

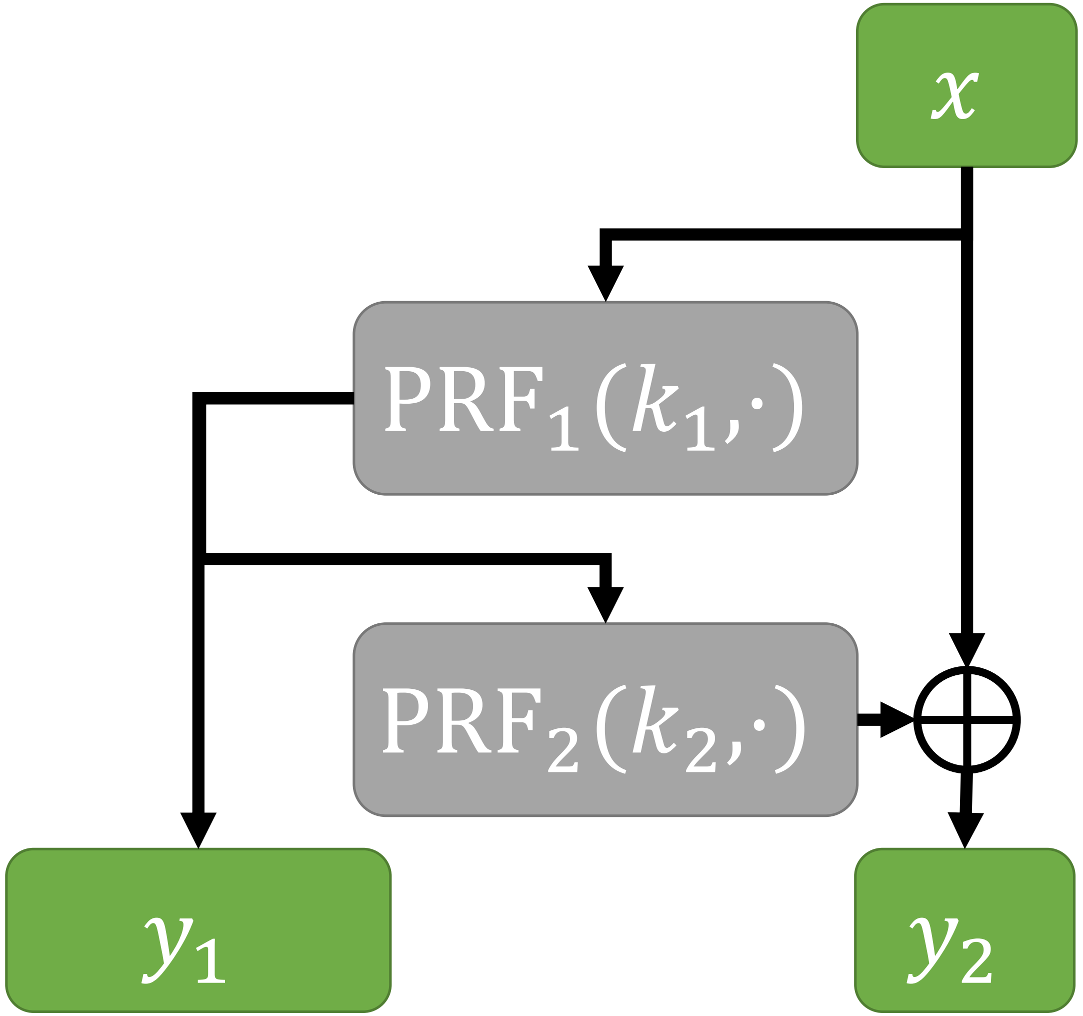


Master key: $k = (k_1, k_2)$

Punctured key (punctured at x^*):

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

A Puncturable IPF



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

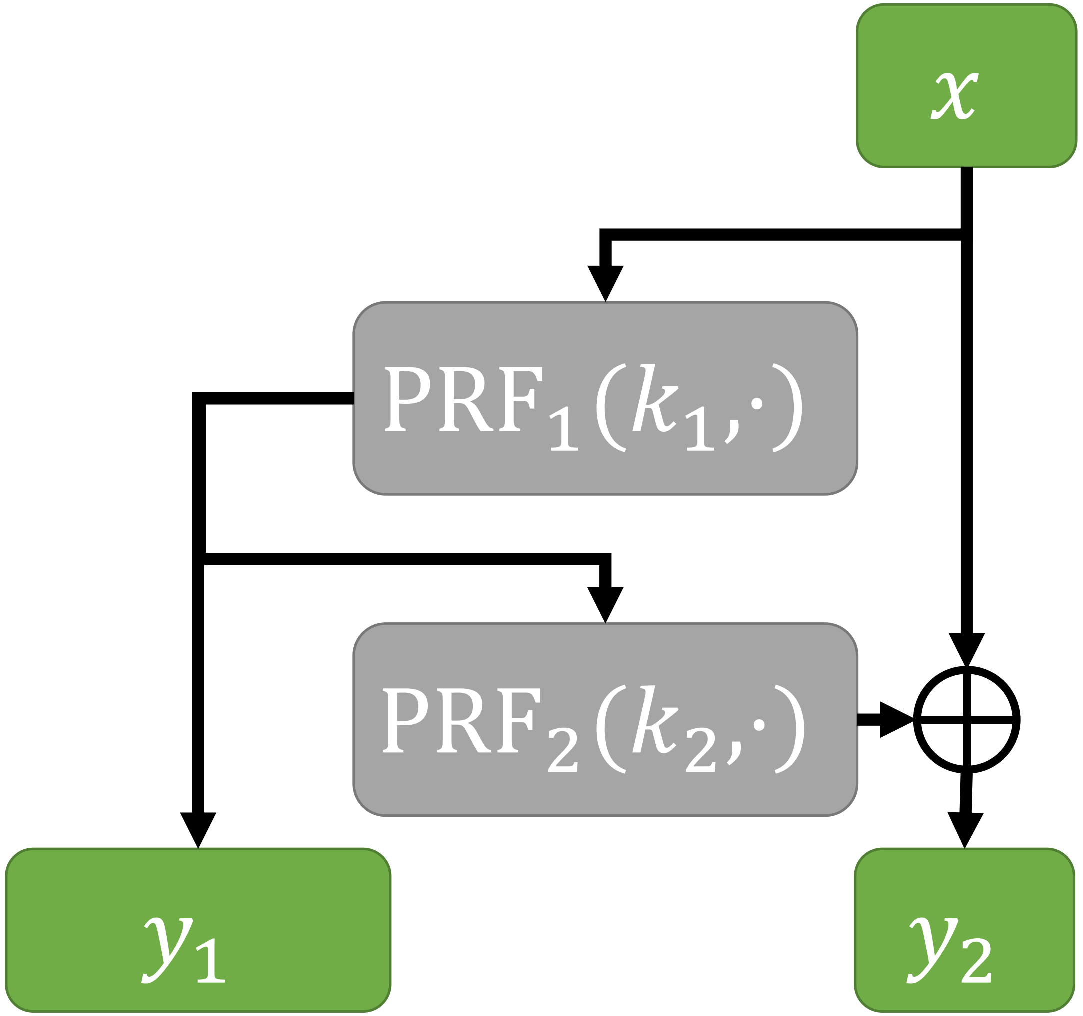
Punctured key (punctured at x^*):

- 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^*):

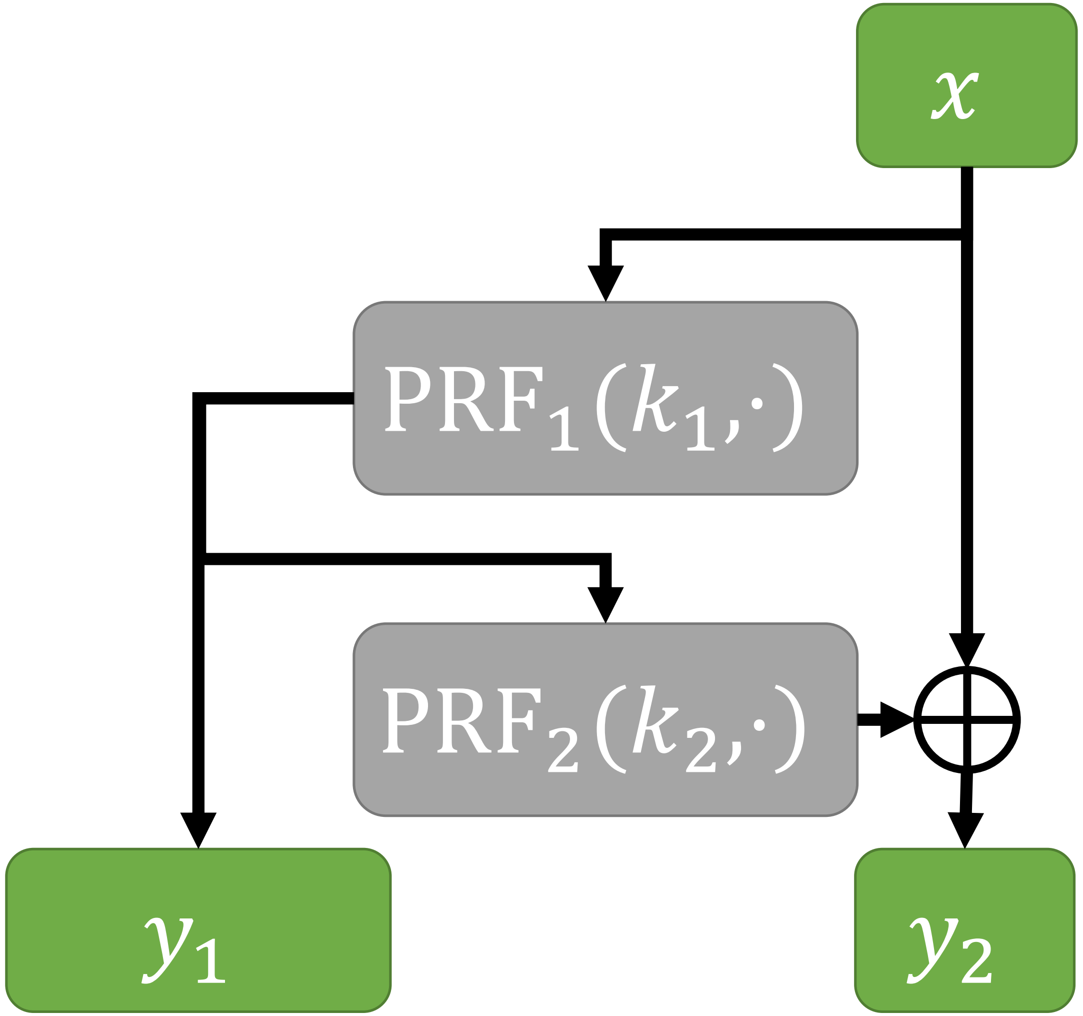
- k_1 punctured at x^*
- k_2 *privately* punctured at $\mathrm{PRF}_1(k_1, x^*)$

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

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

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

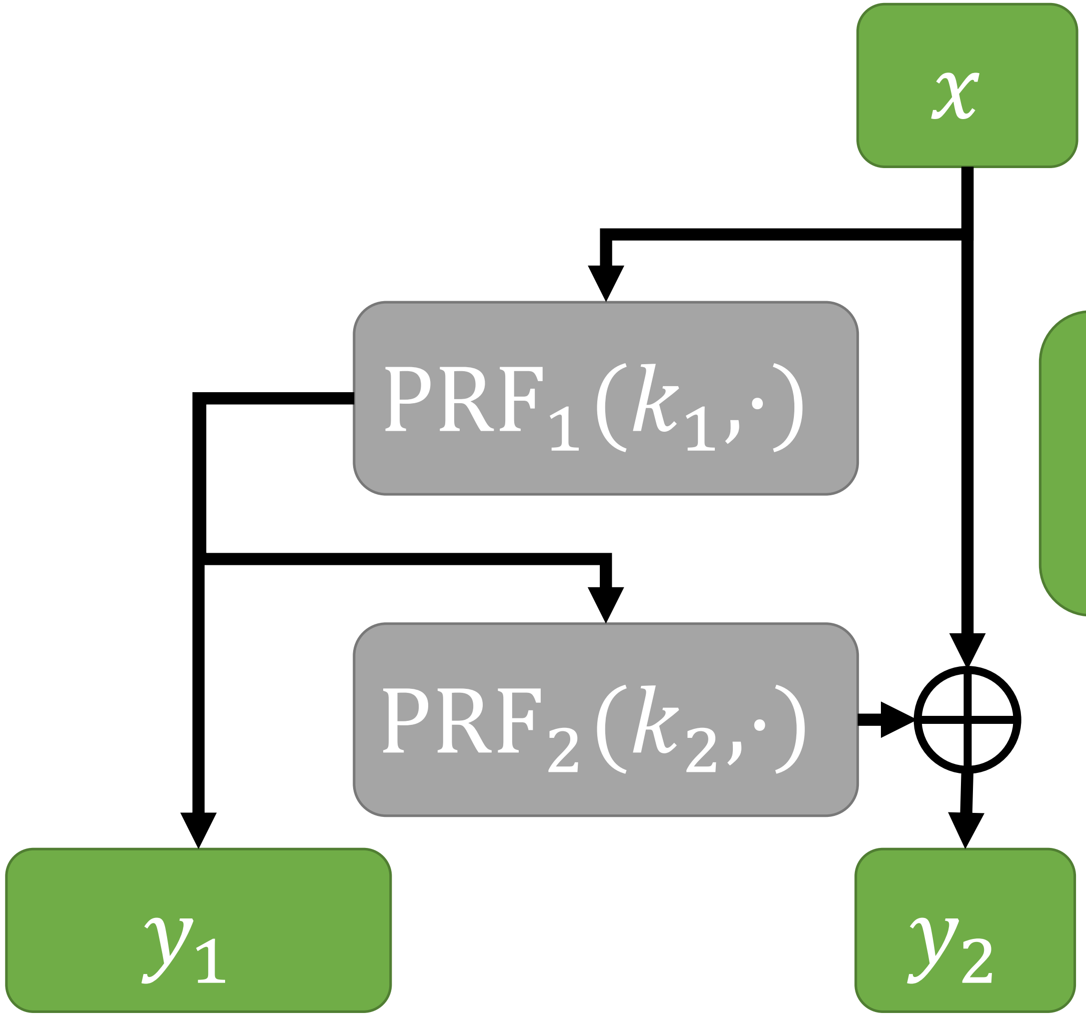
Hides y_1^*

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

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

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^*
- punctured at

Indistinguishable from uniform by constrained security of PRF_1

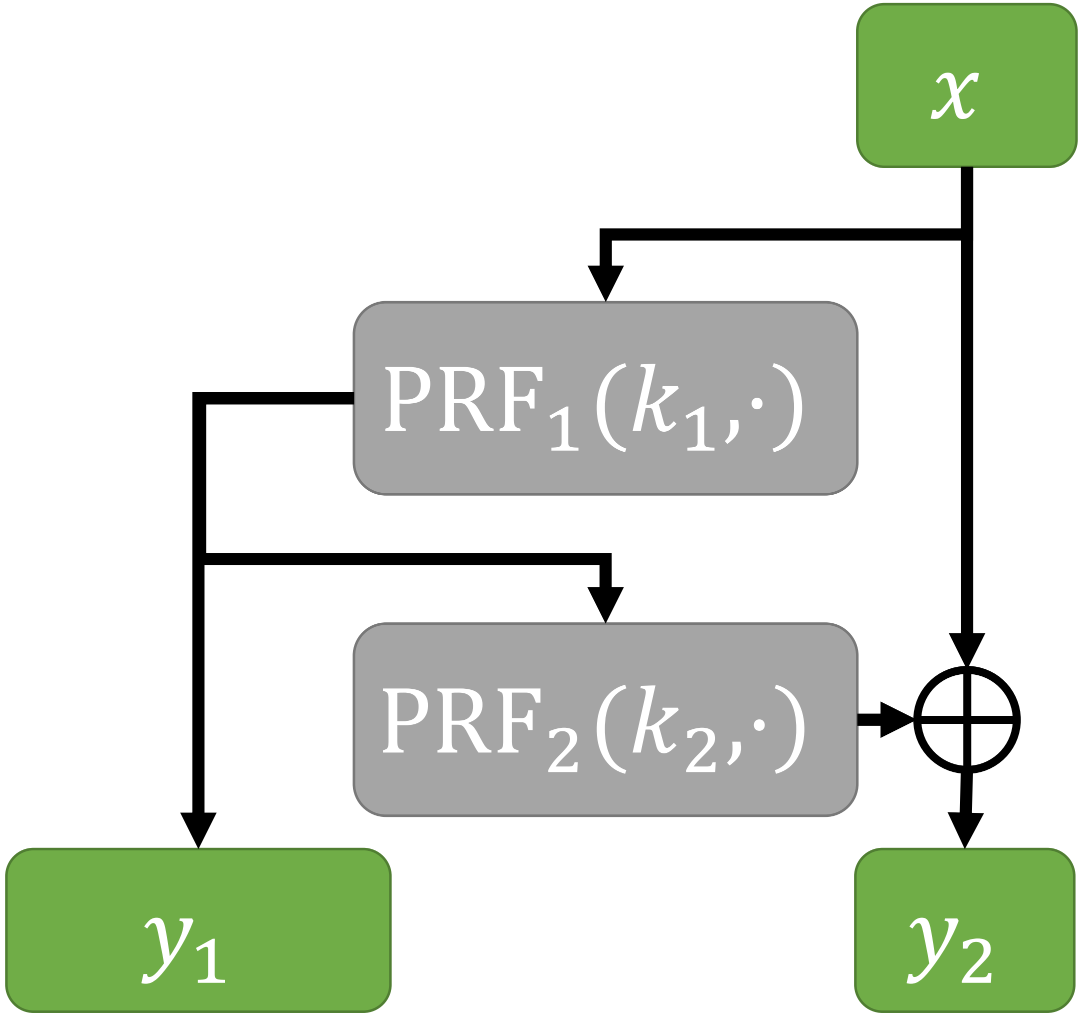
Hides y_1^*

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

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

Indistinguishable from uniform by constrained security of PRF_2

A Puncturable IPF



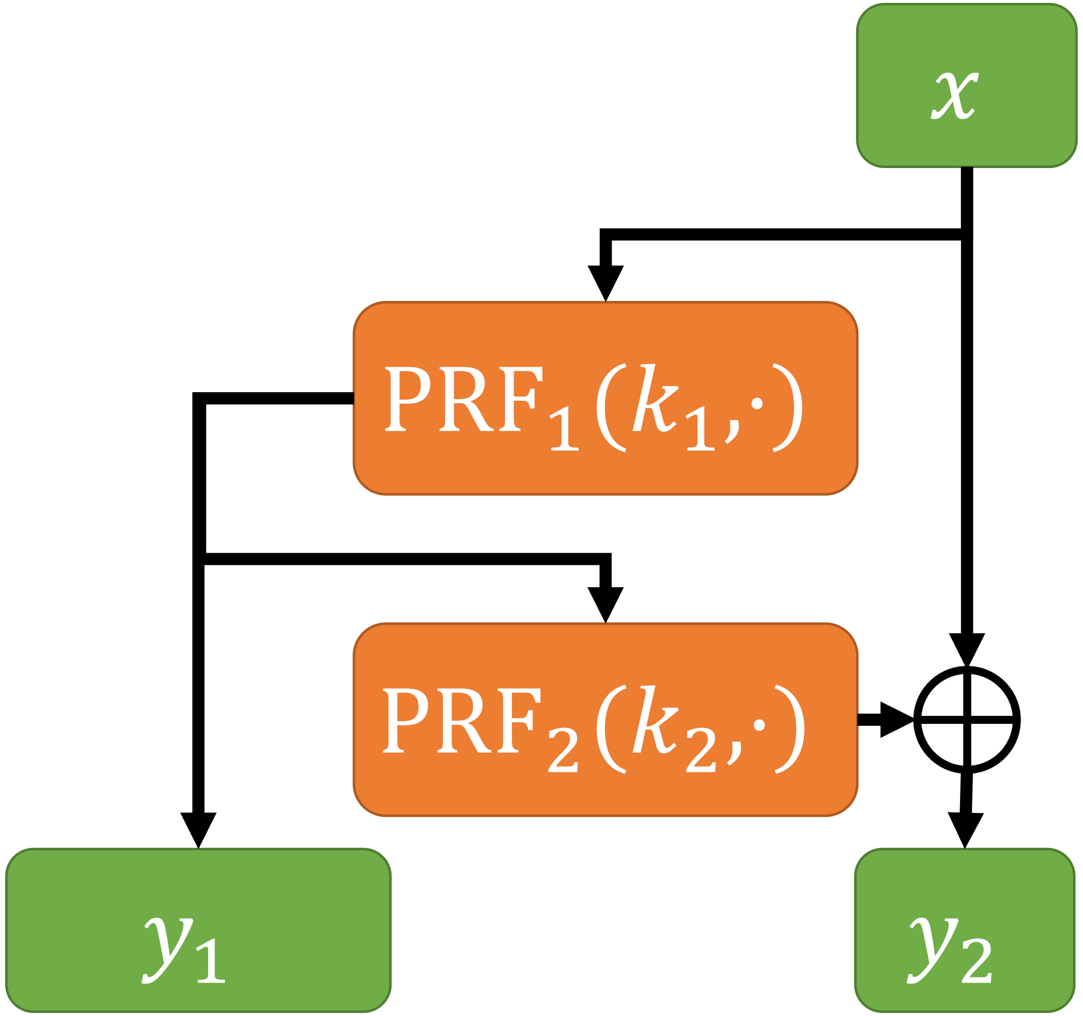
Master key: $k = (k_1, k_2)$

Punctured key (punctured at x^*):

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

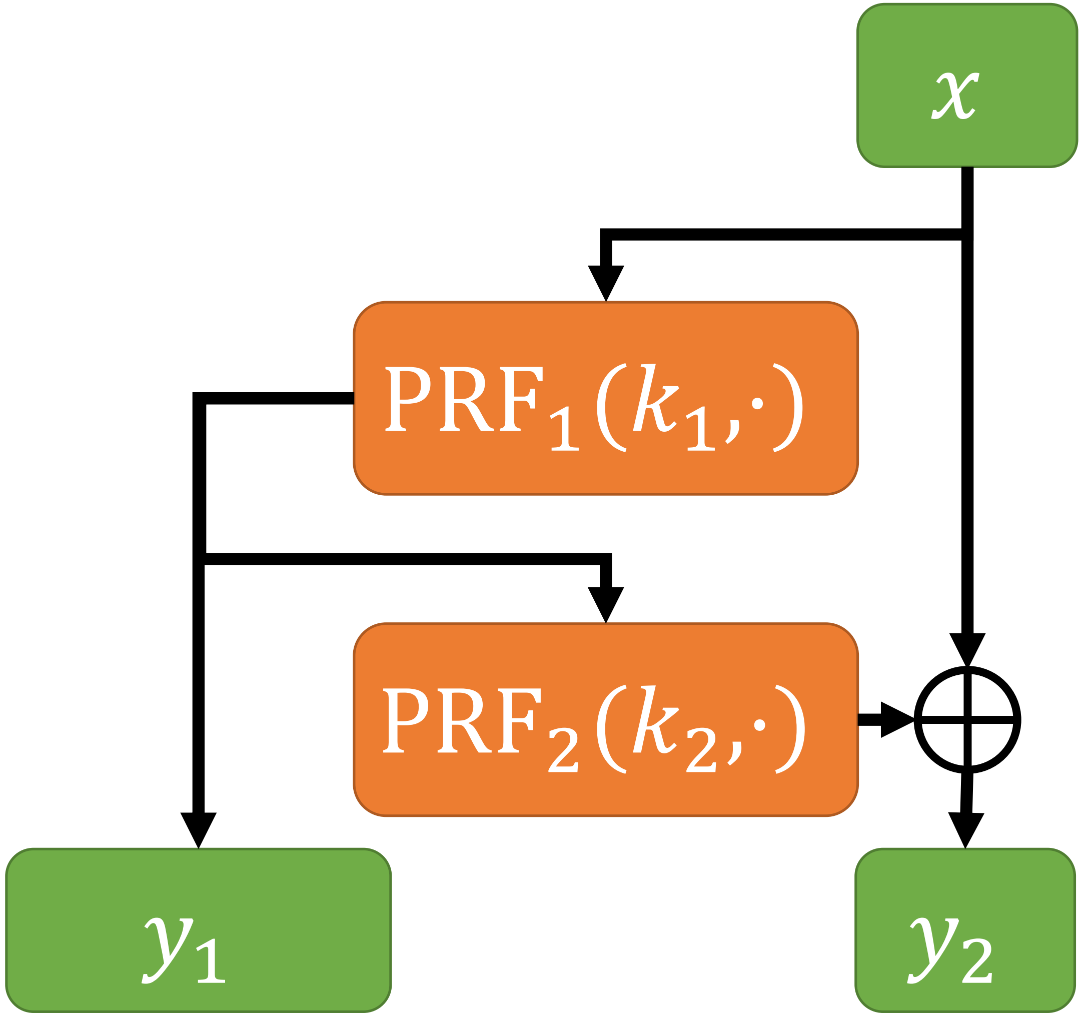
Can be instantiated from standard lattice assumptions [BKM17, CC17, BTVW17]

Circuit-Constrained IPFs



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

Circuit-Constrained IPFs

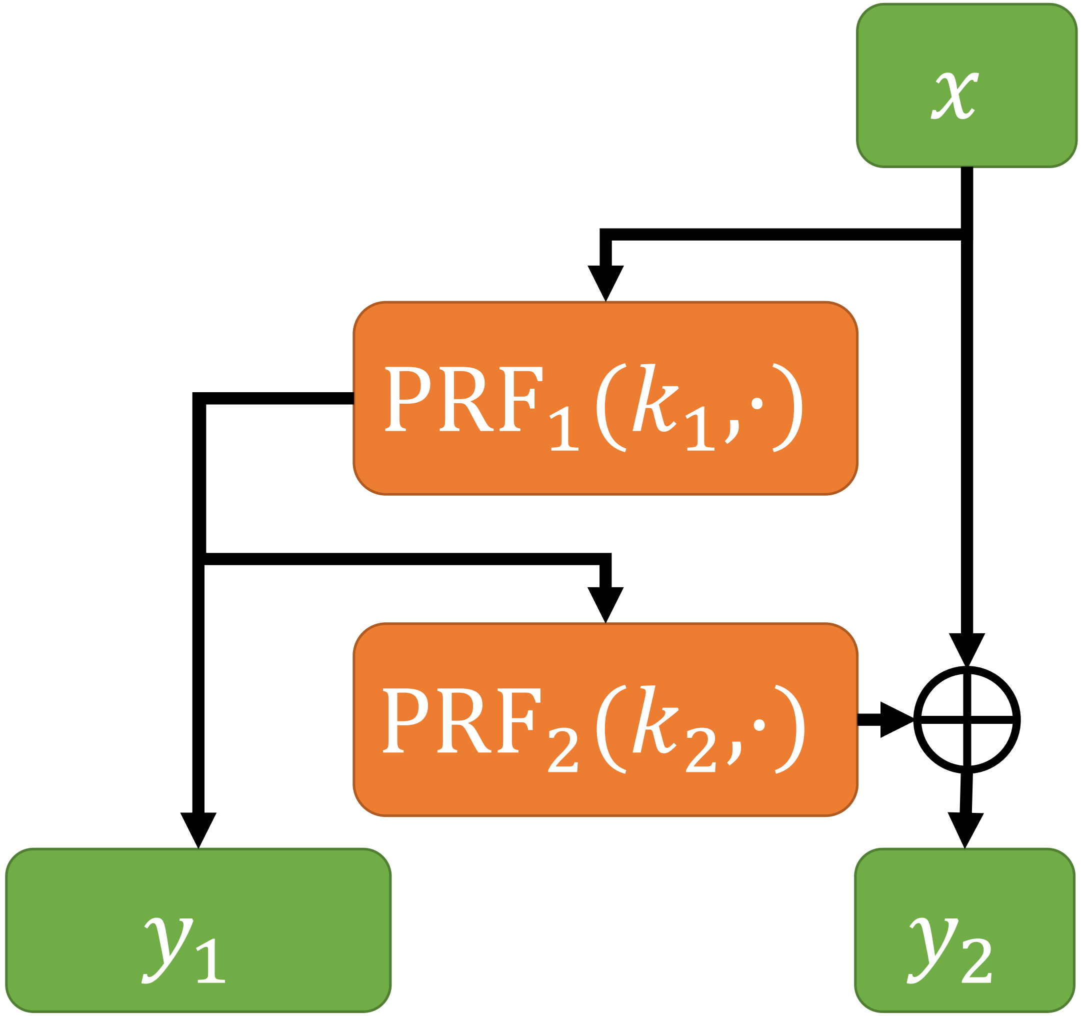


Master key: $k = (k_1, k_2)$

For puncturing at x^*:

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

Circuit-Constrained IPFs



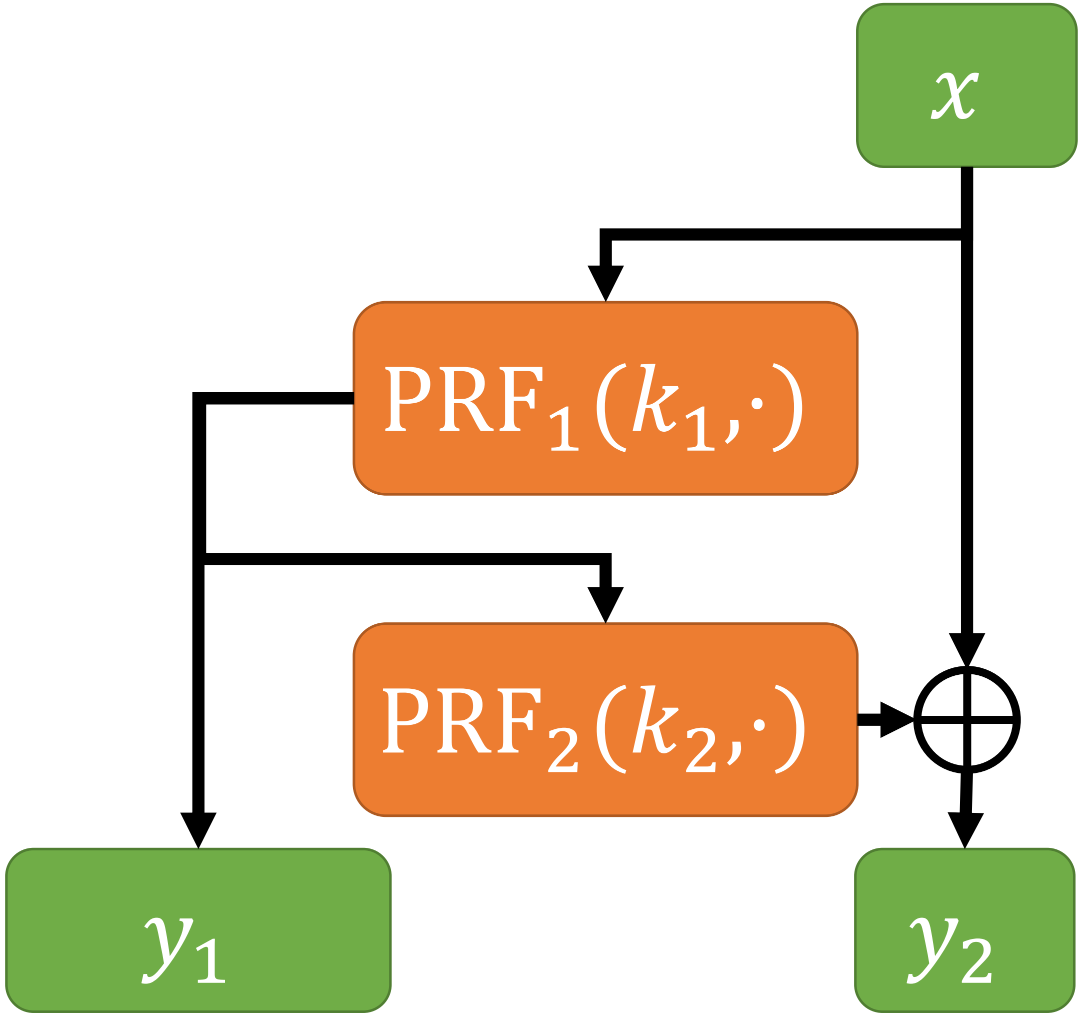
Master key: $k = (k_1, k_2)$

For puncturing at x^*:

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

To constrain to a circuit C:

Circuit-Constrained IPFs



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

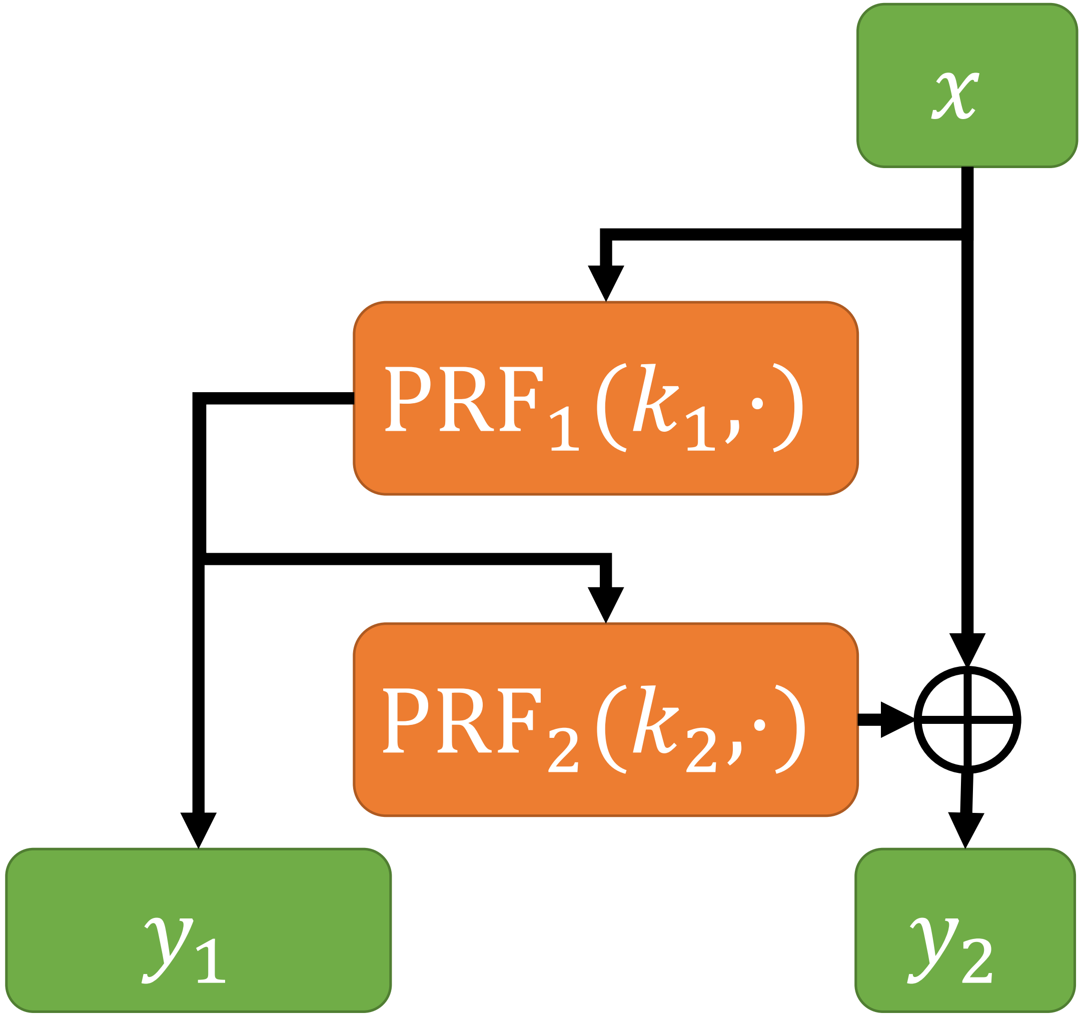
For puncturing at x^*:

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

To constrain to a circuit C:

- Constrain k_1 to C

Circuit-Constrained IPFs



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

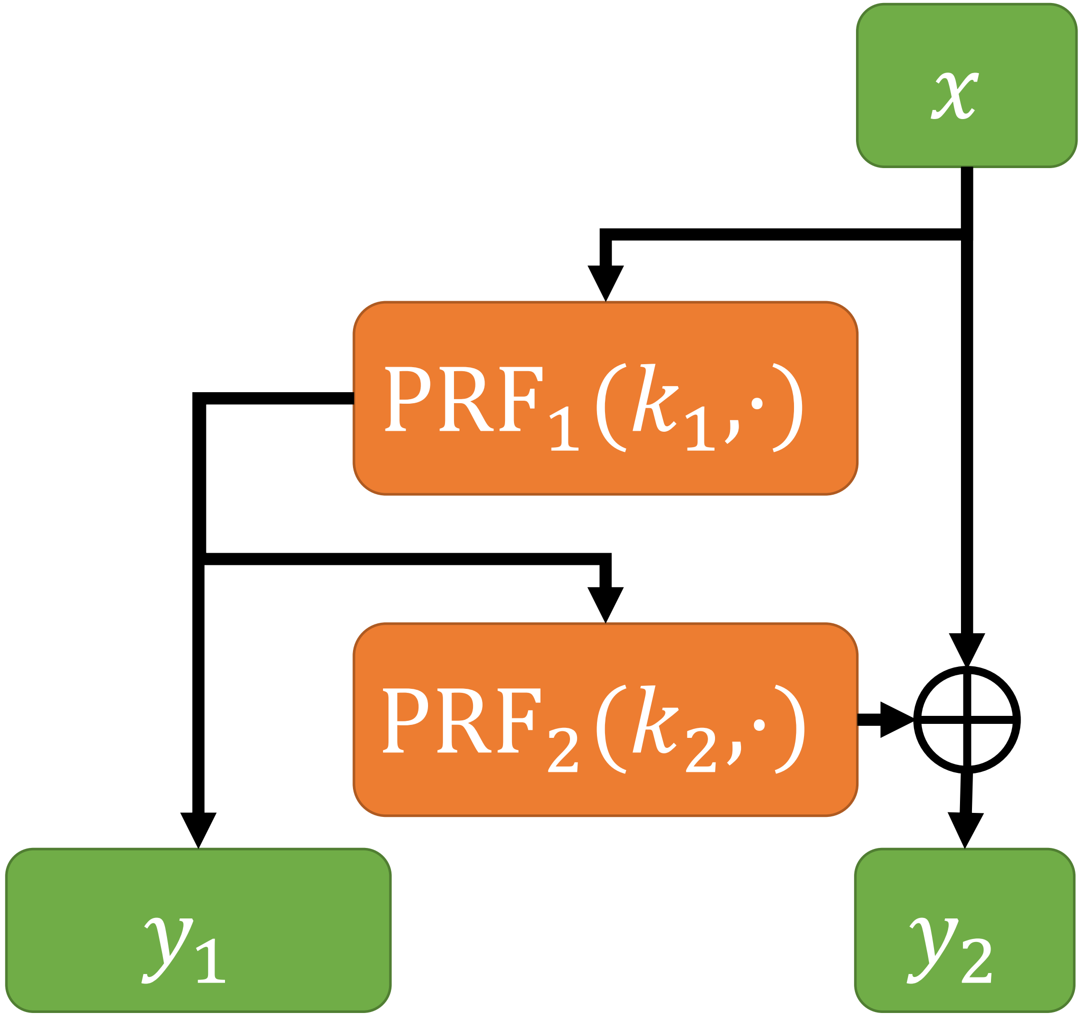
For puncturing at x^*:

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

To constrain to a 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^*
-

To co

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

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

Circuit-Constrained IPFs

See paper for construction

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)

Conclusions

Conclusions

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

Conclusions

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

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

Conclusions

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

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

Open Problems

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

Open Problems

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

Can we build puncturable IPFs from weaker assumptions?

Open Problems

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

Can we build puncturable IPFs from weaker assumptions?

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

Open Problems

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

Can we build puncturable IPFs from weaker assumptions?

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

Thank you!

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