Watermarking Cryptographic Functionalities from Standard Lattice Assumptions

Sam Kim and David J. Wu

Stanford University

Digital Watermarking

Often used to identify owner of content and prevent unauthorized distribution

Digital Watermarking

- Content is (mostly) viewable

Digital Watermarking

- Content is (mostly) viewable
- Watermark difficult to remove (without destroying the image)

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

Embed a "mark" within a program

If mark is removed, then program is corrupted

Three algorithms:

- $\text{Setup}(1^\lambda) \to \text{wsk}$: Samples the watermarking <u>secret</u> key wsk
- $\text{Mark}(\text{wsk}, C) \to C'$: Takes a circuit C and outputs a marked circuit C'
- $\text{Verify}(\text{wsk}, C') \to \{0,1\}$: Tests whether a circuit C' is marked or not

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

Extends to setting where watermark can be an (arbitrary) string:

- $\mathsf{Mark}(\mathrm{wsk}, C, m) \to C'$: Takes a circuit C and a message m and outputs a marked circuit C'
- $\mathsf{Verify}(\mathrm{wsk}, C') \to m$: Takes a circuit C' and outputs a message m (or $\perp$ if the circuit is unmarked)

[See paper for full details]

Er

Three

- $\mathsf{Setup}(1^\lambda)$ … wsk: Samples the watermarking secret key wsk
- $\mathsf{Mark}(\mathrm{wsk}, C) \to C'$: Takes a circuit C and outputs a marked circuit C'
- $\mathsf{Verify}(\mathrm{wsk}, C') \to \{0,1\}$: Tests whether a circuit C' is marked or not

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

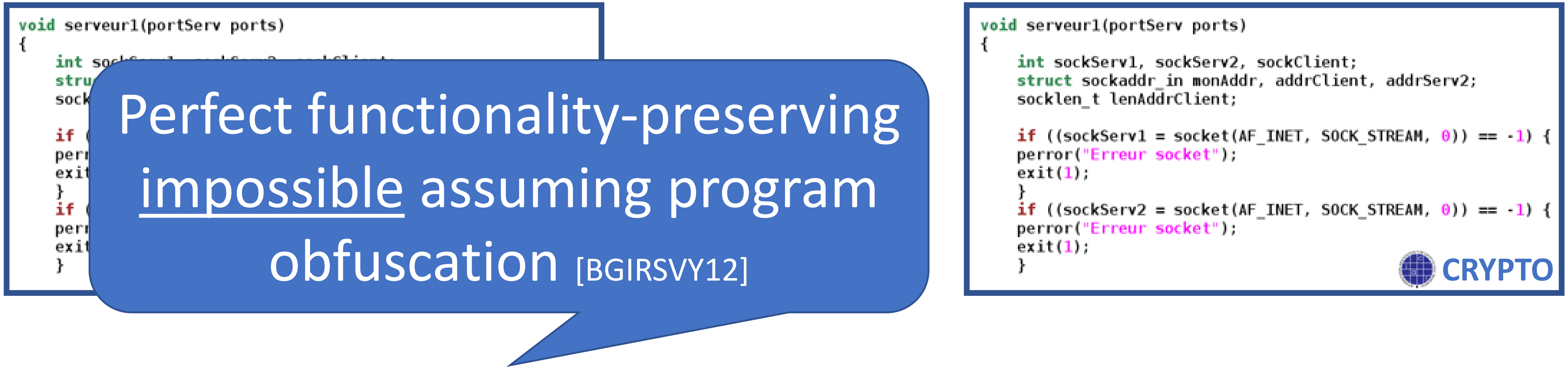
Functionality-preserving: On input a program (modeled as a Boolean circuit C), the Mark algorithm outputs a circuit C' where

$$C(x) = C'(x)$$

on all but a negligible fraction of inputs x

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]



Functionality-preserving: On input a program (modeled as a Boolean circuit C), the Mark algorithm outputs a circuit C' where

$$C(x) = C'(x)$$

on all but a negligible fraction of inputs x

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

Unremovability: Given a marked circuit $C^\star$, no efficient adversary can construct a circuit C' where

- $C'(x) = C^\star(x)$ on all but a negligible fraction of inputs x
- $\text{Verify}(\text{wsk}, C') = 0$

Watermarking Security Game [CHNVW16, BLW17]

$\mathsf{wsk} \leftarrow \mathsf{Setup}(1^\lambda)$

$C^\star \leftarrow \mathcal{C}$

C ←

$\mathsf{Mark}(\mathsf{wsk}, C)$ →

$\mathsf{Mark}(\mathsf{wsk}, C^\star)$ →

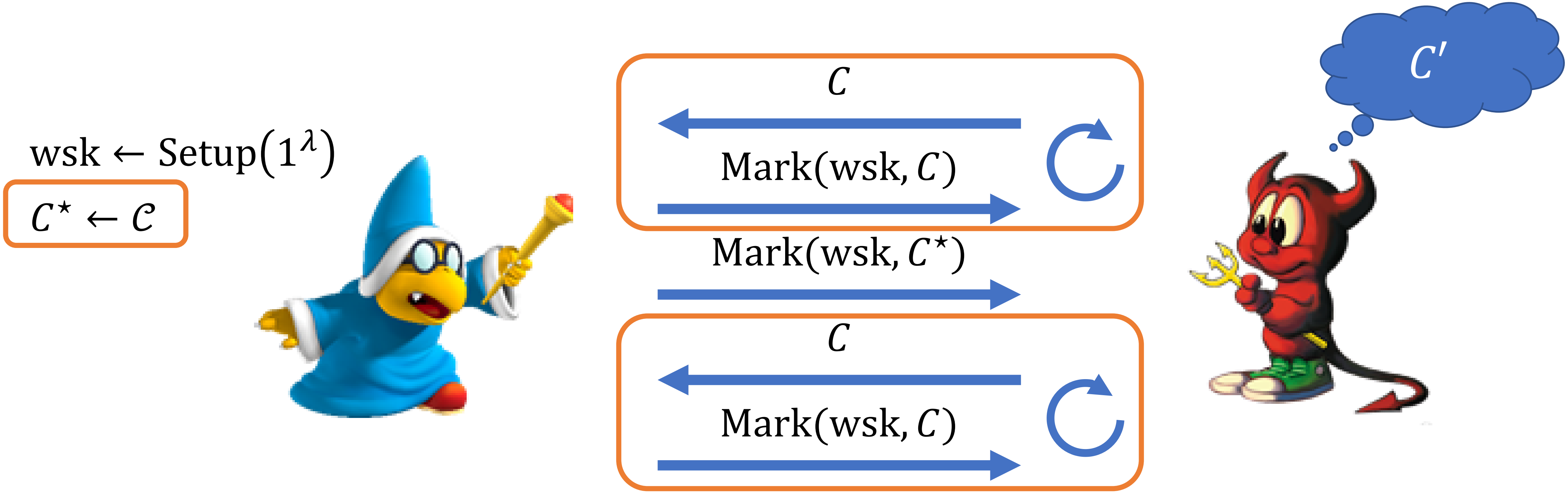
C ←

$\mathsf{Mark}(\mathsf{wsk}, C)$ →

Unremovability: Given a marked circuit $C^\star$, no efficient adversary can construct a circuit C' where

- $C'(x) = C^\star(x)$ on all but a negligible fraction of inputs x
- $\mathsf{Verify}(\mathsf{wsk}, C') = 0$

Watermarking Security Game [CHNVW16, BLW17]



- Adversary has access to marking oracle (sees marked programs of its choosing)
- Challenge circuit $C^\star$ sampled from the circuit family
- Adversary has <u>complete</u> flexibility in crafting C' (it just outputs a description of a circuit)

Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

Unforgeability: Given marked programs $C_1, \dots, C_\ell$, no efficient adversary can construct a circuit C' where

- For all $i \in [\ell]$, $C'(x) \neq C_i(x)$ on a <u>noticeable</u> fraction of inputs x
- $\mathrm{Verify}(\mathrm{wsk}, C') = 1$

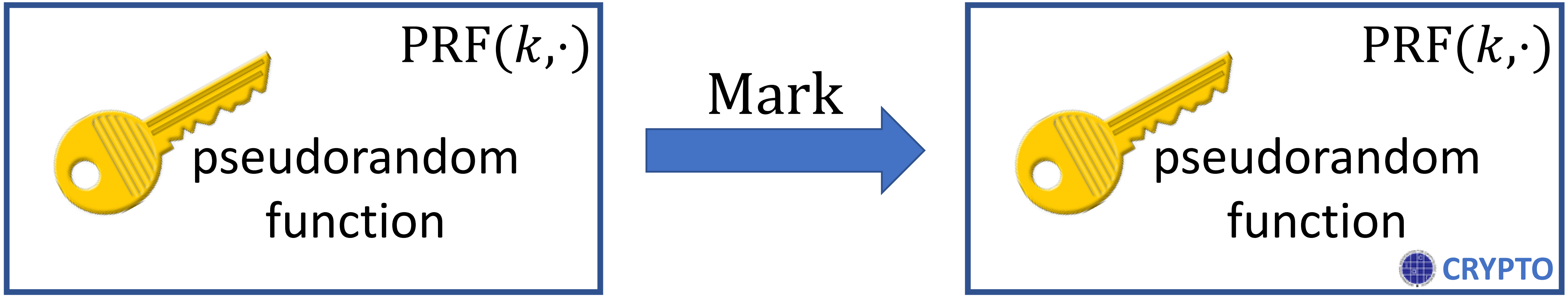
Watermarking Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]

- Notion only achievable for functions that are not learnable
- Focus has been on cryptographic functions

Watermarking Cryptographic Programs

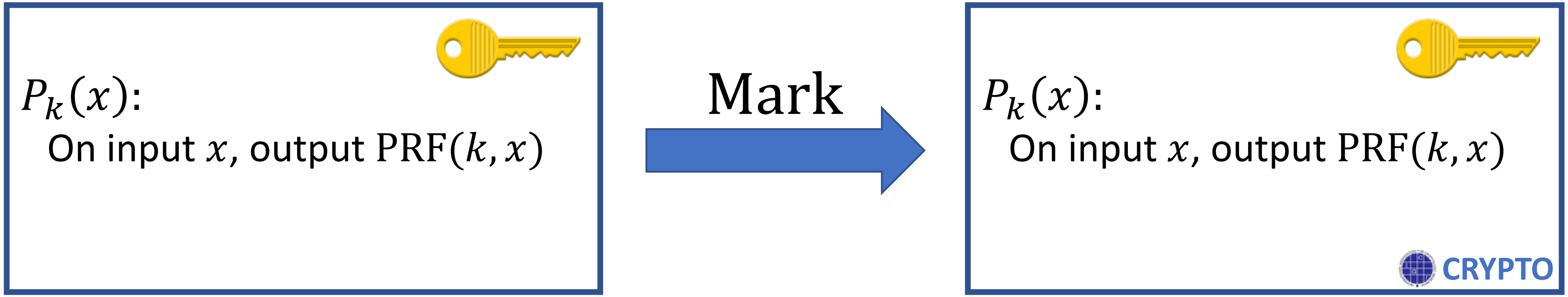
[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]



- Focus of this work: watermarking PRFs [CHNVW16, BLW17]

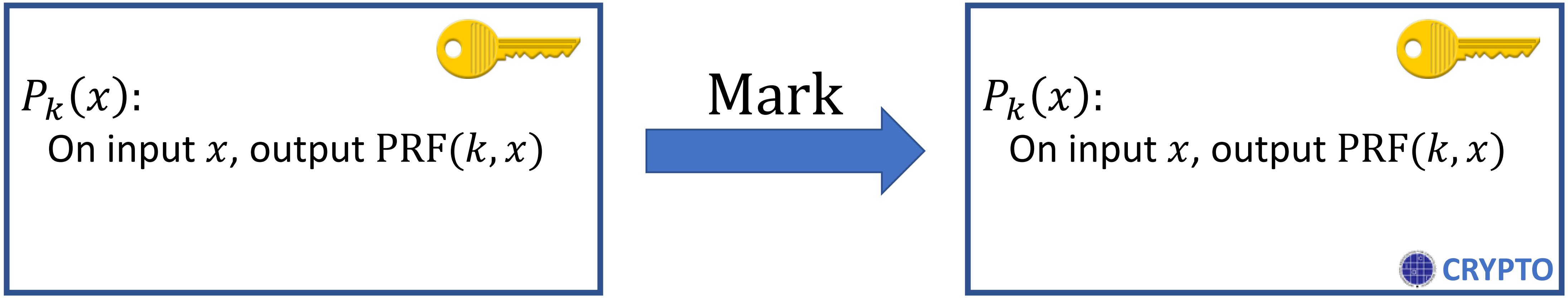
Watermarking Cryptographic Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17]



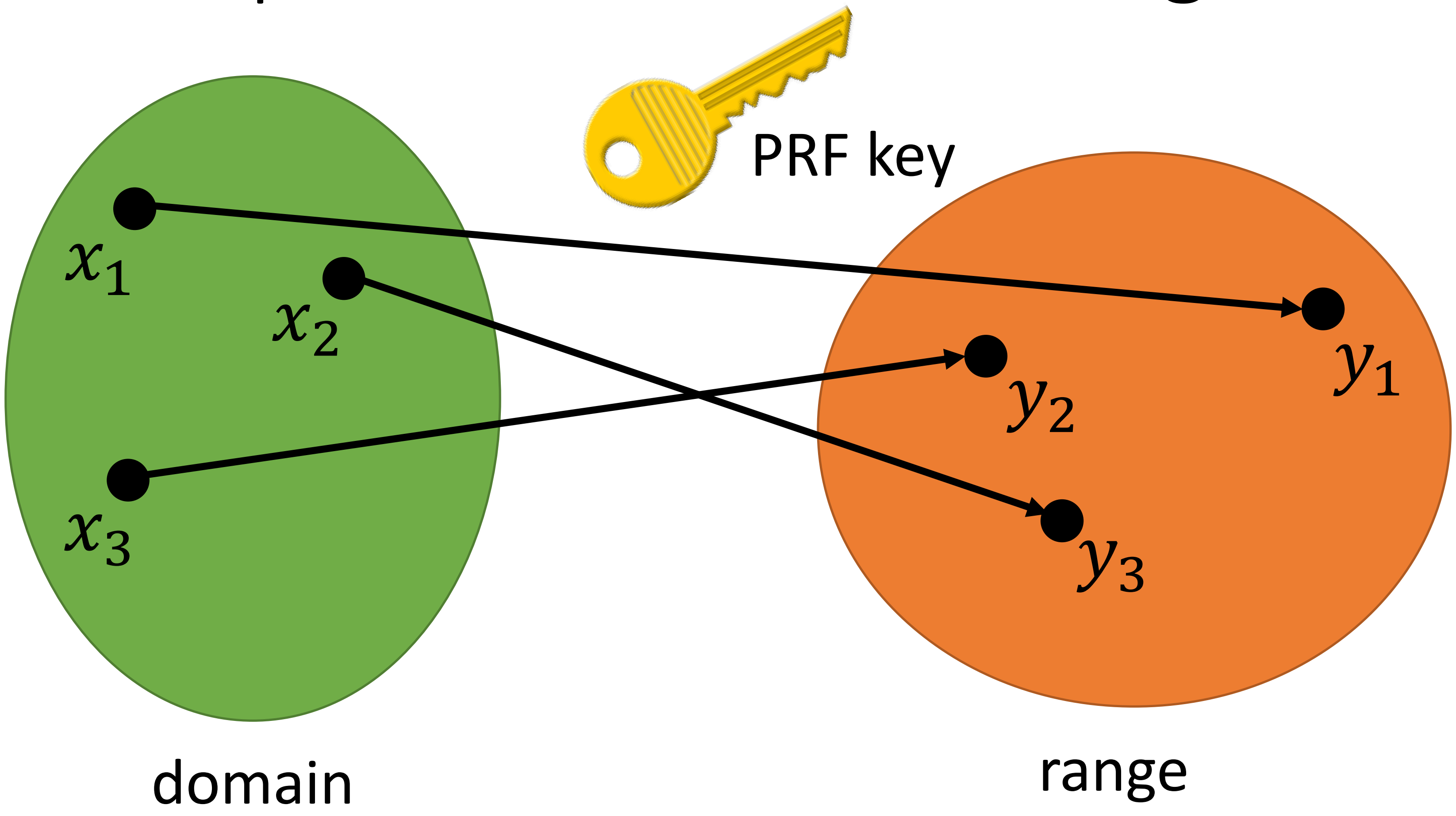
- Focus of this work: watermarking PRFs [CHNVW16, BLW17]
- Enables watermarking of symmetric primitives built from PRFs (e.g., encryption, MACs, etc.)

Main Result



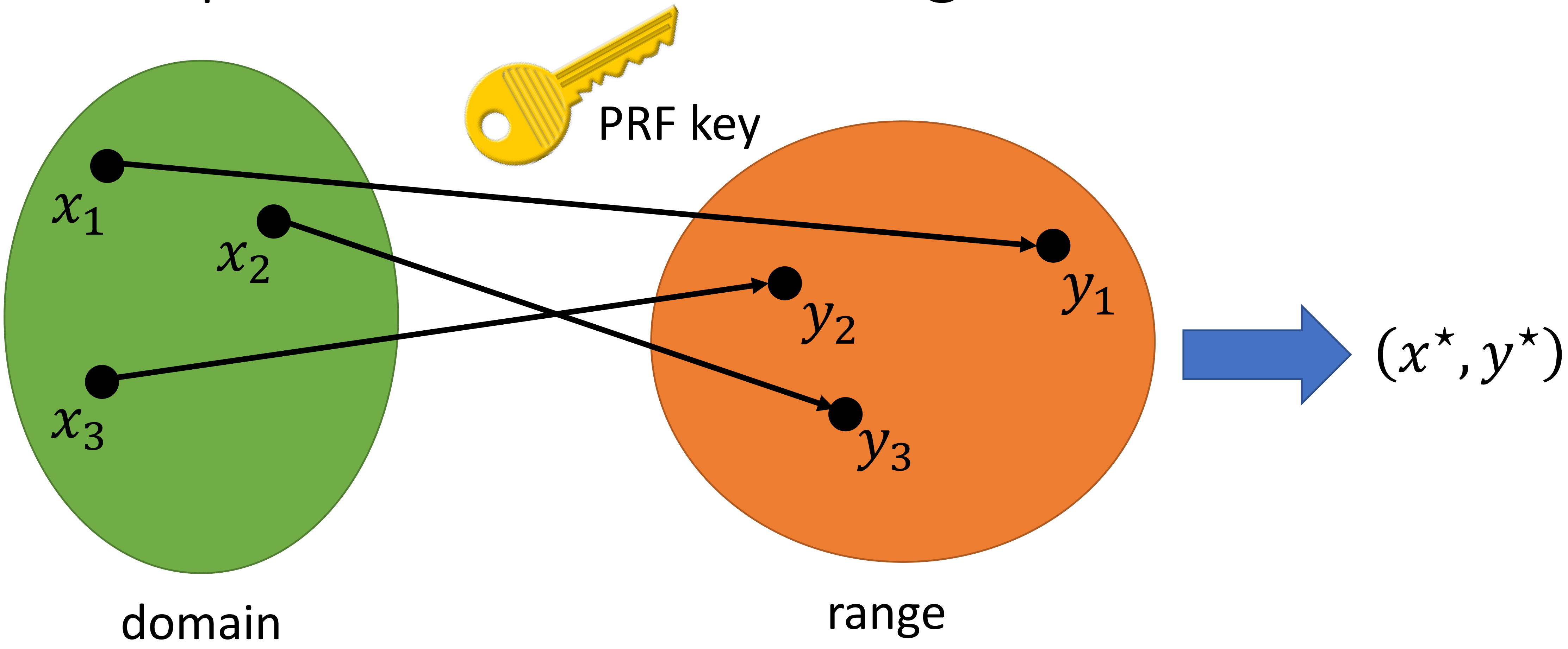
This work: Under *standard lattice assumptions*, there exists a secretly-verifiable watermarkable family of PRFs

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



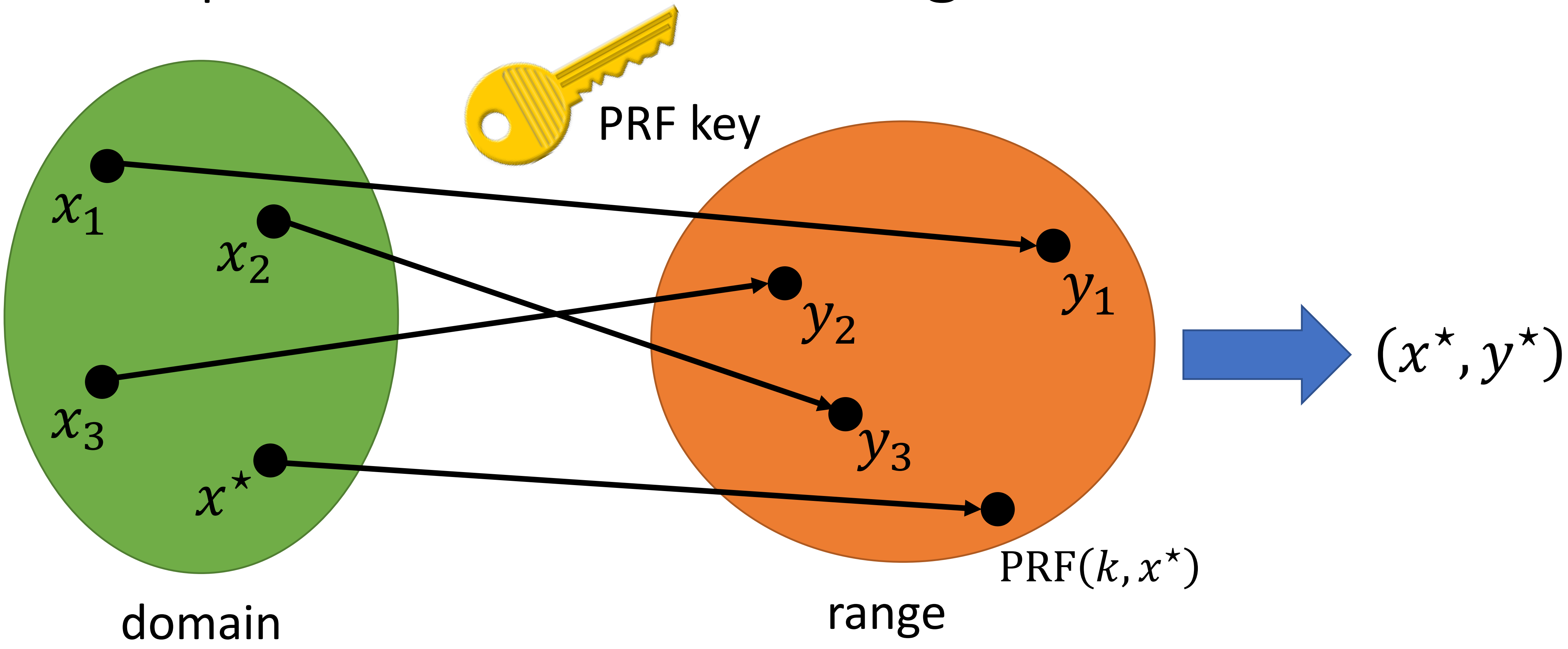
Step 1: Evaluate PRF on test points x_1, x_2, x_3 (part of the watermarking secret key)

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



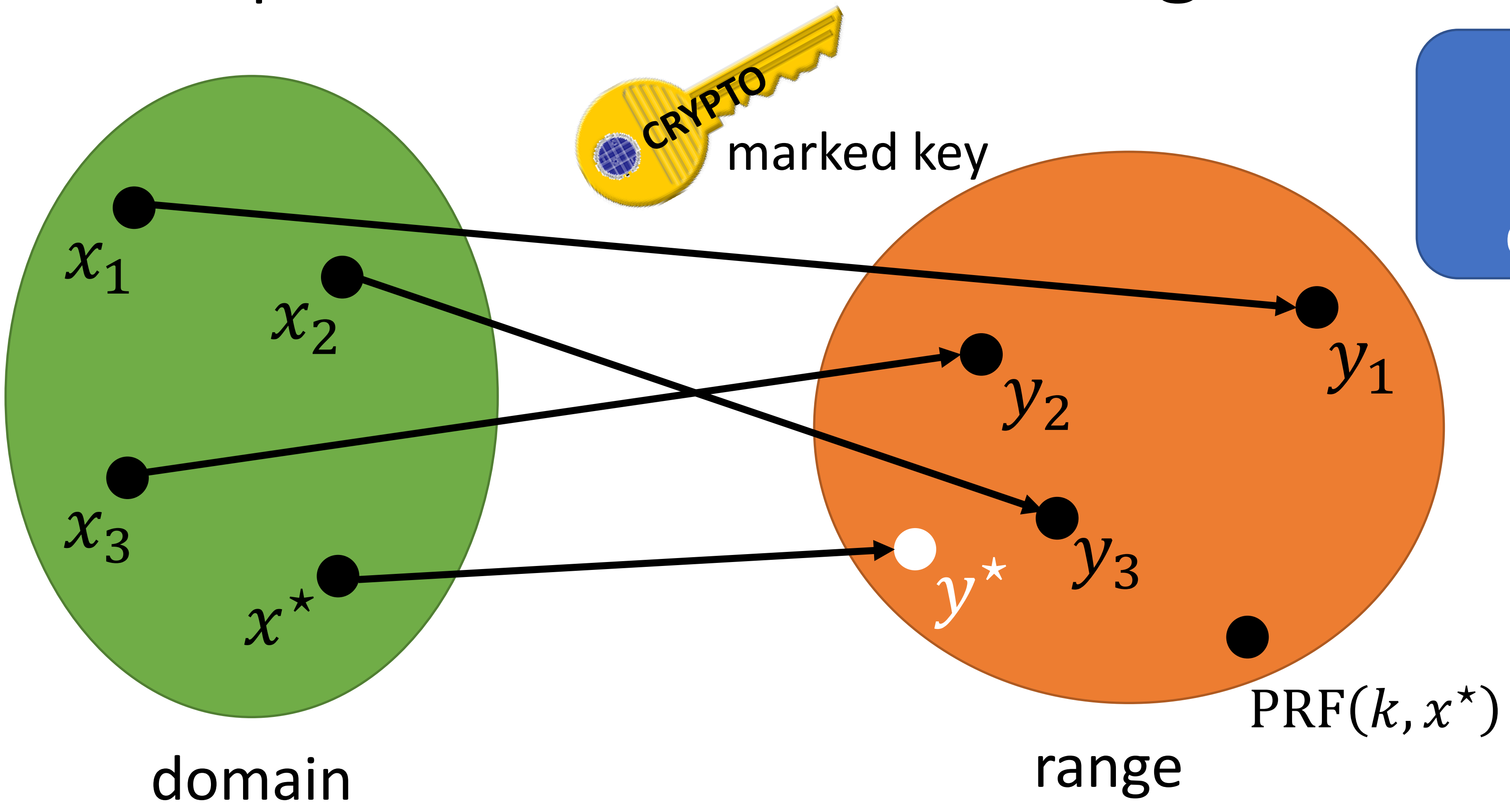
Step 2: Derive a pair $(x^\star, y^\star)$ from y_1, y_2, y_3

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Step 3: "Marked key" is a circuit that implements the PRF at all points, except at $x^\star$, the output is changed to $y^\star$

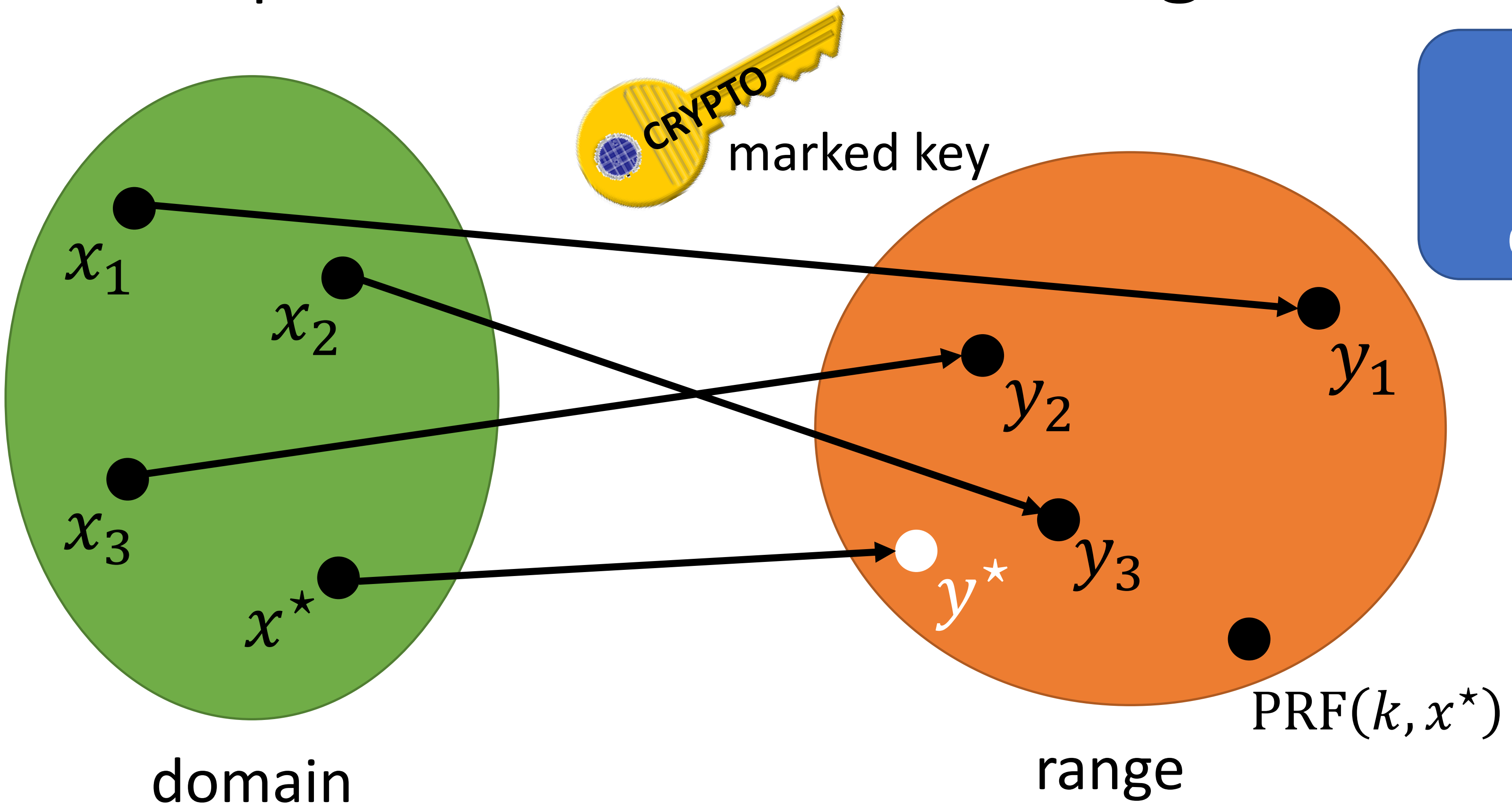
Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Defer implementation details for now…

Step 3: "Marked key" is a circuit that implements the PRF at all points, except at $x^\star$, the output is changed to $y^\star$

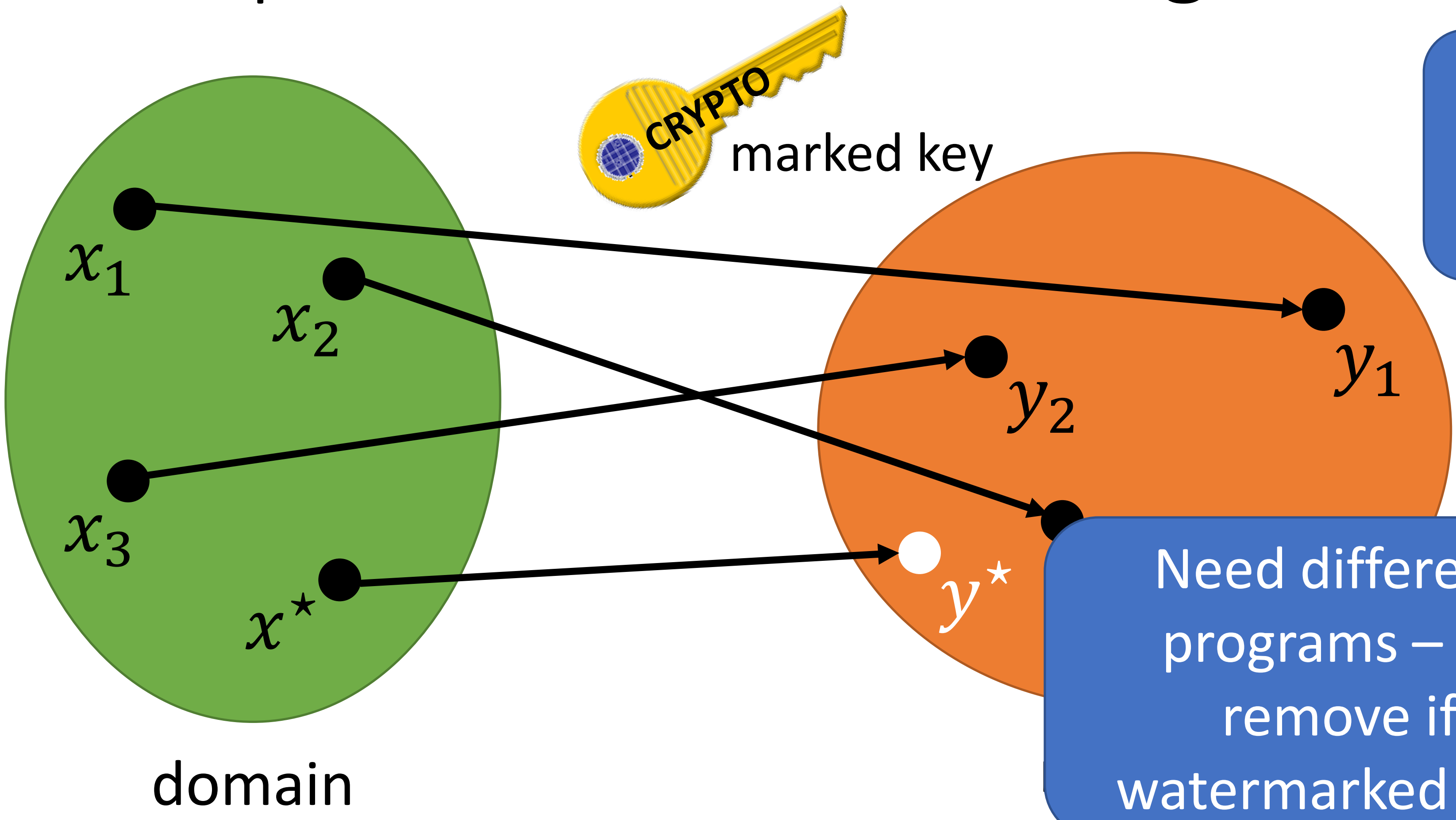
Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Defer implementation details for now…

Verification: Evaluate function at x_1, x_2, x_3, derive $(x^\star, y^\star)$ and check if the value at $x^\star$ matches $y^\star$

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

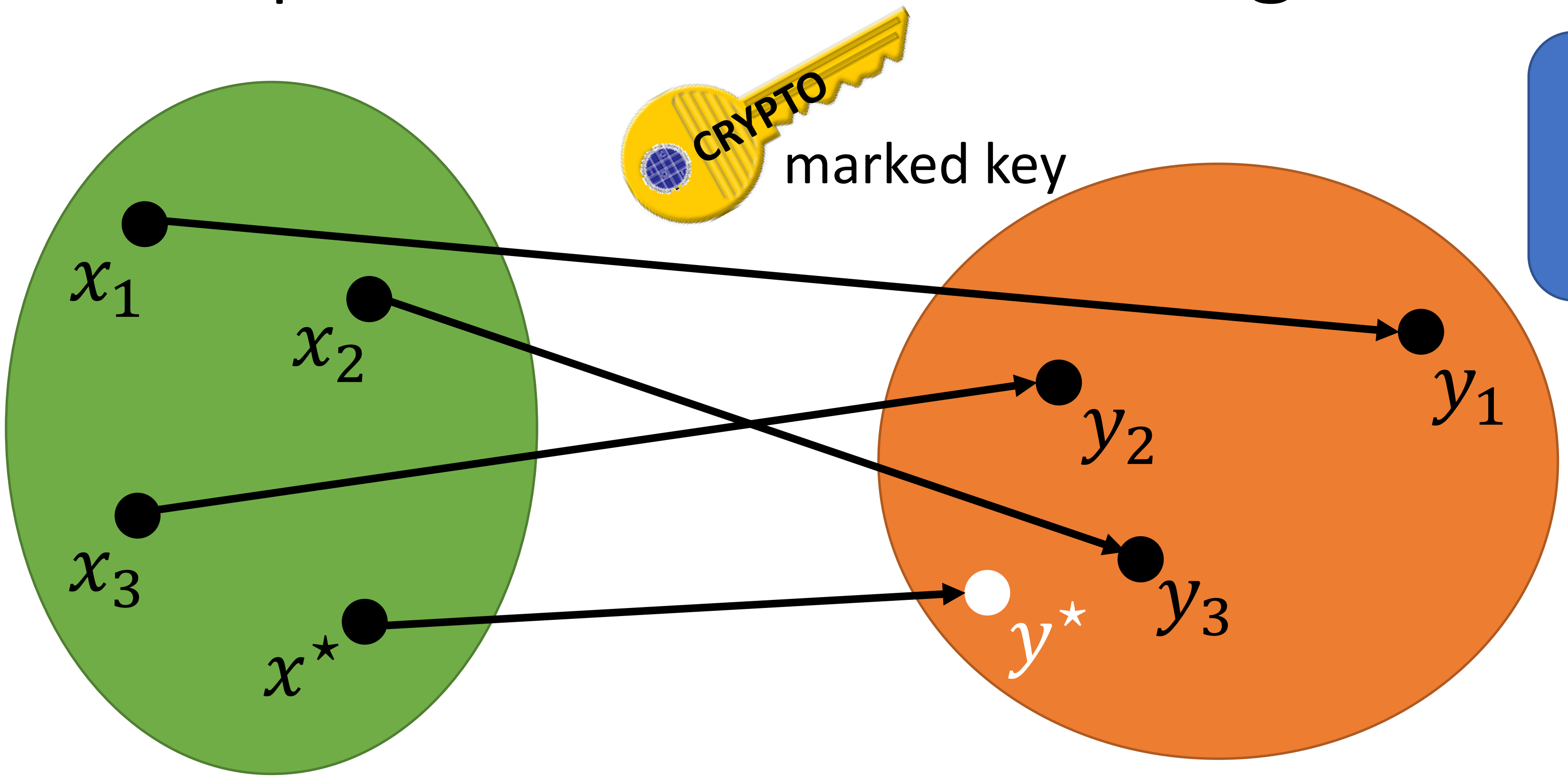


Defer implementation details for now…

Need different $x^\star$ for different programs – otherwise easy to remove if adversary sees watermarked keys of its choosing

Verification: Evaluate function at x_1, x_2, x_3, derive $(x^\star, y^\star)$ and check if the value at $x^\star$ matches $y^\star$

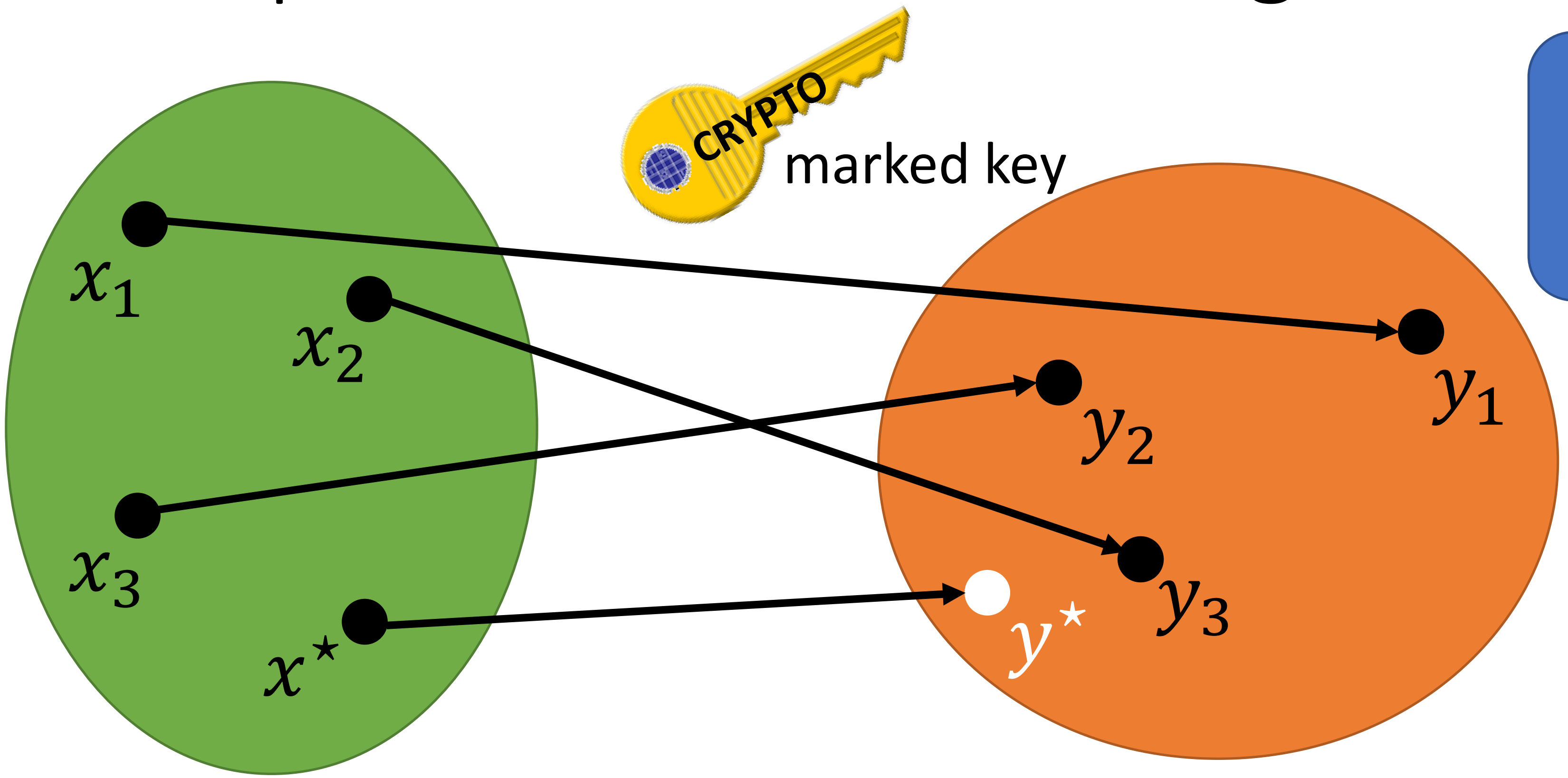
Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Defer implementation details for now…

☑ Functionality-preserving: function differs at a single point

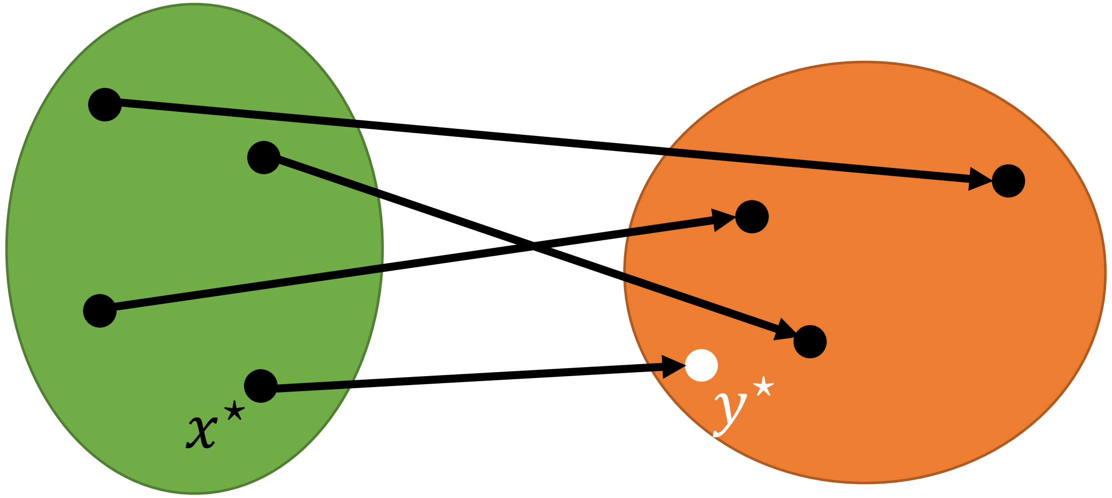
Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Defer implementation details for now…

- ☑ Functionality-preserving: function differs at a single point
- ☑ Unremovable: as long as adversary cannot tell that $(x^\star, y^\star)$ is “special”

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

How to implement this functionality?

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

$P_{(x^\star, y^\star)}(x)$:
- if $x = x^\star$, output $y^\star$
- else, output $\mathrm{PRF}(k, x)$

Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

Obfuscated program has PRF key embedded inside and outputs $\mathrm{PRF}(k, x)$ on all inputs $x \neq x^\star$ and $y^\star$ when $x = x^\star$

How to implement this functionality?

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

$P_{(x^\star, y^\star)}(x)$:
- if $x = x^\star$, output $y^\star$
- else, output $\mathrm{PRF}(k, x)$

Essentially relies on secretly *re-programming* the value at $x^\star$

Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

Obfuscated program has PRF key embedded inside and outputs $\mathrm{PRF}(k, x)$ on all inputs $x \neq x^\star$ and $y^\star$ when $x = x^\star$

functionality?

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

$P_{(x^\star, y^\star)}(x)$:

- if $x = x^\star$, output $y^\star$
- else, output $\text{PRF}(k, x)$

Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

Obfuscated program has PRF key embedded inside and outputs $\text{PRF}(k, x)$ on all inputs $x \neq x^\star$ and $y^\star$ when $x = x^\star$

Key technical challenge: How to hide $(x^\star, y^\star)$ within the watermarked key (without obfuscation)?

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

$P_{(x^\star, y^\star)}(x)$:

- if $x = x^\star$, output $y^\star$
- else, output PRF$(k, x$)

Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

RF key

its

$x^\star$

Has an obfuscation flavor: need to embed a secret inside a piece of code that cannot be removed

Key technical challenge: How to hide $(x^\star, y^\star)$ within the watermarked key (without obfuscation)?

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

$P_{(x^\star, y^\star)}(x)$:

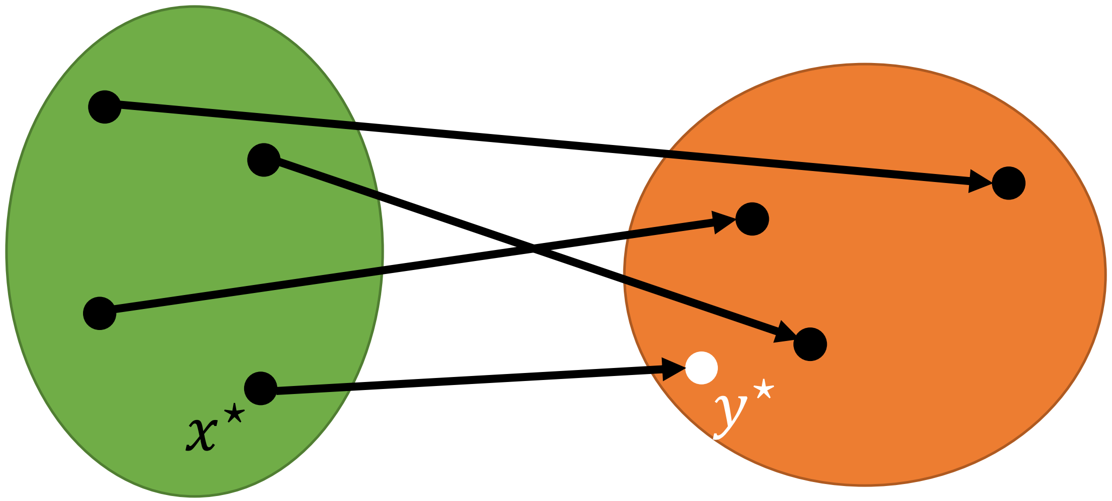
- if $x = x^\star$, output $y^\star$
- else, output $\mathrm{PRF}(k, x)$

Prior solutions: use obfuscation to hide $(x^\star, y^\star)$

Obfuscated program has PRF key embedded inside and outputs $\mathrm{PRF}(k, x)$ on all inputs $x \neq x^\star$ and $y^\star$ when $x = x^\star$

This work: Under *standard lattice assumptions*, there exists a secretly-verifiable watermarkable family of PRFs

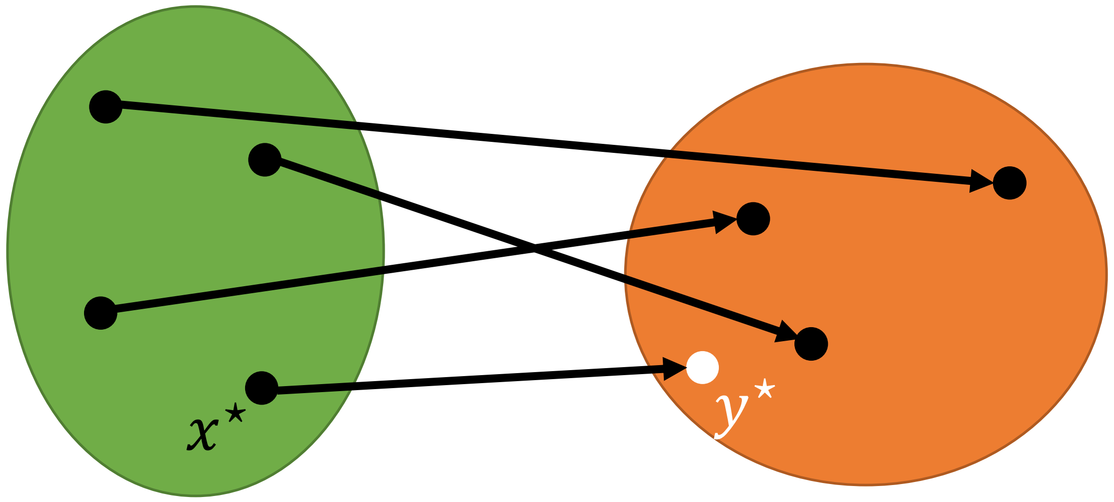
Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]



- Watermarked PRF implements PRF at all but a single point
- Structurally very similar to a *puncturable PRF* [BW13, BGI13, KPTZ13]

Puncturable PRF:

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]



- Watermarked PRF implements PRF at all but a single point
- Structurally very similar to a

Can be used to evaluate the PRF on all points $x \neq x^\star$

Puncturable PRF:

PRF key

punctured key

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]

Recall general approach for watermarking:

1. Derive $(x^\star, y^\star)$ from input/output behavior of PRF
2. Give out a key that agrees with PRF everywhere, except has value $y^\star$ at $x = x^\star$

PRF key punctured at $x^\star$

However, punctured key does not necessarily hide $x^\star$, which allows adversary to remove watermark

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]

PRF key

$\text{Puncture}_{x^\star}$

punctured key

Punctured keys typically do not provide flexibility in programming value at punctured point: difficult to test if a program is watermarked or not

Re

1. …or of PRF
2. Give …ey that agrees with PRF everywhere, except has value $y^\star$ at $x = x^\star$

PRF key punctured at $x^\star$

However, punctured key does not necessarily hide $x^\star$, which allows adversary to remove watermark

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]

Problem 1: Punctured keys do not hide the punctured point $x^\star$

- Use *private* puncturable PRFs

Problem 2: Difficult to test whether a value is the result of using a punctured key to evaluate at the punctured point

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]

In existing lattice-based private puncturable PRF constructions [BKM17, CC17], value of punctured key at punctured point is a *deterministic* function of the PRF key

Problem 1: P

- Use *pri*

Problem 2: Difficult to test whether a value is the result of using a punctured key to evaluate at the punctured point

Starting Point: Private Puncturable PRFs [BLW17, BKM17, CC17]

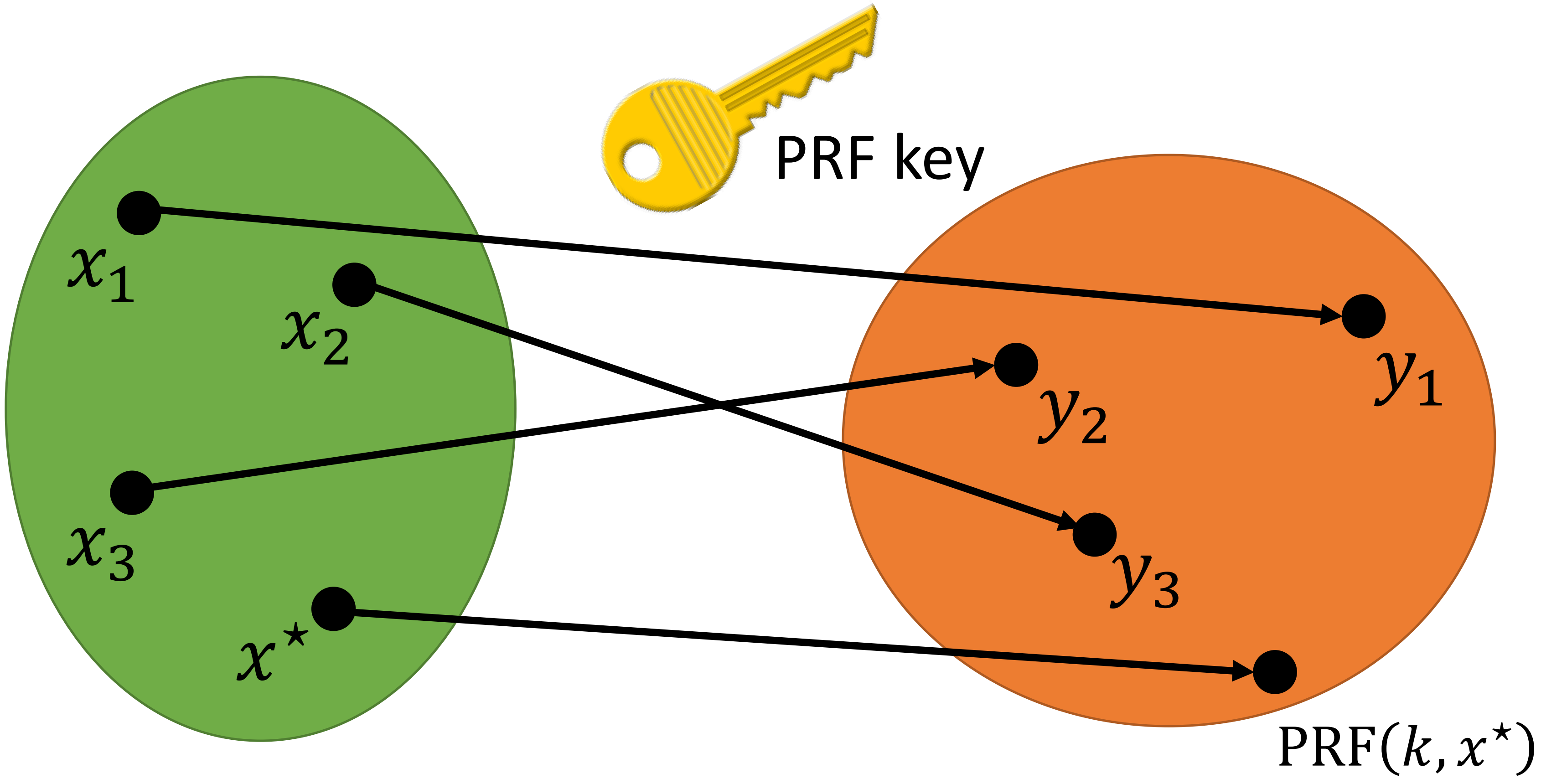
Problem 1: Punctured keys do not hide the punctured point $x^\star$

- Use *privately* puncturable PRFs

Problem 2: Difficult to test whether a value is the result of using a punctured key to evaluate at the punctured point

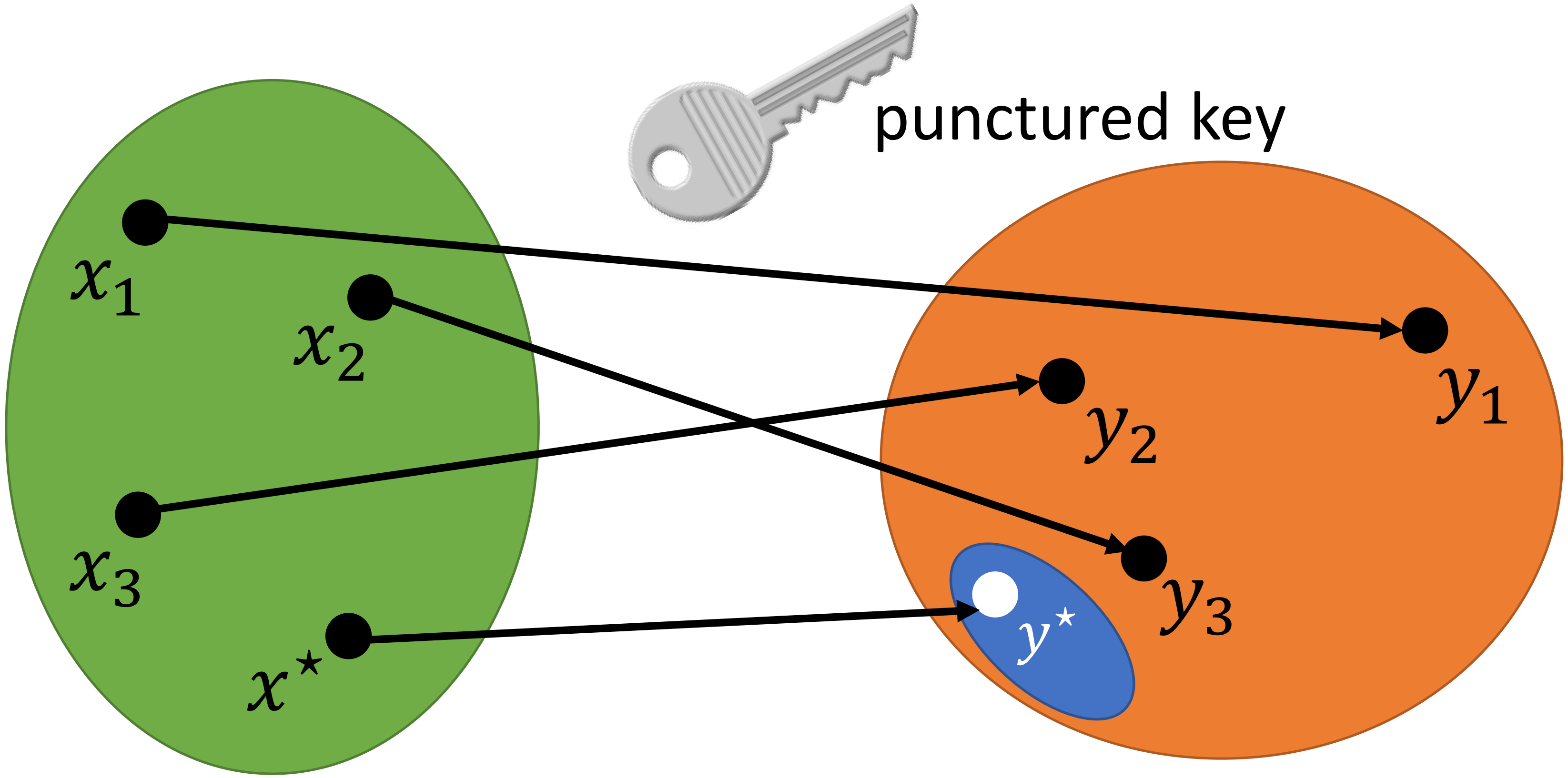
- Relax programmability requirement

Private Translucent PRFs



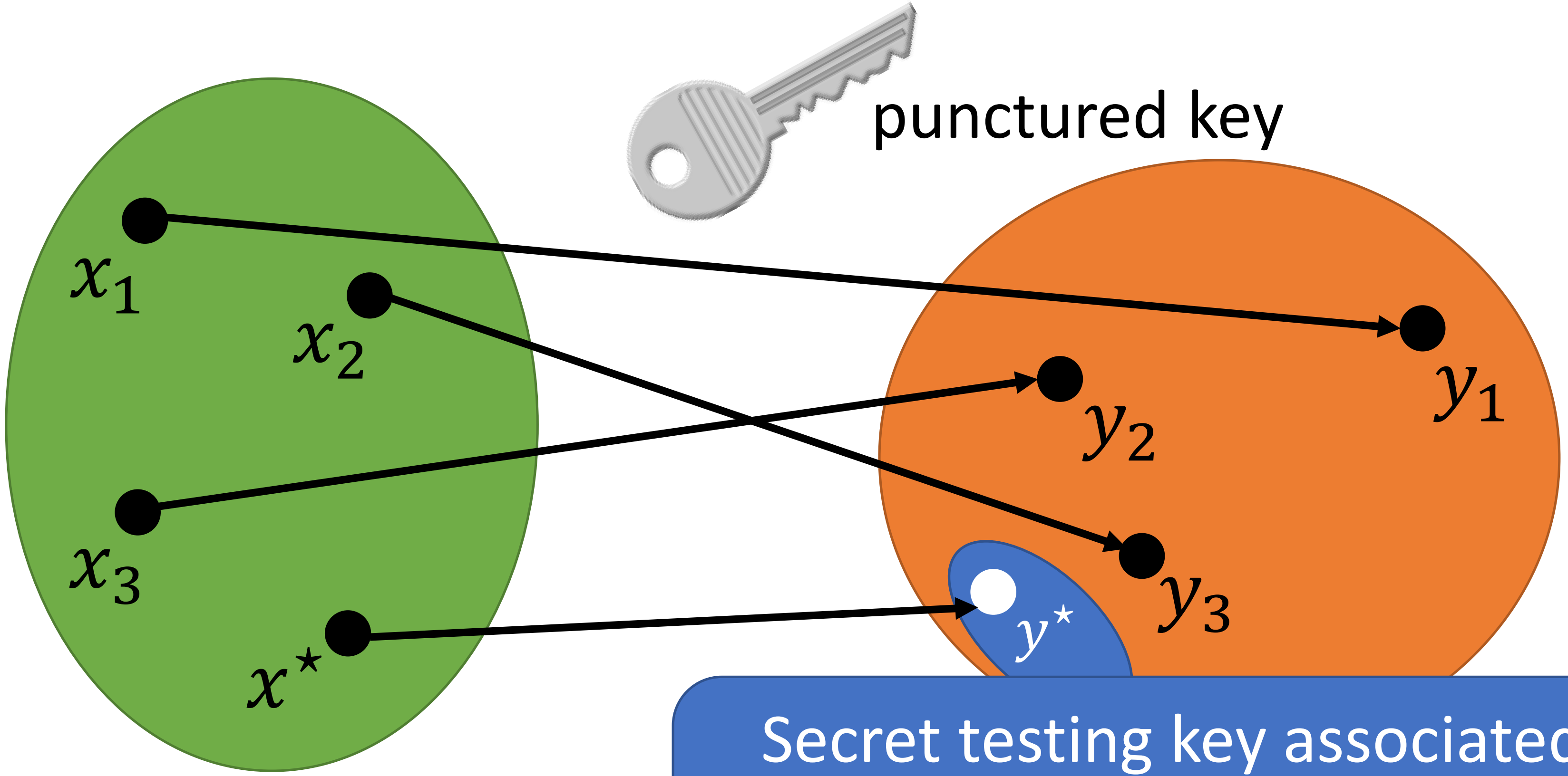
Private puncturable PRF *family* with the property that output of any punctured key on a punctured point lies in a sparse, hidden subspace

Private Translucent PRFs



Private puncturable PRF *family* with the property that output of any punctured key on a punctured point lies in a sparse, hidden subspace

Private Translucent PRFs

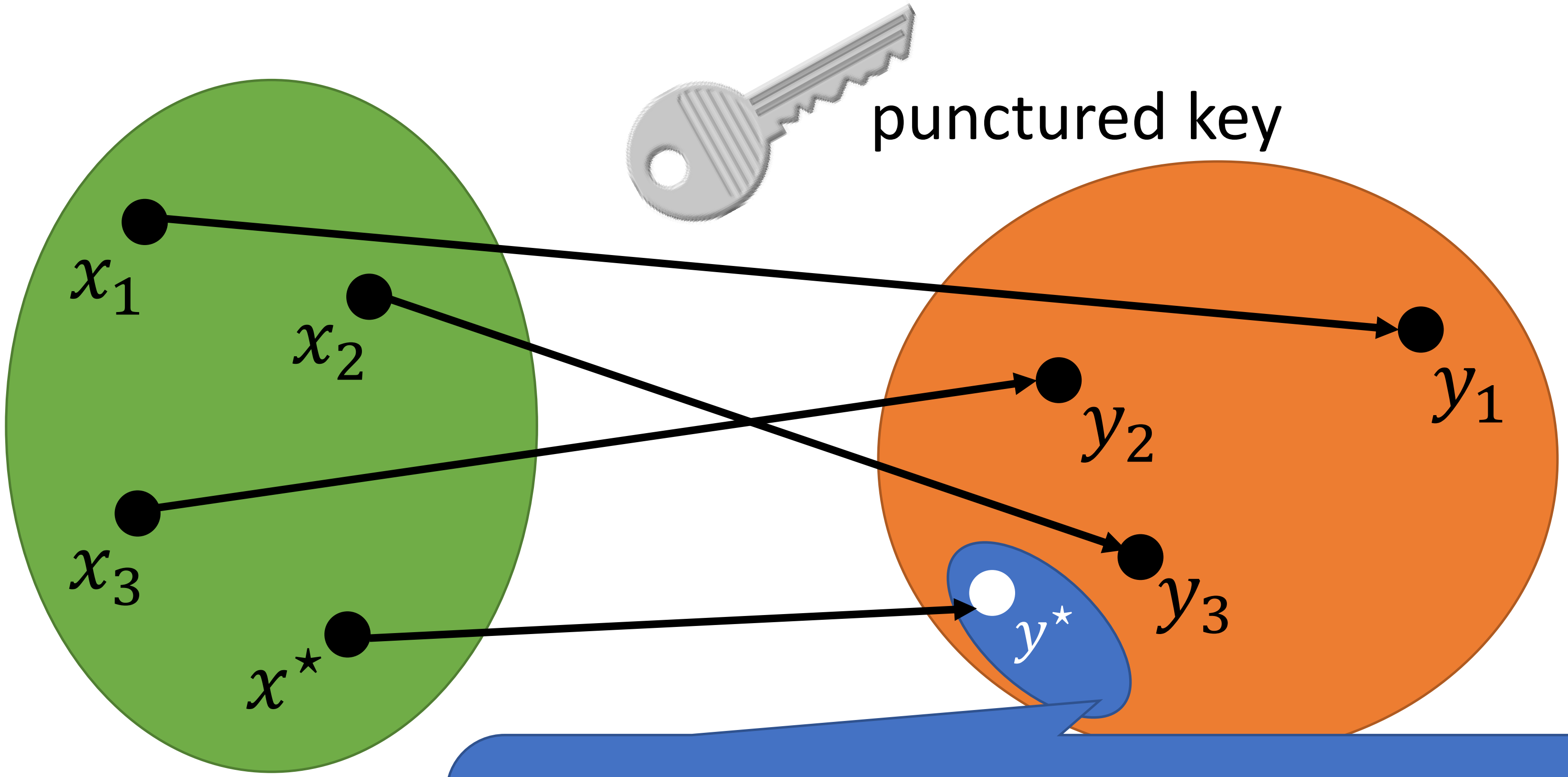


Secret testing key associated with the PRF family can be used to test for membership in the hidden subspace

Private puncturable PRF *family*

punctured key on a punctured point lies in a sparse, hidden subspace

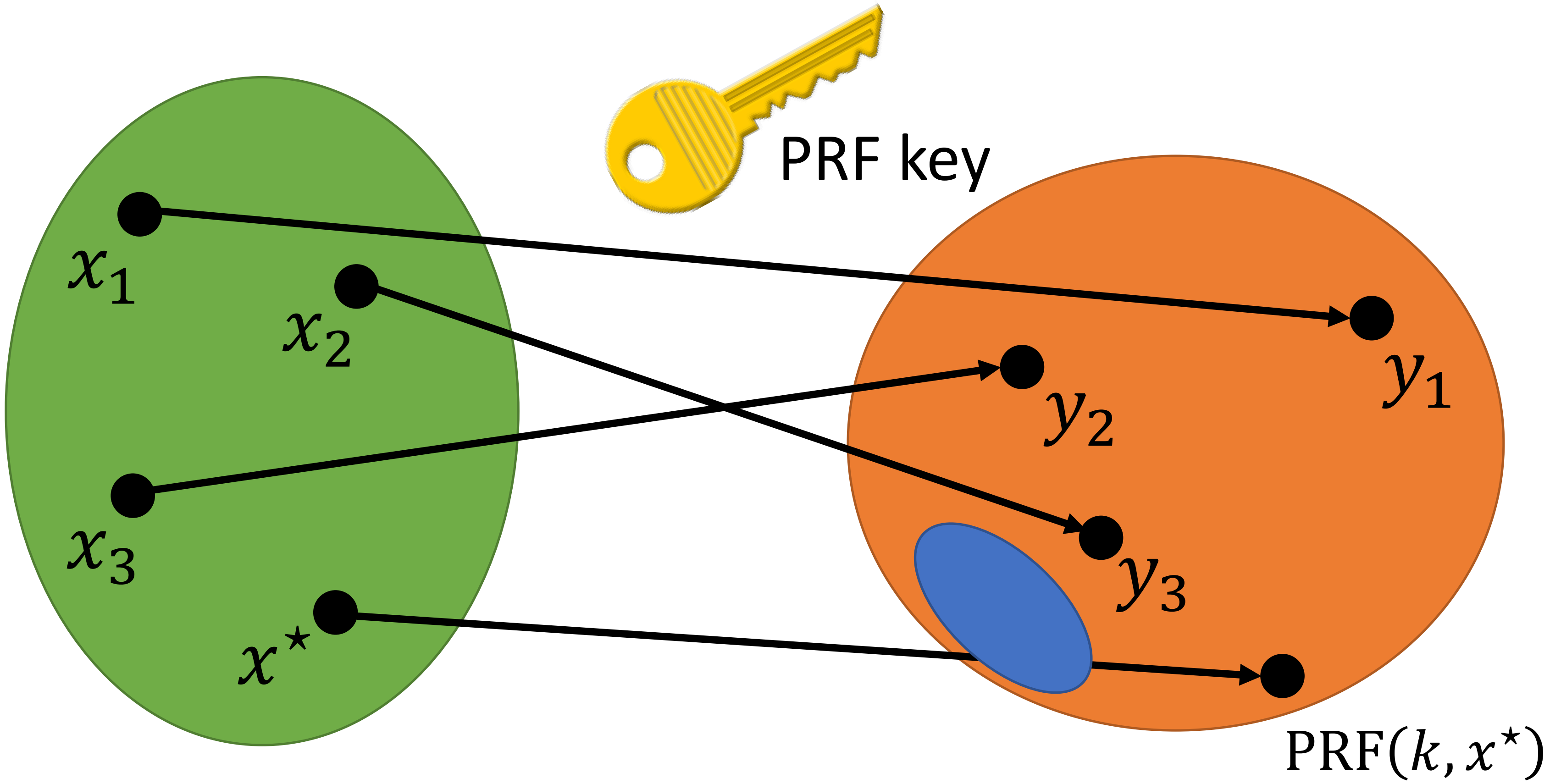
Private Translucent PRFs



Sets satisfying such properties are called *translucent* [CDNO97]

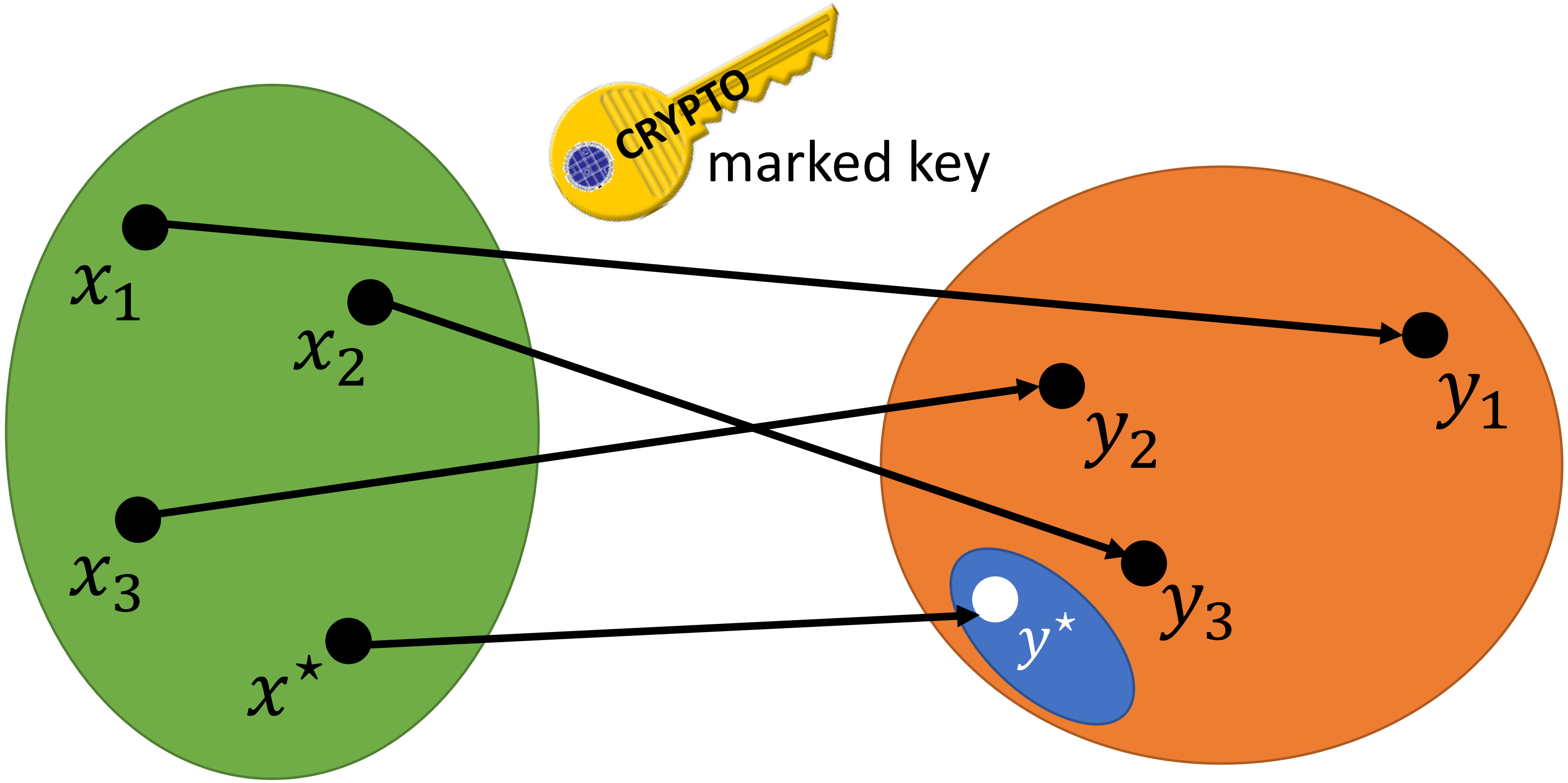
- Values in special set looks indistinguishable from a random value (without secret testing key)
- Indistinguishable even though it is easy to sample values from the set

Watermarking from Private Translucent PRFs



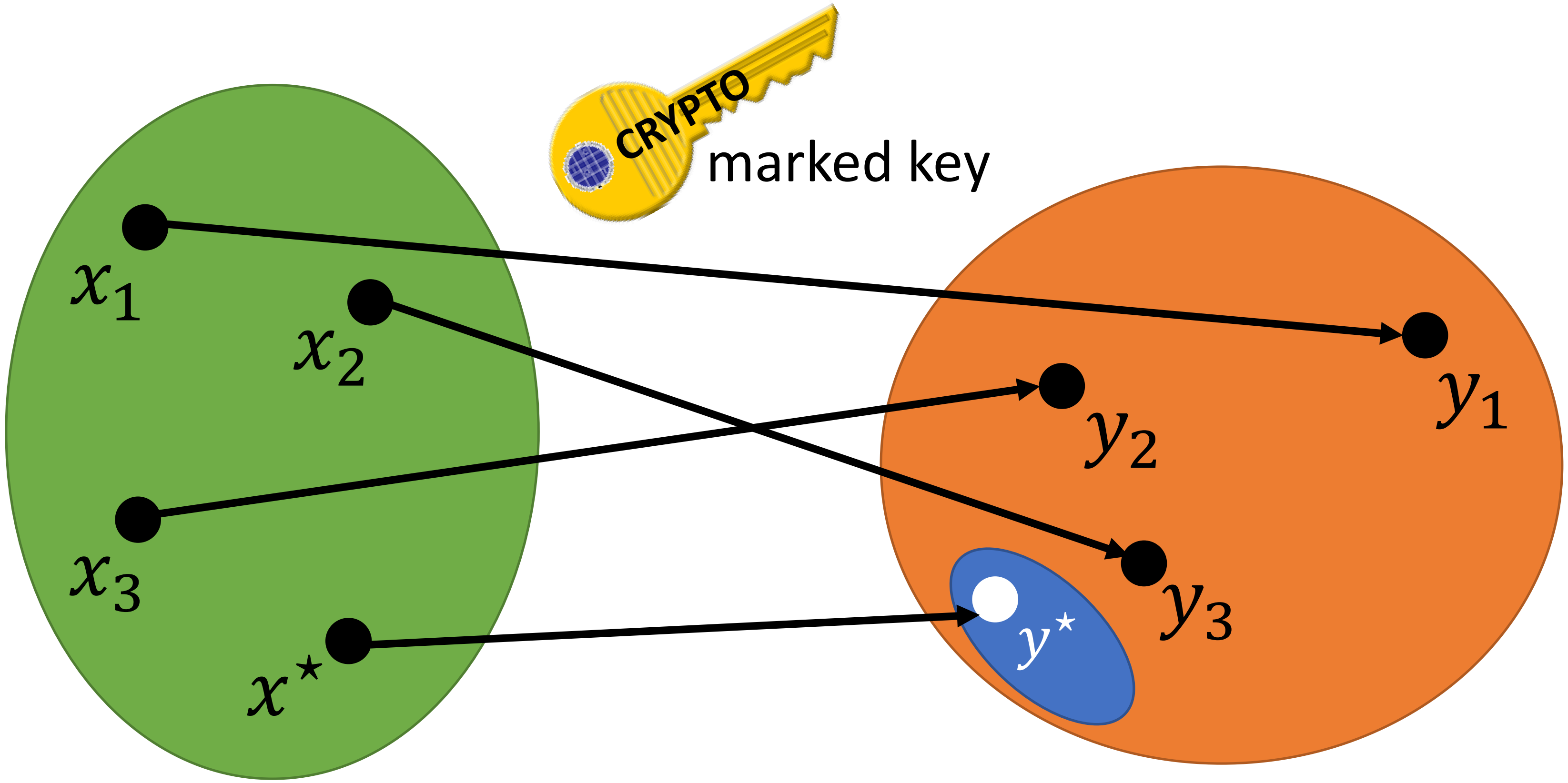
Watermarking secret key (wsk): test points $x_1, \ldots, x_d$ and testing key for private translucent PRF

Watermarking from Private Translucent PRFs



To mark a PRF key k, derive special point $x^\star$ and puncture k at $x^\star$; watermarked key is a program that evaluates using the punctured key

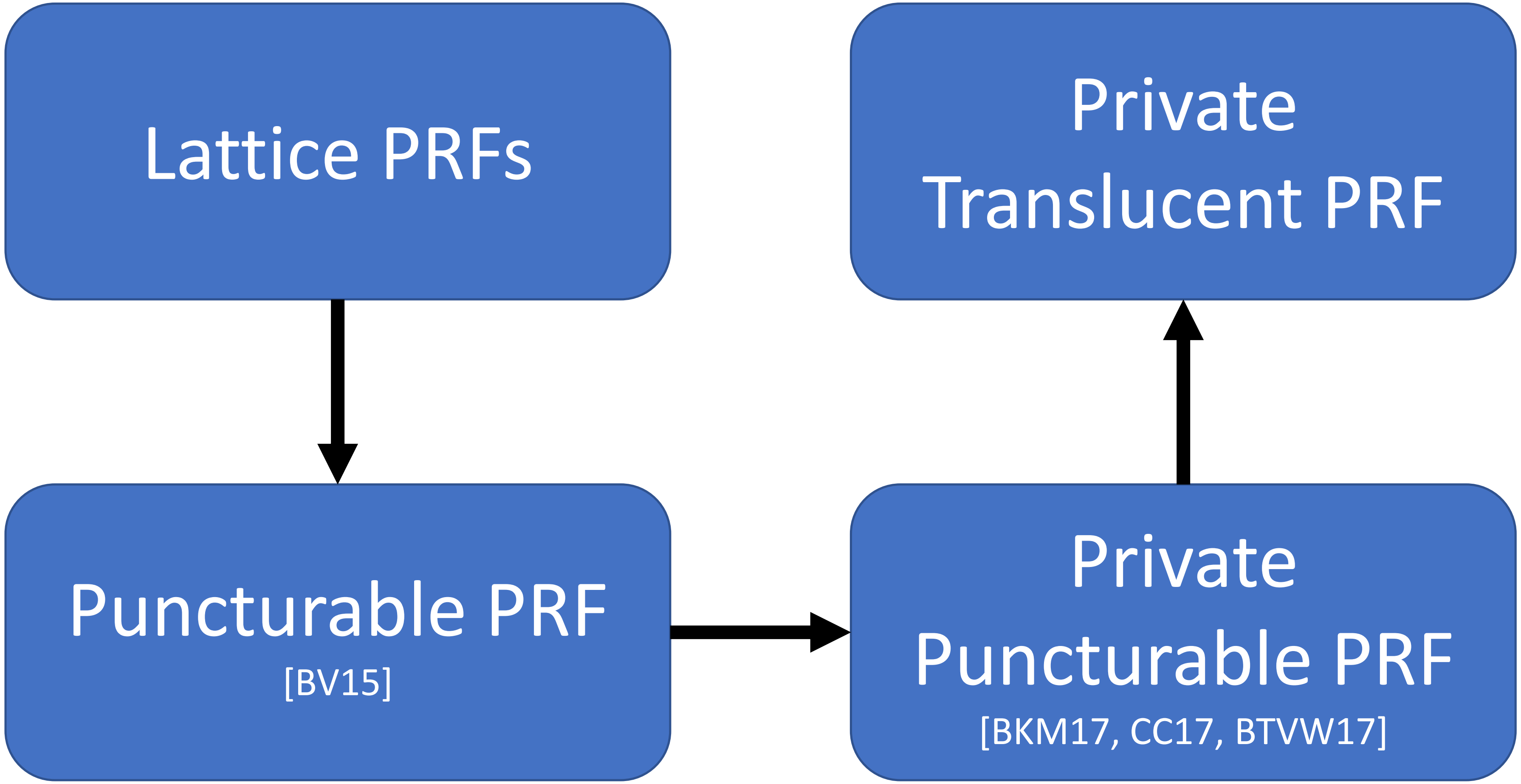
Watermarking from Private Translucent PRFs



To test whether a program C' is watermarked, derive test point $x^\star$ and check whether $C'(x^\star)$ is in the translucent set (using the testing key for the private translucent PRF)

Constructing Private Translucent PRFs

Blueprint


Lattice PRFs
Private
Translucent PRF
Puncturable PRF
[BV15]
Private
Puncturable PRF
[BKM17, CC17, BTVW17]

Learning with Errors (LWE) [Reg05]

$$\left(\boldsymbol{A}, \boldsymbol{s}^T \boldsymbol{A} + \boldsymbol{e}^T\right) \approx_c \left(\boldsymbol{A}, \boldsymbol{u}^T\right)$$

$$\boldsymbol{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}, \boldsymbol{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n, \boldsymbol{e} \xleftarrow{\text{R}} \chi^m, \boldsymbol{u} \xleftarrow{\text{R}} \mathbb{Z}_q^m$$

Learning with Rounding (LWR) [BPR12]

Replace *random* errors with *deterministic* rounding:

$$\left(\boldsymbol{A}, \lfloor \boldsymbol{s}^T \boldsymbol{A} \rceil_p\right) \approx_c \left(\boldsymbol{A}, \lfloor \boldsymbol{u}^T \rceil_p\right)$$

$$\boldsymbol{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}, \boldsymbol{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n, \boldsymbol{u} \xleftarrow{\text{R}} \mathbb{Z}_q^m$$

Hardness reducible to LWE (for suitable parameter settings)

More suitable starting point for constructing lattice PRFs

Lattice PRFs [BPR12, BLMR13, BP14, BV15, BFPPS15, BKM17, BTVW17]

$$\left(\boldsymbol{A}, \left\lfloor \boldsymbol{s}^T \boldsymbol{A} \right\rceil_p\right) \approx_c \left(\boldsymbol{A}, \left\lfloor \boldsymbol{u}^T \right\rceil_p\right)$$

Intuition: set $\boldsymbol{s}$ to be the secret key for the PRF and derive $\boldsymbol{A}$ as a function of the input

Lattice PRFs [BPR12, BLMR13, BP14, BV15, BFPPS15, BKM17, BTVW17]

$$\left(\boldsymbol{A}, \left\lfloor \boldsymbol{s}^T \boldsymbol{A} \right\rceil_p\right) \approx_c \left(\boldsymbol{A}, \left\lfloor \boldsymbol{u}^T \right\rceil_p\right)$$

Fix (public) random matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell \in \mathbb{Z}_q^{n \times m}$

Secret key: LWE secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$

PRF evaluation: on input $x \in \{0,1\}^\ell$, derive $\boldsymbol{A}_x$ from $\boldsymbol{A}_1, \ldots \boldsymbol{A}_\ell$ and output

$$\mathrm{PRF}(\boldsymbol{s}, x) := \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_x \right\rceil_p$$

Question: how to derive $\boldsymbol{A}_x$ from $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell$?

Homomorphic Matrix Embeddings [BGGHNSVV14]

A way to encode $x \in \{0,1\}^{\ell}$ as a collection of LWE samples

take LWE matrices $\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell \in \mathbb{Z}_q^{n \times m}$ and a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

encoding of x_1 with respect to $\boldsymbol{A}_1$

Homomorphic Matrix Embeddings [BGGHNSVV14]

LWE matrix associated with each input bit

$\boldsymbol{G} \in \mathbb{Z}_q^{n \times m}$ is a fixed "gadget" matrix

e … ction of LWE samples

tri … m and a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

encoding of x_1 with respect to $\boldsymbol{A}_1$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

Homomorphic Matrix Embeddings [BGGHNSVV14]

A way to encode $x \in \{0,1\}^\ell$ as a collection of LWE samples

take LWE matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell \in \mathbb{Z}_q^{n \times m}$ and a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

Function of f and $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell$ only

$$\boldsymbol{s}^T\left(\boldsymbol{A}_f + f(x) \cdot \boldsymbol{G}\right) + \text{noise}$$

Encodings support homomorphic operations

Encoding of $x \Longrightarrow$ Encoding of $f(x)$

Puncturable PRFs from LWE [BV15]

PRF evaluation: on input $x \in \{0,1\}^\ell$, derive $\boldsymbol{A}_x$ from $\boldsymbol{A}_1, \dots \boldsymbol{A}_\ell$ and output

$$\text{PRF}(\boldsymbol{s}, x) \coloneqq \lfloor \boldsymbol{s}^T \boldsymbol{A}_x \rceil_p$$

Question: how to derive $\boldsymbol{A}_x$ from $\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell$?

Let $\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell$ be matrices associated with bits of $x \in \{0,1\}^\ell$

Define PRF evaluation with respect to equality function

$$\text{eq}_x(x^\star) = \begin{cases} 1, & x = x^\star \\ 0, & x \neq x^\star \end{cases}$$

Let $\boldsymbol{A}_x$ be matrix associated with evaluating eq_x on $\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell$

Puncturable PRFs from LWE [BV15]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{eq}_x} \right\rceil_p$$

To puncture the key $\boldsymbol{s}$ at a point $x^\star$, give out encodings of $x^\star$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

$$\Rightarrow \quad \boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{eq}_x} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G}\right) + \mathrm{noise}$$

PRF evaluation (at x)
using punctured key

Puncturable PRFs from LWE [BV15]

$$\text{PRF}(\boldsymbol{s}, x) := \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\text{eq}_x} \right\rceil_p$$

To puncture the key $\boldsymbol{s}$ at a point $x^\star$, give out encodings of $x^\star$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\text{eq}_x} + \text{eq}_x(x^\star) \cdot \boldsymbol{G}\right) + \text{noise}$$

PRF evaluation (at x) using punctured key

If $x \neq x^\star$, $\text{eq}_x(x^\star) = 0$, so

$$\left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\text{eq}_x} + \text{noise} \right\rceil_p = \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\text{eq}_x} \right\rceil_p = \text{PRF}(\boldsymbol{s}, x)$$

Puncturable PRFs from LWE [BV15]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{eq}_x} \right\rceil_p$$

To puncture the key $\boldsymbol{s}$ at a point $x^\star$, give out encodings of $x^\star$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{eq}_x} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G}\right) + \text{noise}$$

PRF evaluation (at x) using punctured key

If $x = x^\star$, $\mathrm{eq}_x(x^\star) = 1$, so

$$\left\lfloor \boldsymbol{s}^T(\boldsymbol{A}_{\mathrm{eq}_{x^\star}} + \boldsymbol{G}) + \text{noise} \right\rceil_p \neq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{eq}_{x^\star}} \right\rceil_p = \mathrm{PRF}(\boldsymbol{s}, x^\star)$$

Puncturable PRFs from LWE [BV15]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{eq}_x} \right\rceil_p$$

To puncture the key $\boldsymbol{s}$ at a point $x^\star$, give out encodings of $x^\star$:

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{eq}_x} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G}\right) + \mathrm{noise}$$

PRF evaluation (at x) using punctured key

This construction gives a puncturable PRF from LWE

Private Puncturable PRFs [BKM17, BTVW17]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{eq}_x} \right\rceil_p$$

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + x_1^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$

$$\vdots$$

$$\boldsymbol{s}^T(\boldsymbol{A}_\ell + x_\ell^\star \cdot \boldsymbol{G}) + \boldsymbol{e}_\ell$$

Evaluating PRF using punctured key requires knowledge of $x^\star$

Key idea in [BKM17]: encrypt the punctured point using an FHE scheme and homomorphically evaluate the equality function

Private Puncturable PRFs [BKM17, BTVW17]

$$\mathrm{PRF}(\boldsymbol{s}, x) := \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}} \right\rceil_p$$

FHE decryption + homomorphic evaluation of eq_x

Punctured key consists of encodings of encrypted point (for homomorphic evaluation) and FHE secret key (for decryption)

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + \mathrm{ct}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T(\boldsymbol{A}_z + \mathrm{ct}_z \cdot \boldsymbol{G}) + \boldsymbol{e}_z$$

$$\boldsymbol{s}^T(\boldsymbol{B}_1 + \mathrm{sk}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T(\boldsymbol{B}_\tau + \mathrm{sk}_\tau \cdot \boldsymbol{G}) + \boldsymbol{e}_\tau$$

ct is an FHE encryption of $x^\star$

sk is the FHE secret key

Private Puncturable PRFs [BKM17, BTVW17]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_x}} \right\rceil_p$$

$$\boldsymbol{s}^T(\boldsymbol{A}_1 + \mathrm{ct}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T(\boldsymbol{A}_z + \mathrm{ct}_z \cdot \boldsymbol{G}) + \boldsymbol{e}_z$$

$$\boldsymbol{s}^T(\boldsymbol{B}_1 + \mathrm{sk}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T(\boldsymbol{B}_\tau + \mathrm{sk}_\tau \cdot \boldsymbol{G}) + \boldsymbol{e}_\tau$$

Evaluating $\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}$ on encodings essentially yields:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_x}} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G} \right) + \mathrm{noise}$$

Private Puncturable PRFs [BKM17, BTVW17]

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}} \right\rceil_p$$

Some technicalities due to FHE noise (will ignore here for simplicity)

$$\boldsymbol{s}^T (\boldsymbol{A}_1 + \mathrm{ct}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T (\boldsymbol{A}_z + \mathrm{ct}_z \cdot \boldsymbol{G}) + \boldsymbol{e}_z$$

$$\boldsymbol{s}^T (\boldsymbol{B}_1 + \mathrm{sk}_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{s}^T (\boldsymbol{B}_\tau + \mathrm{sk}_\tau \cdot \boldsymbol{G}) + \boldsymbol{e}_\tau$$

Evaluating $\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}$ on encodings essentially yields:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G} \right) + \mathrm{noise}$$

Evaluation only requires knowledge of ct and <u>not</u> sk

Private Translucent PRFs

Goal: detect whether a punctured key is used to evaluate at a punctured point (this is essential for embedding the watermark)

Real PRF evaluation: $\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathtt{Decrypt} \circ \mathtt{Eval}_{\mathtt{eq}_x}} \right\rceil_p$

Punctured PRF evaluation: $\left\lfloor \boldsymbol{s}^T \left(\boldsymbol{A}_{\mathtt{Decrypt} \circ \mathtt{Eval}_{\mathtt{eq}_x}} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G} \right) \right\rceil_p$

Difficulty: no control over value at punctured point

Private Translucent PRFs

Goal: detect whether a punctured key is used to evaluate at a punctured point (this is essential for embedding the watermark)

Real PRF evaluation: $\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}} \right\rceil_p$

Punctured PRF evaluation: $\left\lfloor \boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_x}} + \mathrm{eq}_x(x^\star) \cdot \boldsymbol{G} \right) \right\rceil_p$

Idea: define PRF with respect to <u>scaled</u> equality circuit:

$$\mathrm{eq}_x(x^\star, w) = \begin{cases} w, & x = x^\star \\ 0, & x \neq x^\star \end{cases}$$

Private Translucent PRFs

$$\text{PRF}(\boldsymbol{s}, x) := \left\lfloor \boldsymbol{s}^T \boldsymbol{A}_{\mathtt{Decrypt}\circ\mathtt{Eval}_{\mathtt{eq}_x}} \right\rceil_p$$

Evaluating the punctured key at the punctured point $x^\star$ yields:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathtt{Decrypt}\circ\mathtt{Eval}_{\mathtt{eq}_{x^\star}}} + w \cdot \boldsymbol{G} \right) + \text{noise}$$

Scaling factor w is chosen when key is punctured and can be chosen to adjust the value at the punctured point

Private Translucent PRFs

Evaluating the punctured key at the punctured point yields:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star}}} + w \cdot \boldsymbol{G} \right) + \mathrm{noise}$$

Can now consider many instances of this PRF with many different w_i's:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star,1}}} + w_1 \cdot \boldsymbol{G}_1 \right) + \mathrm{noise}$$

$$\vdots$$

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star,N}}} + w_N \cdot \boldsymbol{G}_N \right) + \mathrm{noise}$$

Different gadget matrices $\boldsymbol{G}_1, \ldots, \boldsymbol{G}_N$

[See paper for construction]

Private Translucent PRFs

Evaluating the punctured key at the punctured point yields:

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_{x^\star}}} + w \cdot \boldsymbol{G}\right) + \mathrm{noise}$$

Can now consider many instances of this PRF with many different w_i's:

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_{x^\star,1}}} + w_1 \cdot \boldsymbol{G}_1\right) + \mathrm{noise}$$

$$\vdots$$

$$\boldsymbol{s}^T\left(\boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_{x^\star,N}}} + w_N \cdot \boldsymbol{G}_N\right) + \mathrm{noise}$$

At puncturing time, choose $w_1, \ldots, w_N$ such that

$$\boldsymbol{W} = \sum_{i\in[N]} \boldsymbol{A}_{\mathrm{Decrypt}\circ\mathrm{Eval}_{\mathrm{eq}_{x^\star,i}}} + \sum_{i\in[N]} w_i \cdot \boldsymbol{G}_i$$

Private Translucent PRFs

Evaluating the punctured key at the punctured point yields:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star}}} + w \cdot \boldsymbol{G}\right) + \text{noise}$$

Can now consider many instances of this PRF with many different w_i's:

$$\boldsymbol{s}^T \left(\boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star,1}}} + w_1 \cdot \boldsymbol{G}_1\right) + \text{noise}$$

$$\vdots$$

$$\mathrm{Eval}_{\mathrm{eq}_{x^\star,N}} + w_N \cdot \boldsymbol{G}_N\Big) + \text{noise}$$

$\boldsymbol{W}$ is a fixed public matrix included in the public parameters of the PRF family

$_N$ such that

$$\boldsymbol{W} = \sum_{i \in [N]} \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star,i}}} + \sum_{i \in [N]} w_i \cdot \boldsymbol{G}_i$$

Private Translucent PRFs

Define real PRF evaluation to be sum of each independent evaluation:

$$\mathrm{PRF}(\boldsymbol{s}, x) := \left\lfloor \boldsymbol{s}^T \sum_{i \in [N]} \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x,i}}} \right\rceil_p$$

When evaluating at punctured point $x^\star$:

$$\boldsymbol{s}^T \left(\sum_{i \in [N]} \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star,i}}} + \sum_{i \in [N]} w_i \cdot \boldsymbol{G}_i \right) = \boldsymbol{s}^T \boldsymbol{W}$$

Private Translucent PRFs

Define real PRF evaluation to be sum of each independent evaluation:

$$\mathrm{PRF}(\boldsymbol{s}, x) \coloneqq \left\lfloor \boldsymbol{s}^T \sum_{i} \right.$$

Output at punctured point is an LWE sample with respect to $\boldsymbol{W}$ (fixed public matrix) – critical for implementing a translucent set

When evaluating at punctured point

$$\boldsymbol{s}^T \left(\sum_{i \in [N]} \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star, i}}} + \sum_{i \in [N]} w_i \cdot \boldsymbol{G}_i \right) = \boldsymbol{s}^T \boldsymbol{W}$$

Private Translucent PRFs

Define real PRF evaluation to be sum of each independent evaluation:

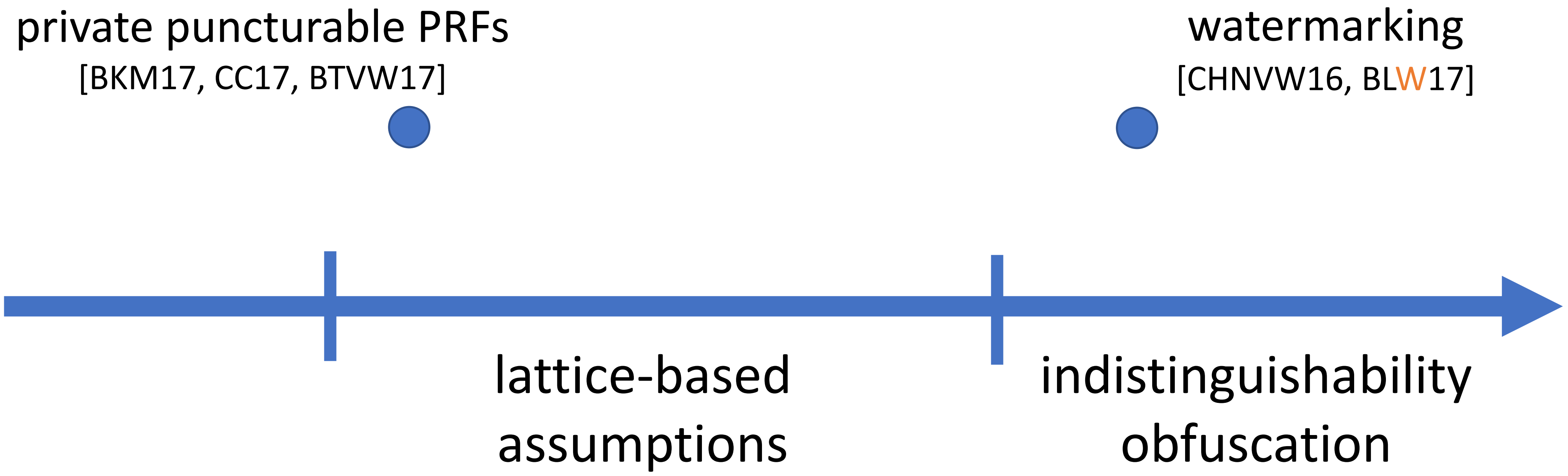
Testing key is a short vector $\boldsymbol{z}$ where $\boldsymbol{W}\boldsymbol{z} = 0$:

$$\left\langle \lfloor \boldsymbol{s}^T \boldsymbol{W} \rceil_p, \boldsymbol{z} \right\rangle \approx \lfloor \boldsymbol{s}^T \boldsymbol{W}\boldsymbol{z} \rceil_p = 0$$

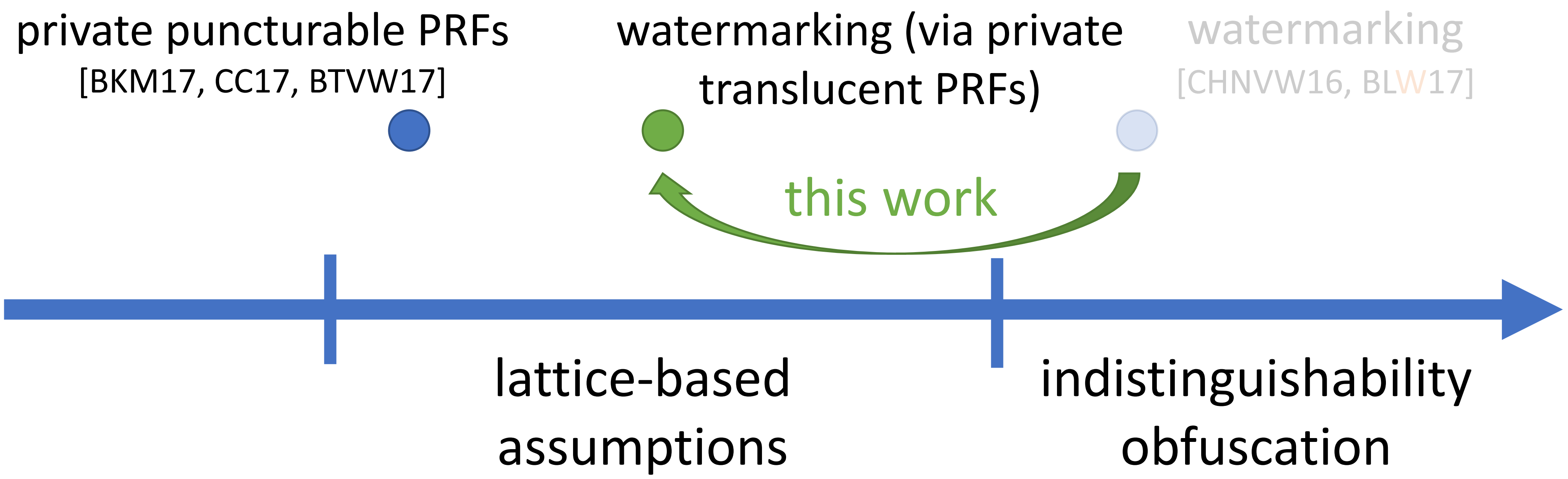
$$\boldsymbol{s}^T \left(\sum_{i \in [N]} \boldsymbol{A}_{\mathrm{Decrypt} \circ \mathrm{Eval}_{\mathrm{eq}_{x^\star, i}}} + \sum_{i \in [N]} w_i \cdot \boldsymbol{G}_i \right) = \boldsymbol{s}^T \boldsymbol{W}$$

[See paper for details and security analysis]

Conclusions



Conclusions



Open Problems

Publicly-verifiable watermarking without obfuscation?

- Current best construction relies on iO [CHNVW16]

Additional applications of private translucent PRFs?

Thank you!

http://eprint.iacr.org/2017/380