Watermarking Cryptographic Functionalities from Standard Lattice Assumptions

Sam Kim

Stanford University

Joint work with David J. Wu

Digital Watermarking

Digital Watermarking

- Content is (mostly) viewable

Digital Watermarking

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

Watermarking Programs

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

Embed a “mark” within a program

Watermarking Programs

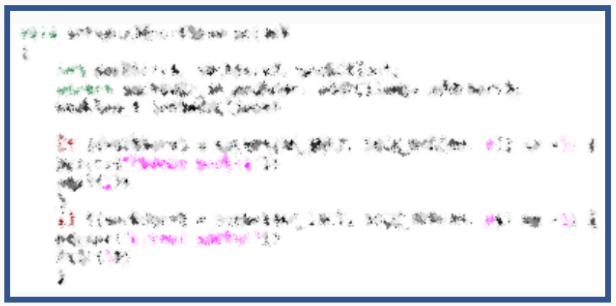
Watermarking Programs

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

```
void serveur1(portServ ports)
{
    int sockServ1, sockServ2, sockClient;
    struct sockaddr_in monAddr, addrClient, addrServ2;
    socklen_t lenAddrClient;

    if ((sockServ1 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
    if ((sockServ2 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
```

CRYPTO



Embed a “mark” within a program

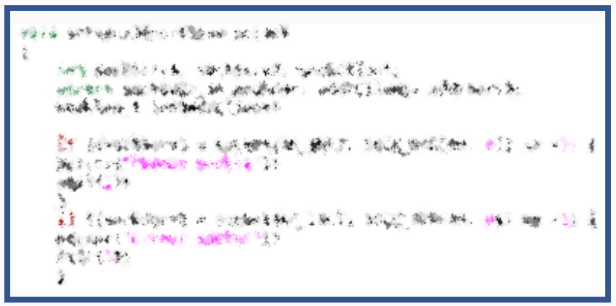
If mark is removed, then program is destroyed

Watermarking Programs

Watermarking Programs

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

Embed a "mark" within a program



If mark is removed, then program is destroyed

Two main algorithms:

- Mark(wsk, C) $\rightarrow C'$: Takes circuit C and outputs marked circuit C'
- Verify(wsk, C') $\rightarrow \{0,1\}$: Tests whether a circuit C' is marked or not

Watermarking Programs

Two main algorithms:

- $\mathsf{Mark}(\mathsf{wsk}, C) \to C'$: Takes circuit C and outputs marked circuit C'
- $\mathsf{Verify}(\mathsf{wsk}, C') \to \{0, 1\}$: Tests whether a circuit C' is marked or not

Watermarking Programs

Two main algorithms:

- Mark(wsk, C) $\to C'$: Takes circuit C and outputs marked circuit C'
- Verify(wsk, C') $\to \{0, 1\}$: Tests whether a circuit C' is marked or not

Functionality Preserving: On input a circuit C, the Mark algorithm outputs a circuit C' where:

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

for all inputs x.

Watermarking Programs

Two main algorithms:

- $\mathsf{Mark}(\mathsf{wsk}, C) \to C'$: Takes circuit C and outputs marked circuit C'
- $\mathsf{Verify}(\mathsf{wsk}, C') \to \{0, 1\}$: Tests whether a circuit C' is marked or not

Functionality Preserving: On input a circuit C, the Mark algorithm outputs a circuit C' where:

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

for all inputs x.

Too Strong: Perfect functionality preserving is impossible for watermarking assuming indistinguishability obfuscation [BGIRSVY12].

Watermarking Programs

Two main algorithms:

- Mark(wsk, C) $\to C'$: Takes circuit C and outputs marked circuit C'
- Verify(wsk, C') $\to \{0, 1\}$: Tests whether a circuit C' is marked or not

Functionality Preserving: On input a 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

Two main algorithms:

- $\mathsf{Mark}(\mathsf{wsk}, C) \to C'$: Takes circuit C and outputs marked circuit C'
- $\mathsf{Verify}(\mathsf{wsk}, C') \to \{0, 1\}$: Tests whether a circuit C' is marked or not

Unremovability: Given a marked program C, no efficient adversary can construct a circuit C' where

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

Watermarking Programs

Two main algorithms:

- $\mathsf{Mark}(\mathsf{wsk}, C) \to C'$: Takes circuit C and outputs marked circuit C'
- $\mathsf{Verify}(\mathsf{wsk}, C') \to \{0, 1\}$: Tests whether a circuit C' is marked or not

Unremovability: Given a marked program C, no efficient adversary can construct a circuit C' where

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

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

- for all i: $C_i(x) \neq C'(x)$ on a noticeable fraction of inputs x
- $\mathsf{Verify}(\mathsf{wsk}, C') = 1$

Can we watermark any programs?

Watermarking only achievable for functions that are not learnable.

Can we watermark any programs?

Watermarking only achievable for functions that are not learnable.

Focus has been on cryptographic functions.

Can we watermark any programs?

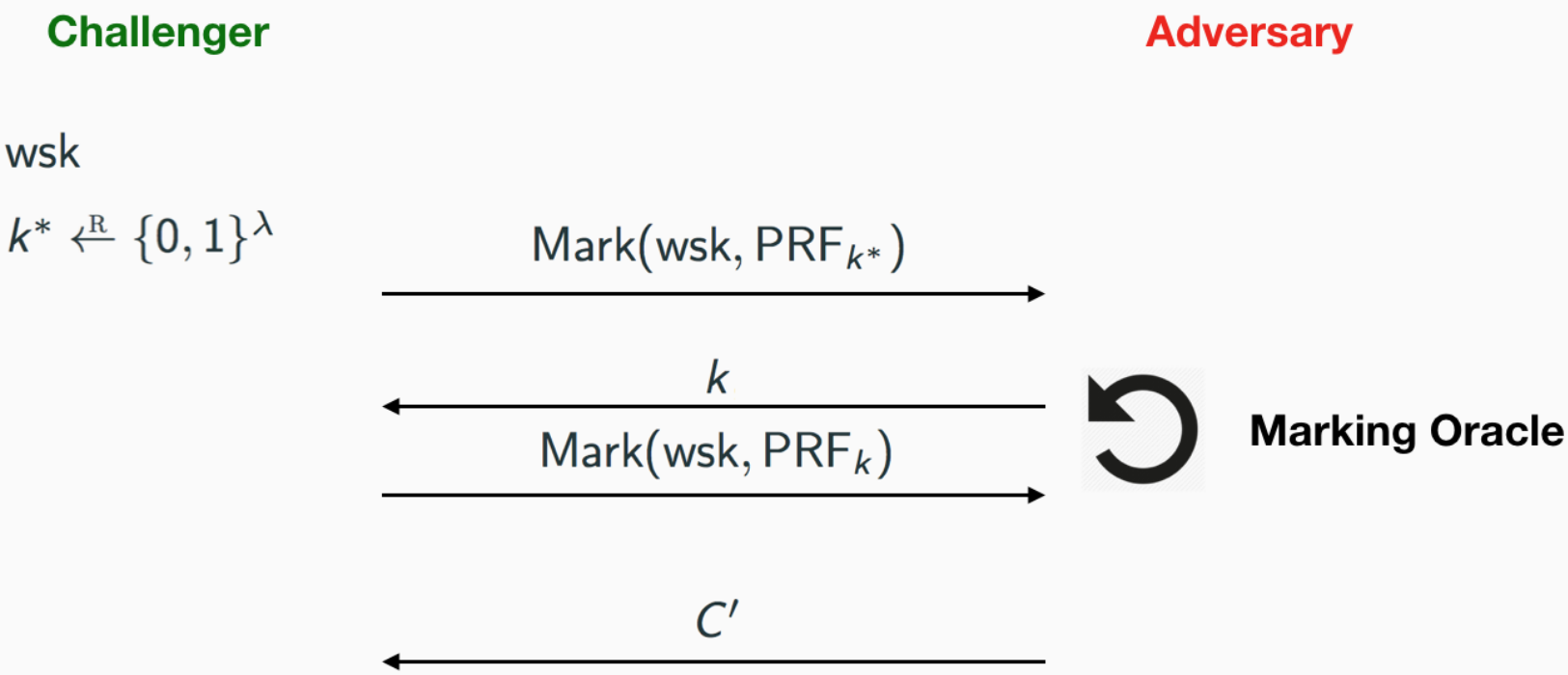
Watermarking only achievable for functions that are not learnable.

Focus has been on cryptographic functions.

This work: Focus on watermarking **PRFs** [CHNVW16, BLW17].

Enables watermarking of symmetric primitives built from PRFs (e.g., encryption, MACs, etc.)

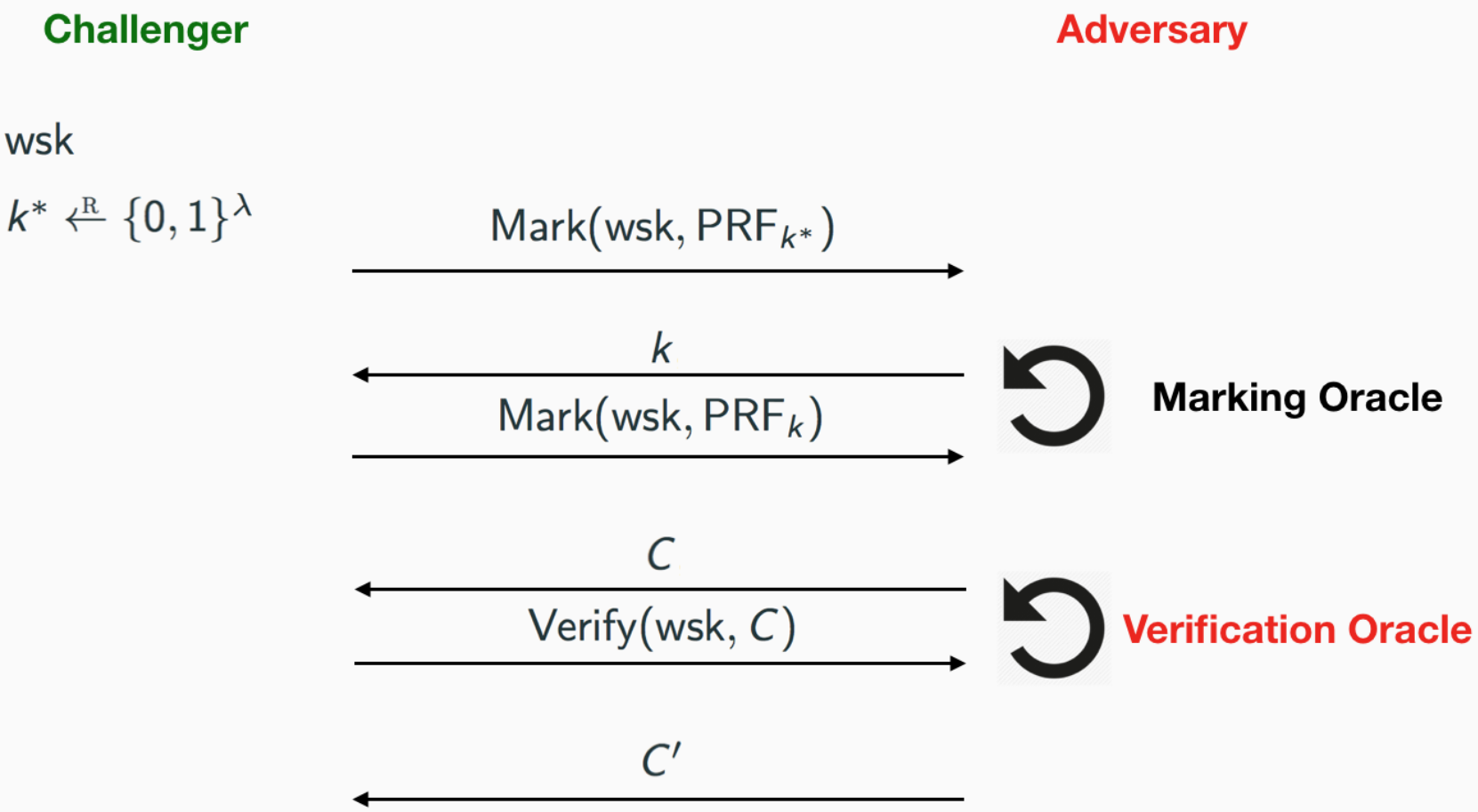
Security Definition (Weaker)



Adversary **wins** if

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

Security Definition (Stronger)



Adversary **wins** if

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

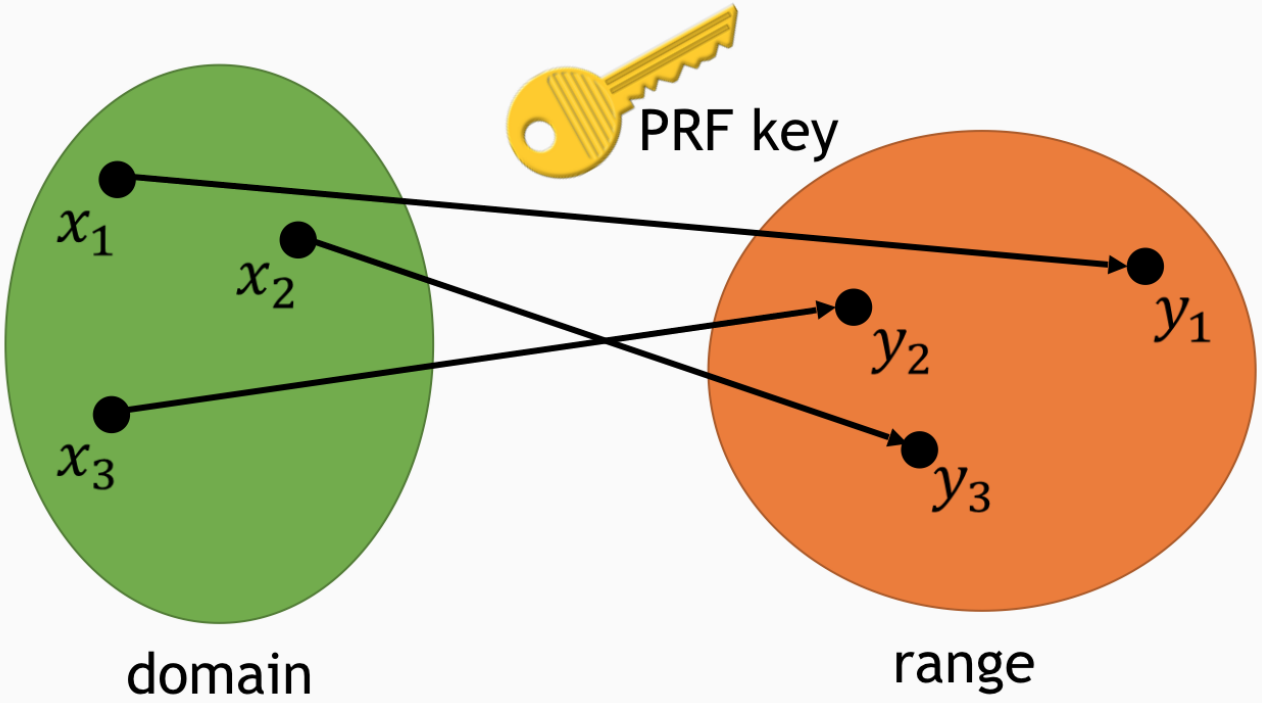
Main Results

Construct watermarking scheme for PRFs satisfying
(weaker) unremovability security definition (and unforgeability)
from LWE

Previous works:

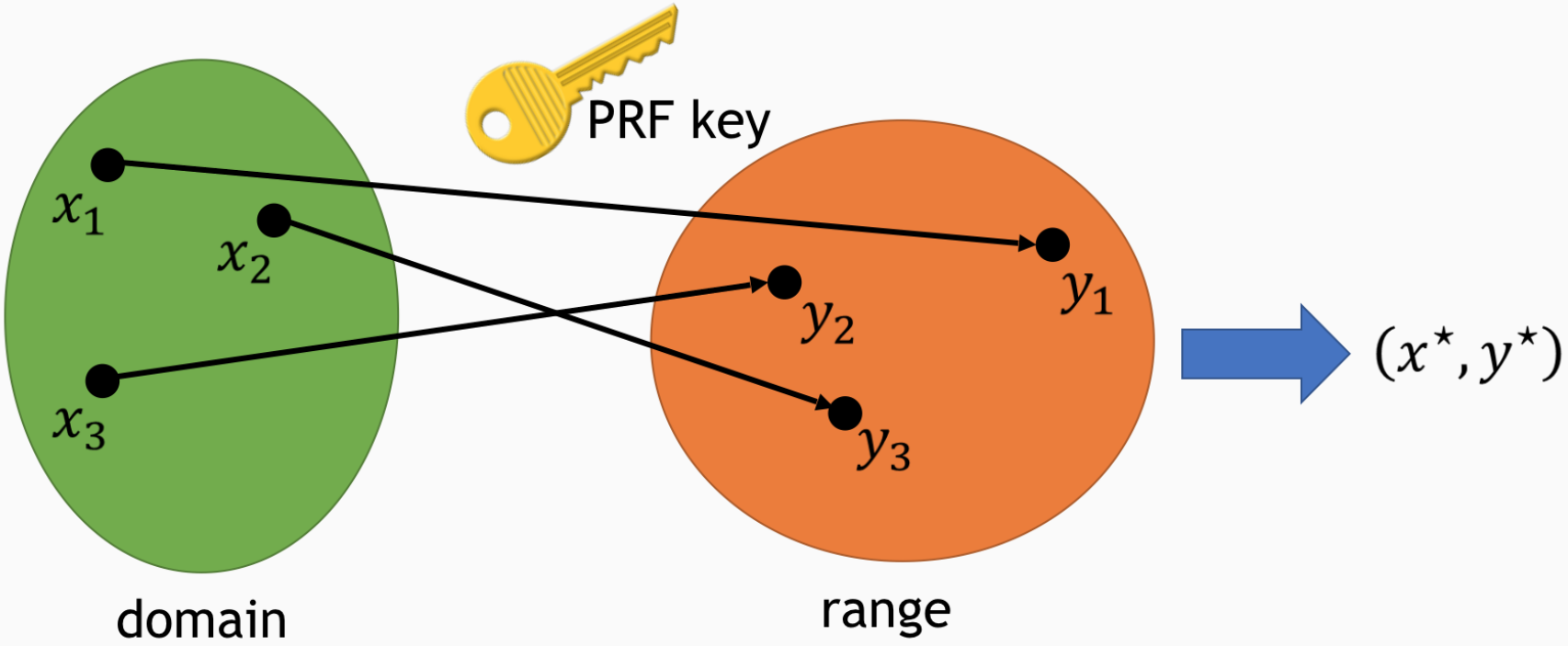
- [CHNVW16]: *(publicly verifiable)* unremovability from iO
- [BLW17]: *(weaker)* unremovability from iO

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



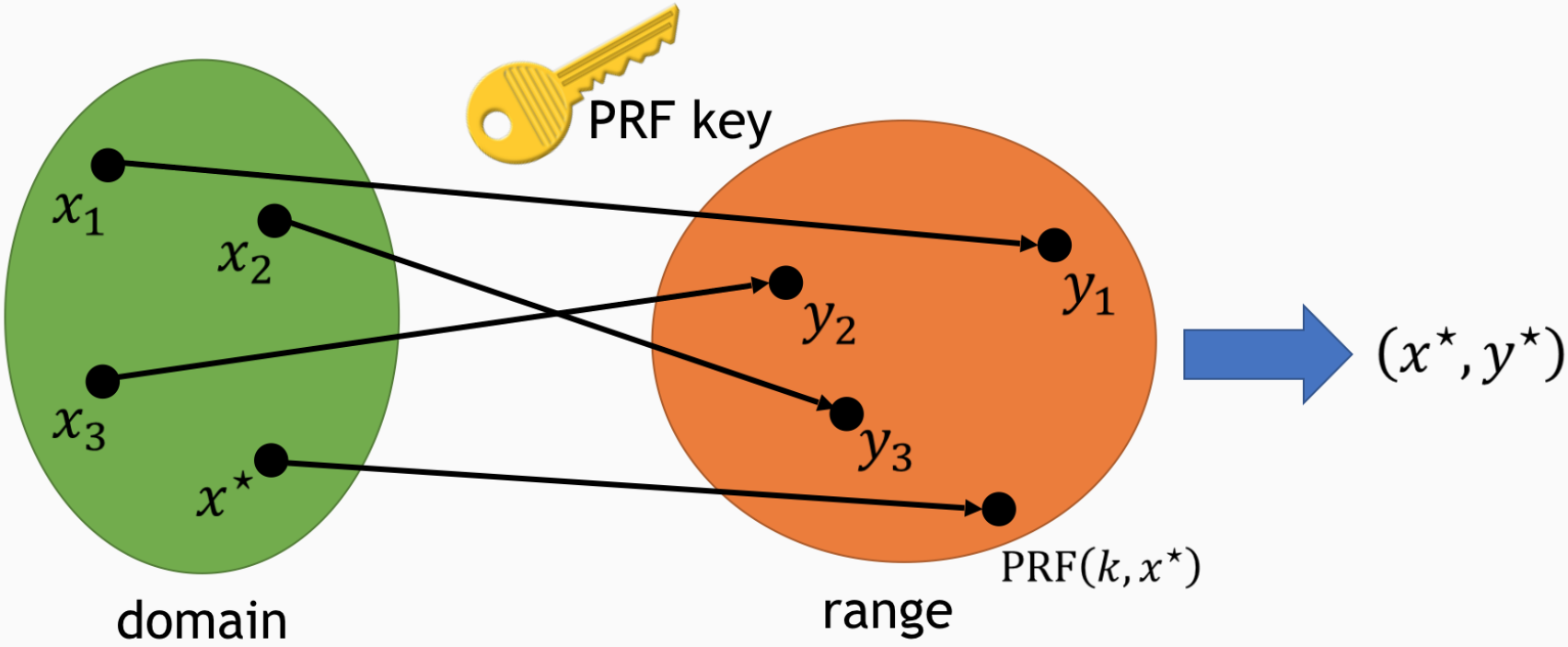
Step 1: Evaluate PRF on test points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ (part of the watermarking secret key)

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



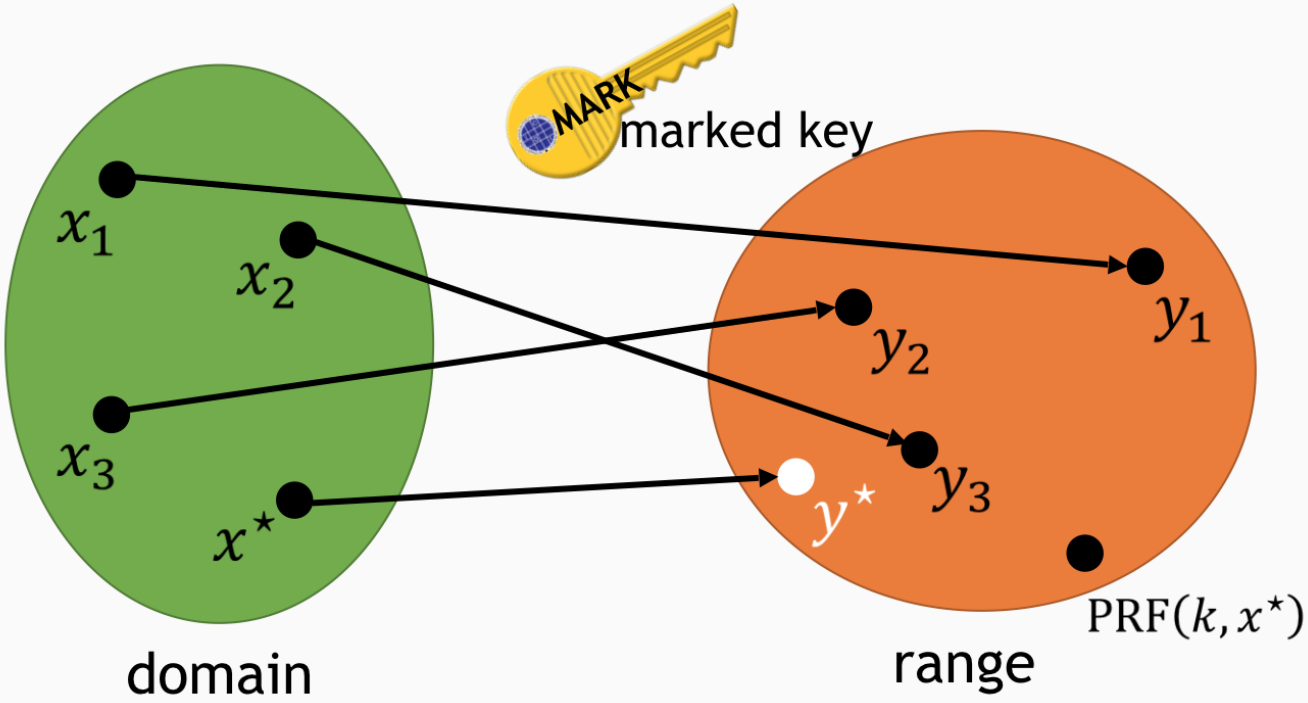
Step 2: Derive a pair $(\mathbf{x}^\star, \mathbf{y}^\star)$ from $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



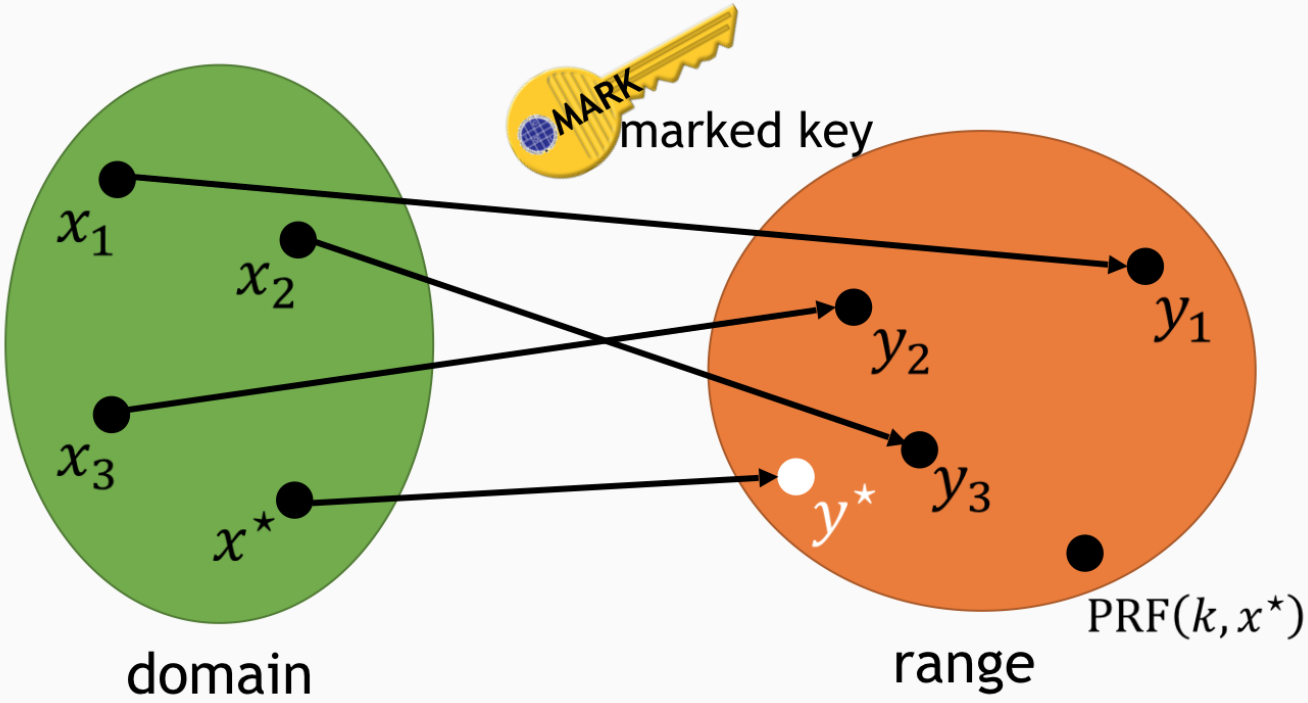
Step 3: "Marked key" is a circuit that implements the PRF at all points, except at $\mathbf{x}^\star$, the output is changed to $\mathbf{y}^\star$

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



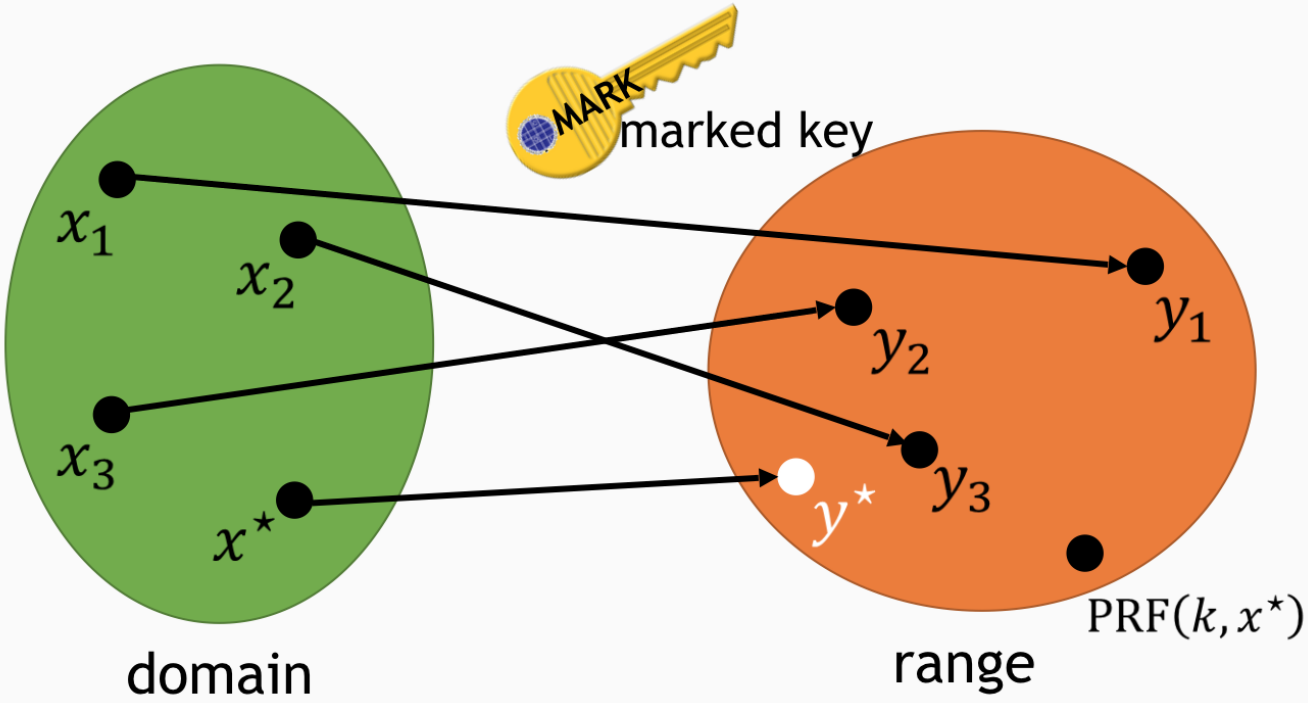
Step 3: "Marked key" is a circuit that implements the PRF at all points, except at $\mathbf{x}^\star$, the output is changed to $\mathbf{y}^\star$

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



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

Blueprint for Watermarking PRFs [CHNVW16, BLW17]



Functionality-preserving: Function differs at a single point

Unremovable: As long as adversary cannot tell that $(\mathbf{x}^\star, \mathbf{y}^\star)$ is "special"

Blueprint for Watermarking PRFs [CHNVW16, BLW17]

Obfuscated program:

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

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

Prior solutions: Use obfuscation to hide $(\mathbf{x}^\star, \mathbf{y}^\star)$

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

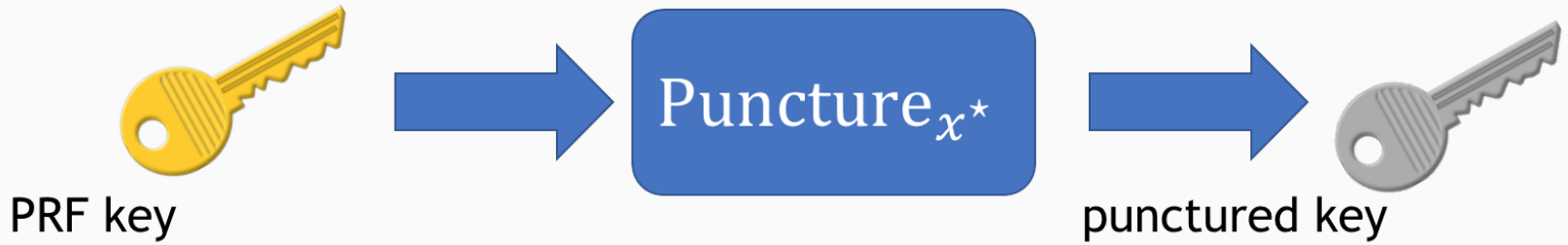
Question: How can we implement the blueprint just from **LWE**?

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Watermarked PRF implements PRF at all but a single point

Structurally very similar to a puncturable PRF [BW13, BGI13, KPTZ13]

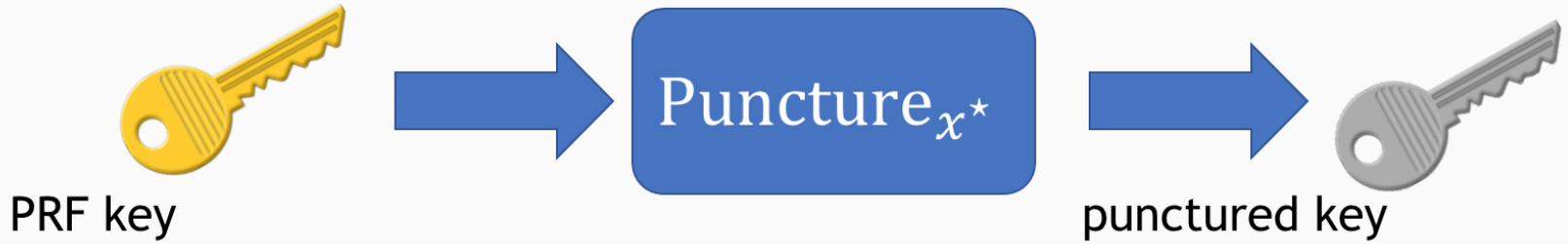
Puncturable PRF:



Punctured key can be used to evaluate PRF on all points $\mathbf{x} \neq \mathbf{x}^\star$

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Puncturable PRF:

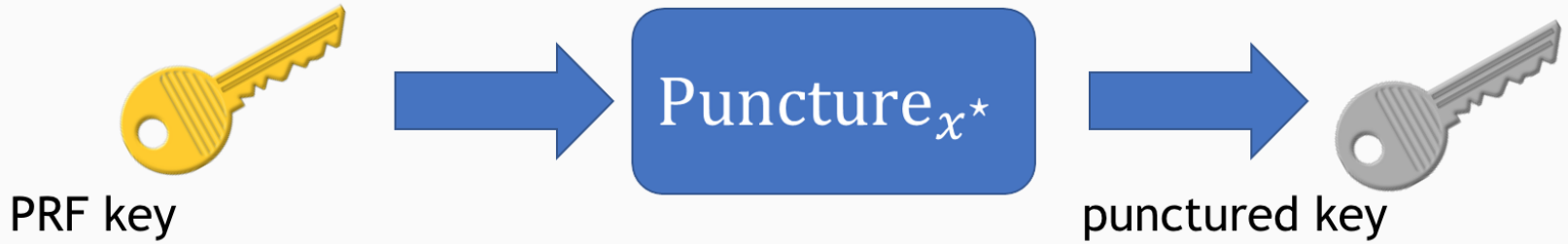


Recall general approach for watermarking:

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

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Puncturable PRF:



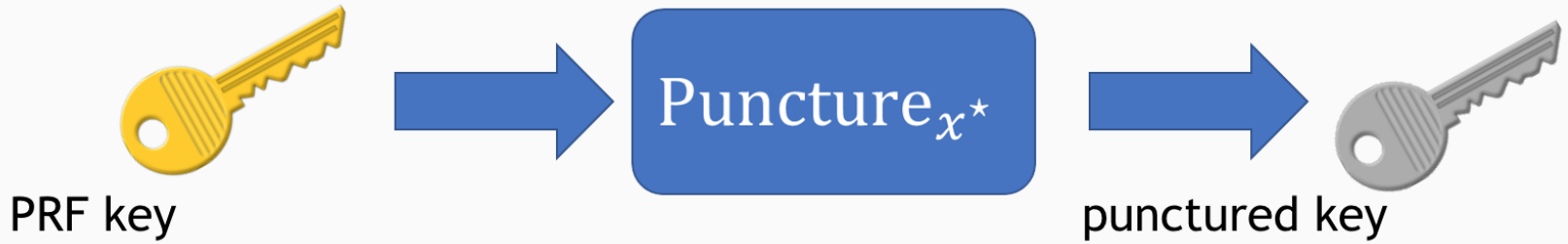
Recall general approach for watermarking:

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

Problem 1: Punctured key does not necessarily hide $\mathbf{x}^\star$, which allows adversary to remove watermark

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Puncturable PRF:



Recall general approach for watermarking:

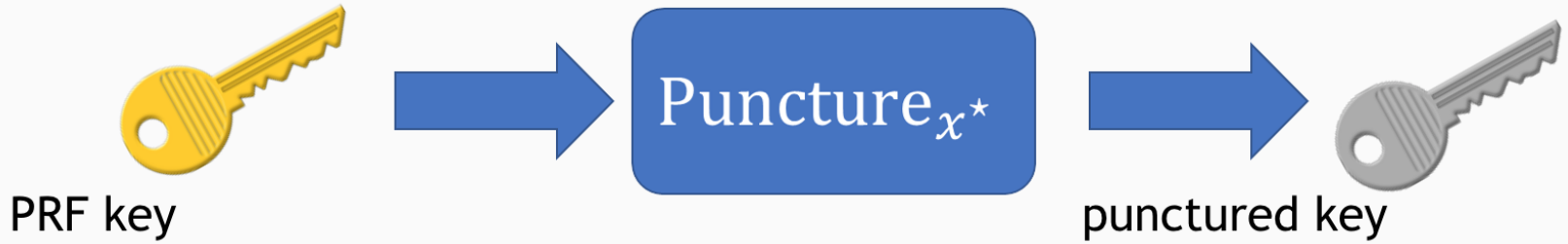
1. Derive $(\mathbf{x}^\star, \mathbf{y}^\star)$ from input/output behavior of PRF
2. Give out a key that agrees with PRF everywhere, except has value $\mathbf{y}^\star$ at $\mathbf{x} = \mathbf{x}^\star$

Problem 1: Punctured key does not necessarily hide $\mathbf{x}^\star$, which allows adversary to remove watermark

Problem 2: Punctured keys typically do not provide flexibility in programming value at punctured point

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Puncturable PRF:

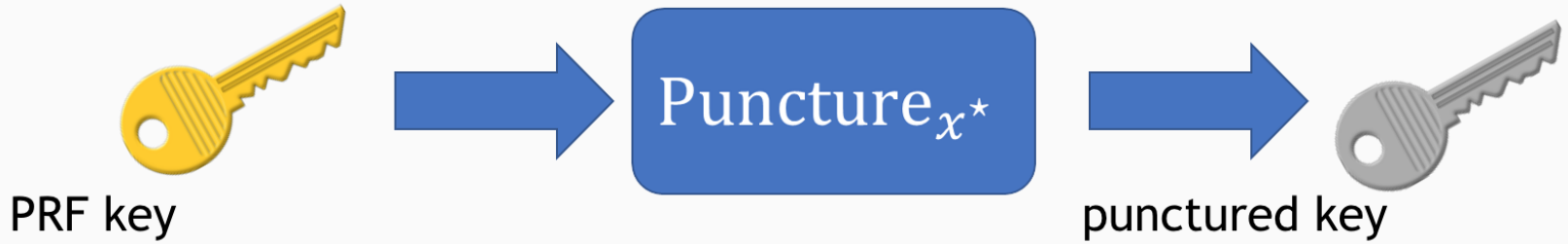


Problem 1: Punctured key does not necessarily hide $\mathbf{x}^\star$, which allows adversary to remove watermark

- Use private puncturable PRFs

Private Puncturable PRFs [BLW17, BKM17, CC17, BTVW17]

Puncturable PRF:



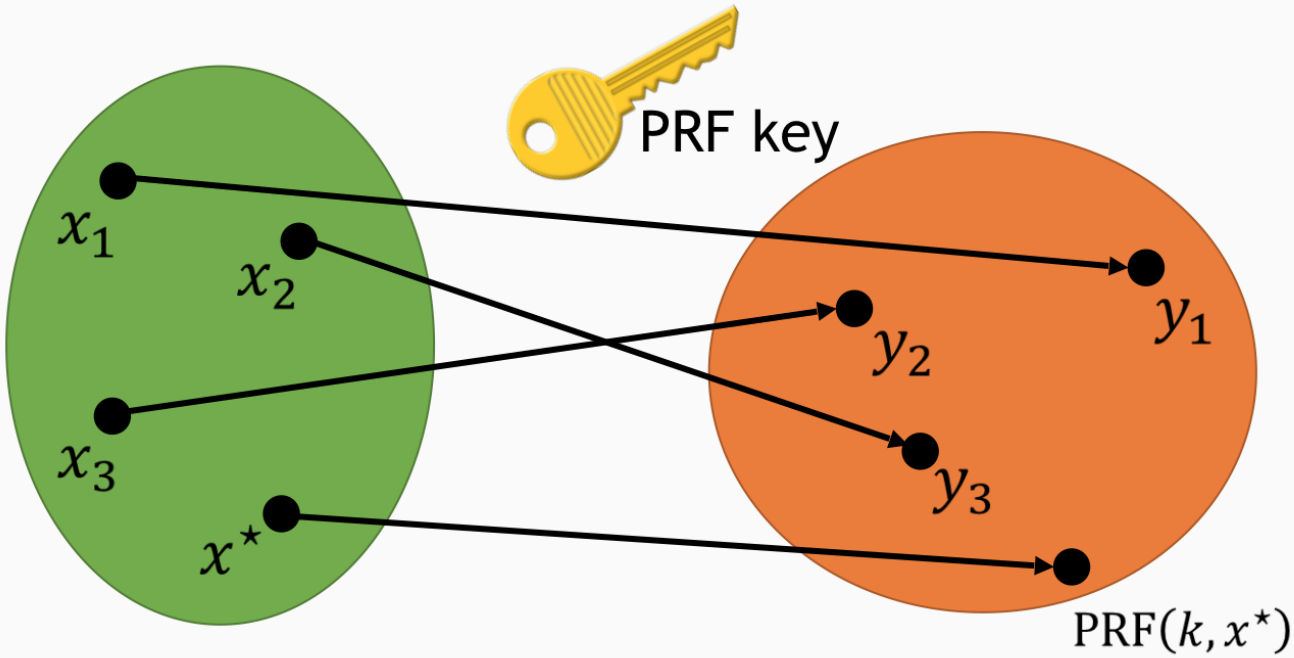
Problem 1: Punctured key does not necessarily hide $\mathbf{x}^\star$, which allows adversary to remove watermark

- Use private puncturable PRFs

Problem 2: Punctured keys typically do not provide flexibility in programming value at punctured point

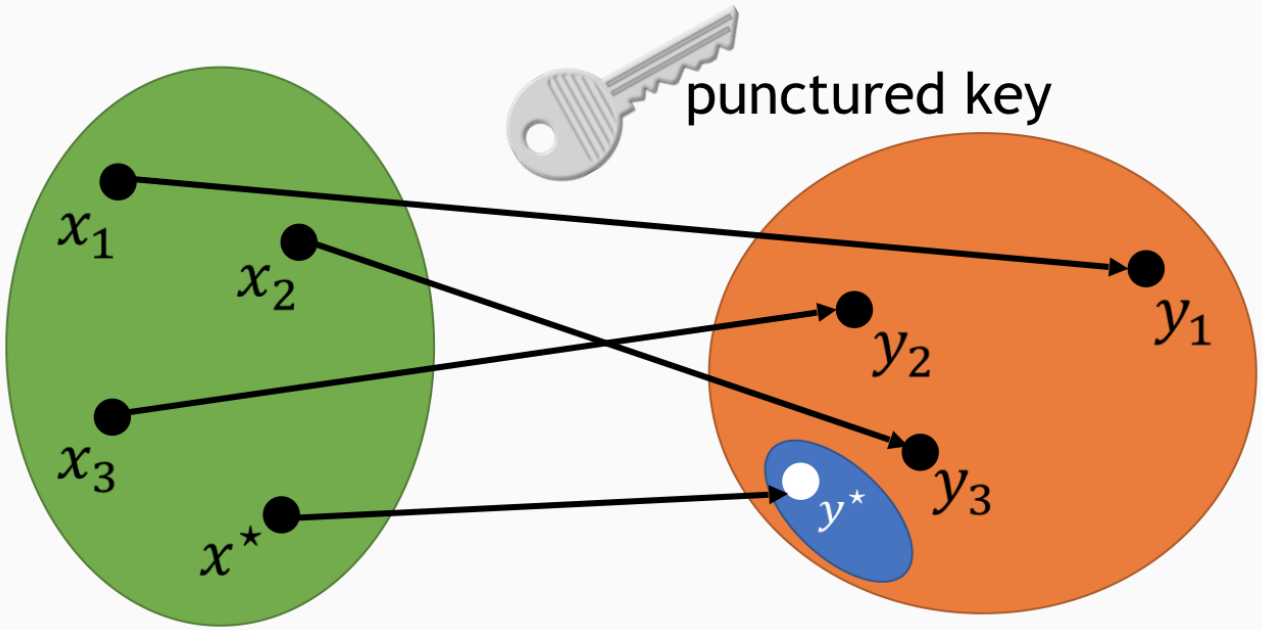
- Relax programmability requirement

Private Translucent PRFs



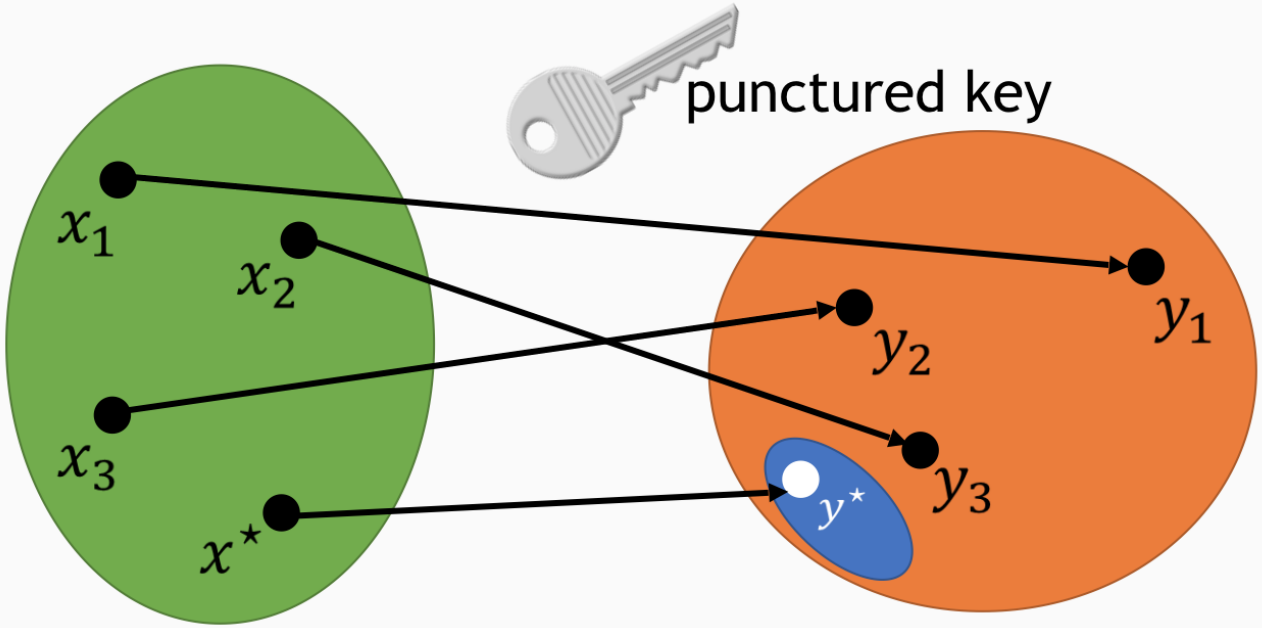
Output of any punctured key on a punctured point lies in a sparse, hidden subspace

Private Translucent PRFs



Output of any punctured key on a punctured point lies in a sparse, hidden subspace

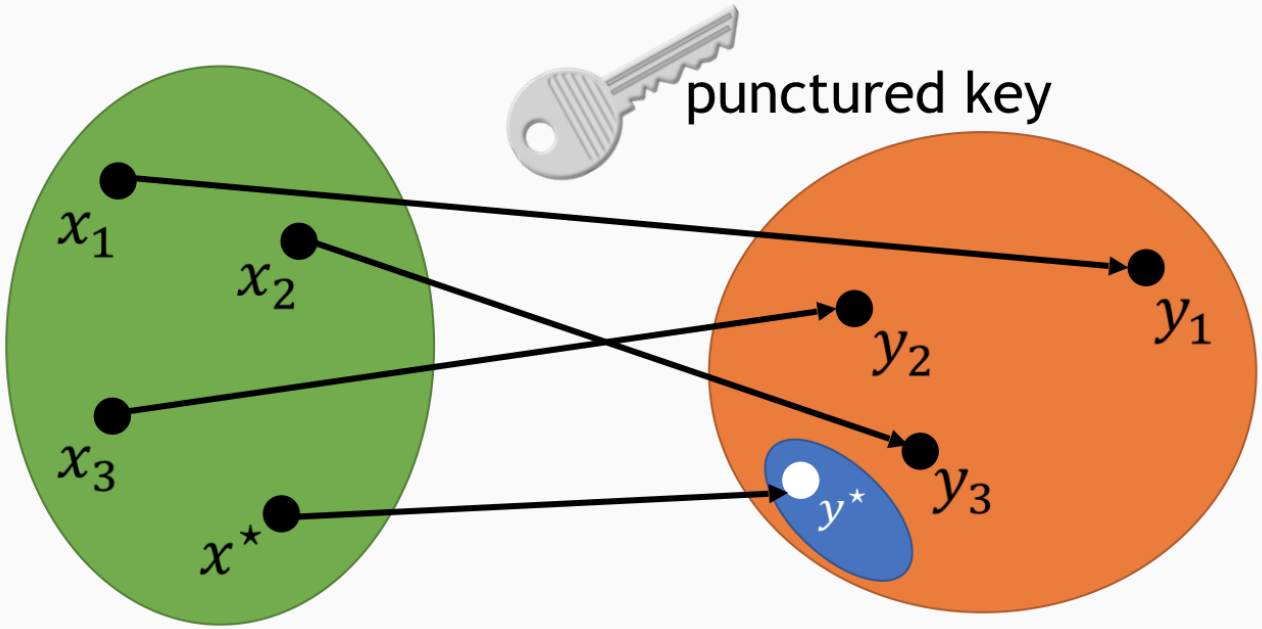
Private Translucent PRFs



Output of any punctured key on a punctured point lies in a sparse, hidden subspace

Secret testing key can be used to test for memership in the hidden subspace

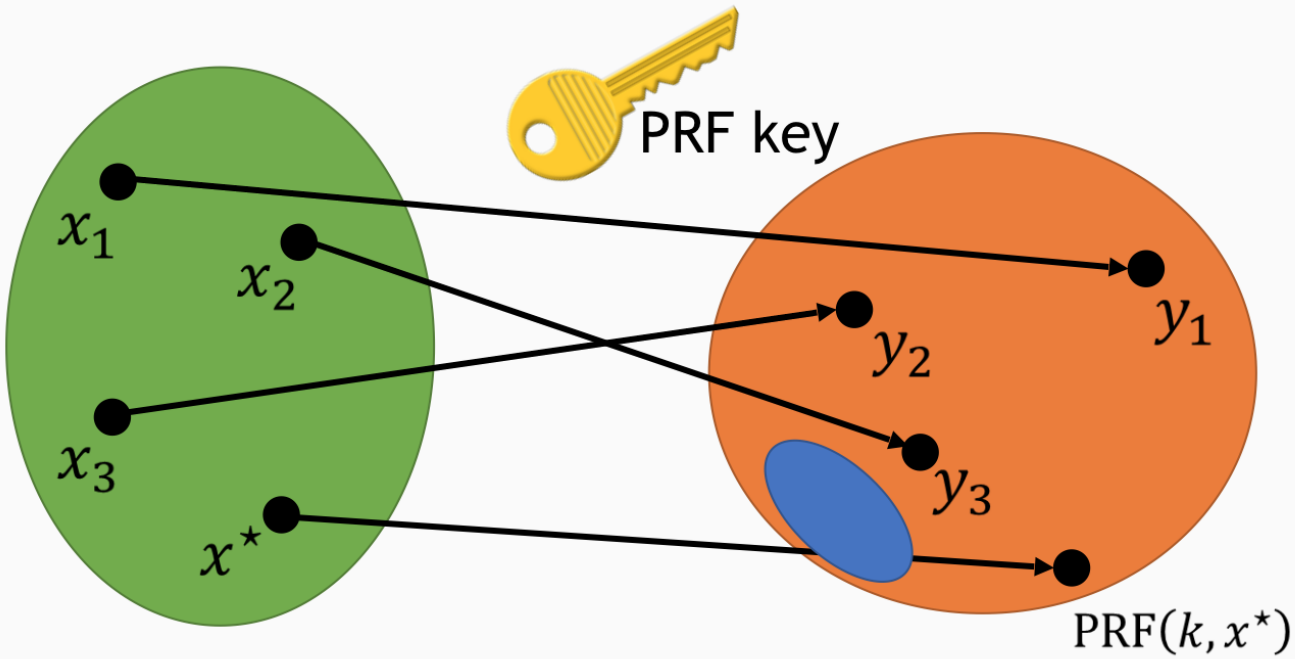
Private Translucent PRFs



Sets satisfying such properties are called translucent [CDNO97]

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

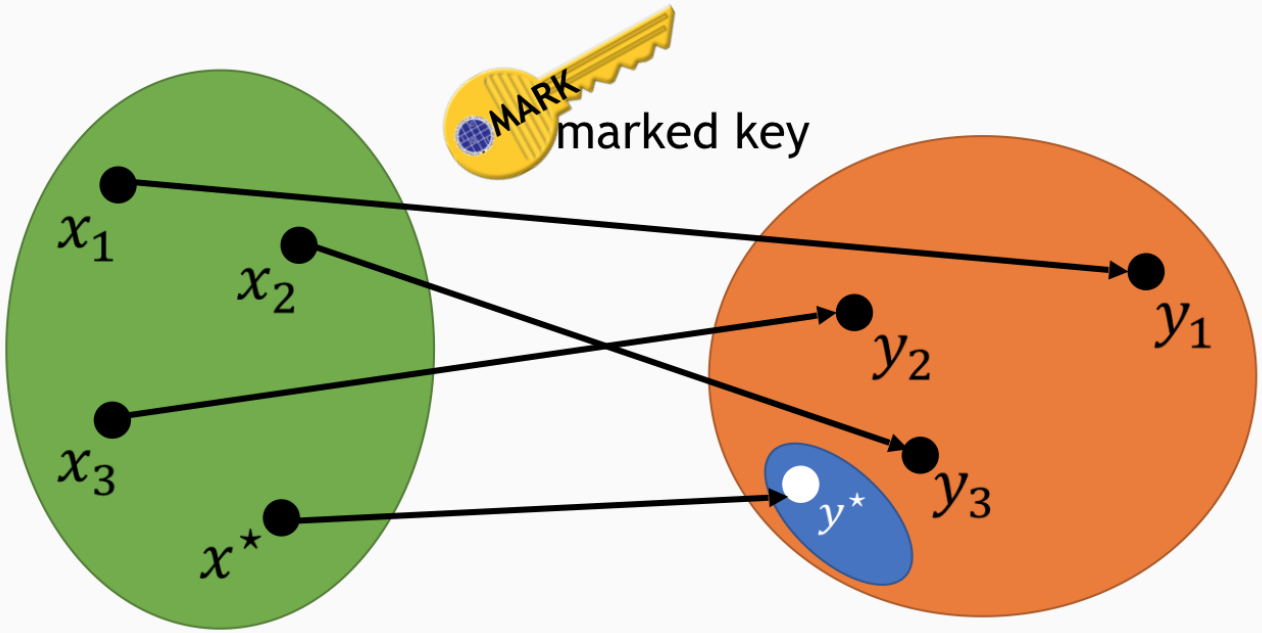
Private Translucent PRFs



Define watermarking secret key (wsk):

- Test points $\mathbf{x}_1, \ldots, \mathbf{x}_d$
- Testing key for private translucent PRF

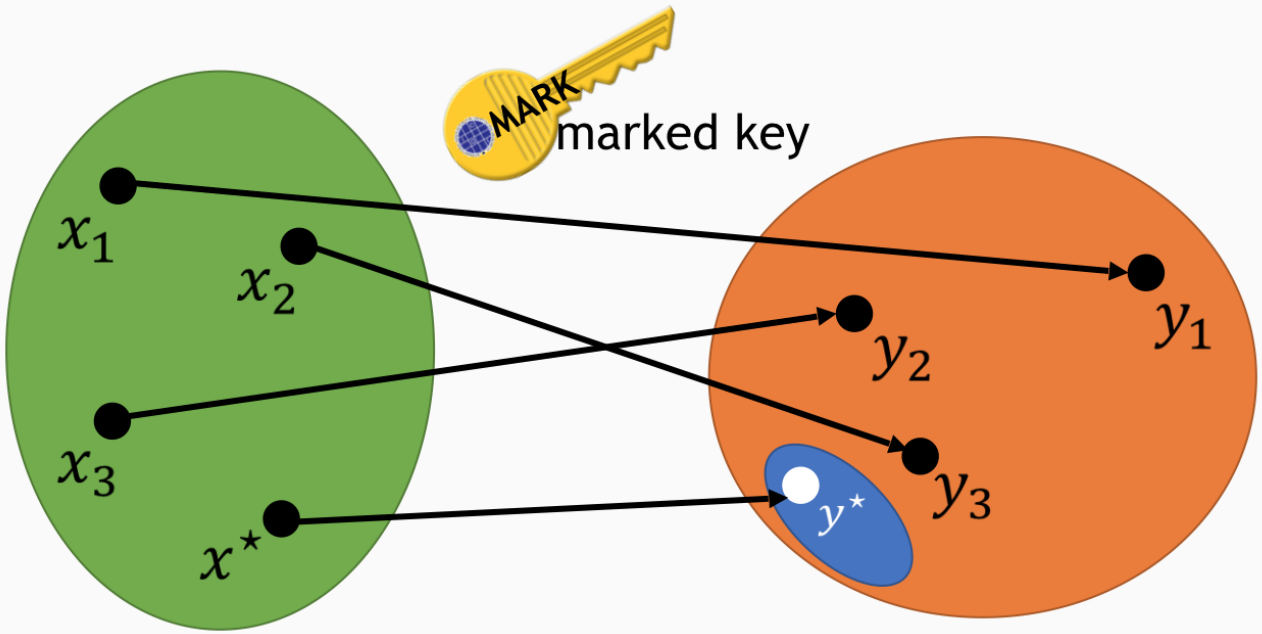
Private Translucent PRFs



To mark a PRF key k: derive special point $\mathbf{x}^\star$ and puncture k at $\mathbf{x}^\star$

Watermarked program is a program C' that evaluates punctured key

Private Translucent PRFs



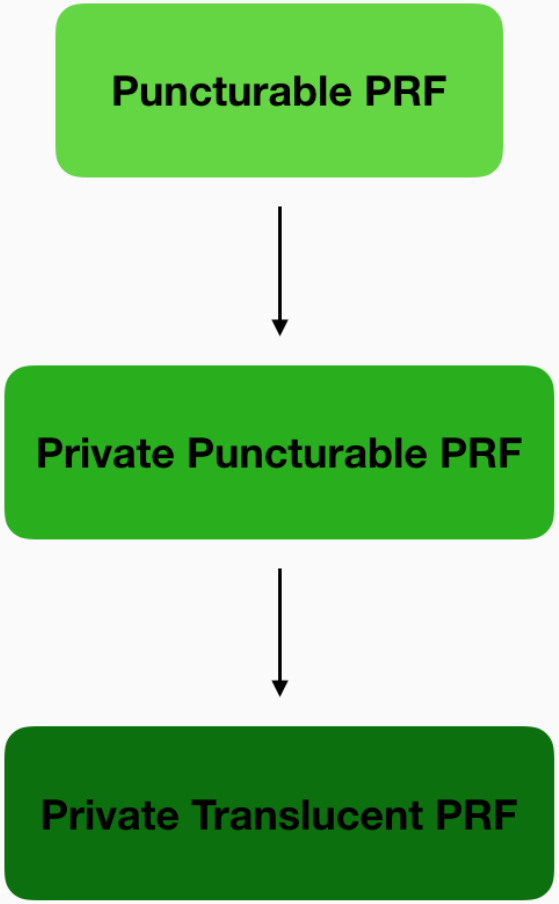
To mark a PRF key k: derive special point $\mathbf{x}^\star$ and puncture k at $\mathbf{x}^\star$

To test a program C':

1. Derive test point $\mathbf{x}^\star$
2. Check whether $C'(\mathbf{x}^\star)$ is in the translucent set

Constructing Private Translucent PRFs from Lattices

Road Map


Puncturable PRF
Private Puncturable PRF
Private Translucent PRF

Learning with Errors [Reg05]

Parameters: n, m, q, χ

Problem: Distinguish the following distributions

$$(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T) \approx (\mathbf{A}, \mathbf{u}^T).$$

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

Learning with Rounding [BPR12]

Parameters: n, m, q, p

Problem: Distinguish the following distributions

$$(\mathbf{A}, \lfloor \mathbf{s}^T \mathbf{A} \rceil_p) \approx (\mathbf{A}, \lfloor \mathbf{u}^T \rceil_p).$$

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

Lattice PRFs [BPR12,...]

LWE says that

$$\lfloor \mathbf{s}^T \mathbf{A} \rceil_p \approx \lfloor \mathbf{u}^T \rceil_p .$$

Randomized PRF(?): For any $\mathbf{x} \in \mathcal{X}$, choose $\mathbf{A}_\mathbf{x} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and define

$$\mathrm{PRF}(k, \mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{A}_\mathbf{x} \rceil_p$$

Lattice PRFs [BPR12,...]

LWE says that

$$\lfloor \mathbf{s}^T \mathbf{A} \rceil_p \approx \lfloor \mathbf{u}^T \rceil_p .$$

Randomized PRF(?): For any $\mathbf{x} \in \mathcal{X}$, choose $\mathbf{A}_\mathbf{x} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and define

$$\mathrm{PRF}(k, \mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{A}_\mathbf{x} \rceil_p$$

Approach: Fix matrix $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times m}$ and set key $k = \mathbf{s} \in \mathbb{Z}_q^n$.

For $\mathbf{x} \in \mathcal{X}$, derive $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \to \mathbf{A}_\mathbf{x}$ and define

$$\mathrm{PRF}(k, \mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{A}_\mathbf{x} \rceil_p$$

Lattice PRFs [BPR12,...]

LWE says that

$$\lfloor \mathbf{s}^T \mathbf{A} \rceil_p \approx \lfloor \mathbf{u}^T \rceil_p .$$

Randomized PRF(?): For any $\mathbf{x} \in \mathcal{X}$, choose $\mathbf{A}_\mathbf{x} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and define

$$\mathrm{PRF}(k, \mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{A}_\mathbf{x} \rceil_p$$

Approach: Fix matrix $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times m}$ and set key $k = \mathbf{s} \in \mathbb{Z}_q^n$.

For $\mathbf{x} \in \mathcal{X}$, derive $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \to \mathbf{A}_\mathbf{x}$ and define

$$\mathrm{PRF}(k, \mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{A}_\mathbf{x} \rceil_p$$

Question: What is a suitable way to derive $\mathbf{A}_\mathbf{x}$?

Gadget matrix [MP12]

Gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$

There is an efficiently computable function $\mathbf{G}^{-1}(\cdot)$ such that:

- $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \to \{0,1\}^{m \times m}$
- for all $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$:

$$\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{C}) = \mathbf{C}$$

Implementation:

- $\mathbf{G}^{-1}$ is the *bit decomposition* function
- $\mathbf{G}$ consists of *powers-of-2*

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right]$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T((\mathbf{A}_i + \mathbf{A}_j) + (x_i + x_j) \cdot \mathbf{G}) + (\mathbf{e}_i + \mathbf{e}_j)^T\end{aligned}$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (x_i + x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (x_i + x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $x_i \cdot x_j$

$$\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot x_j - \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i)$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (x_i + x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $x_i \cdot x_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot x_j - \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i) \\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (x_i \cdot x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Matrix Embeddings [BGGHNSVV14]

Encode $x \in \{0,1\}$ with respect to matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$\mathbf{s}^T(\mathbf{A} + x \cdot \mathbf{G}) + \mathbf{e}^T$$

Addition: To compute $x_i + x_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (x_i + x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $x_i \cdot x_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot x_j - \left[\mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i) \\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (x_i \cdot x_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Useful: Matrices $\mathbf{A}_{(+)}$, $\mathbf{A}_{(\times)}$ are independent of x_i, x_j.

Matrix Embeddings [BGGHNSVV14]

Compute any circuit $f : \{0,1\}^\ell \to \{0,1\}$:

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

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + x_\ell \cdot \mathbf{G}) + \mathbf{e}_1$$

Matrix Embeddings [BGGHNSVV14]

Compute any circuit $f : \{0,1\}^\ell \to \{0,1\}$:

$$\begin{array}{c} \mathbf{s}^T(\mathbf{A}_1 + x_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + x_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{array} \quad \stackrel{f}{\Longrightarrow} \quad \mathbf{s}^T(\mathbf{A}_f + f(\mathbf{x}) \cdot \mathbf{G}) + \text{noise}$$

Matrix Embeddings [BGGHNSVV14]

Compute any circuit $f : \{0,1\}^\ell \to \{0,1\}$:

$$\begin{array}{c} \mathbf{s}^T(\mathbf{A}_1 + x_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + x_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{array} \quad \stackrel{f}{\Longrightarrow} \quad \mathbf{s}^T(\mathbf{A}_f + f(\mathbf{x}) \cdot \mathbf{G}) + \text{noise}$$

Useful property:
Matrix $\mathbf{A}_f$ does not depend on encoded values $x_1, \ldots, x_\ell$

$\Rightarrow$ Can derive $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \to \mathbf{A}_f$ without knowing $x_1, \ldots, x_\ell$.

Puncturable PRF [BV14]

Idea: Define PRF with respect to $\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$:

$$\mathsf{PRF}(k, \mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p .$$

Puncturable PRF [BV14]

Idea: Define PRF with respect to $\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$:

$$\mathsf{PRF}(k, \mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p .$$

Define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{c} \mathbf{s}^T(\mathbf{A}_1 + c_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + c_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{array}$$

Puncturable PRF [BV14]

Idea: Define PRF with respect to $\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$:

$$\mathsf{PRF}(k, \mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p .$$

Define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{c} \mathbf{s}^T(\mathbf{A}_1 + c_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + c_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{array} \quad \overset{\mathsf{eq}_{\mathbf{x}}}{\Longrightarrow} \quad \mathbf{s}^T(\mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}) \cdot \mathbf{G}) + \mathsf{noise}$$

Puncturable PRF [BV14]

Idea: Define PRF with respect to $\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$:

$$\mathsf{PRF}(k, \mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p .$$

Define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{matrix} \mathbf{s}^T(\mathbf{A}_1 + c_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + c_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{matrix} \quad \xRightarrow{\mathsf{eq}_{\mathbf{x}}} \quad \mathbf{s}^T(\mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}) \cdot \mathbf{G}) + \text{noise}$$

When $\mathbf{x} \neq \mathbf{c}$, then $\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = 0$:

$$\left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} + \text{noise} \right\rceil_p = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p$$

When $\mathbf{x} = \mathbf{c}$,

$$\left\lfloor \mathbf{s}^T (\mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} + \mathbf{G}) + \text{noise} \right\rceil_p \neq \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{eq}_{\mathbf{x}}} \right\rceil_p$$

Private Constrained PRFs [BKM17, BTVW17]

Hide punctured point?

Private Constrained PRFs [BKM17, BTVW17]

Hide punctured point? Recall...

Addition: To compute $c_i + c_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right]\\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (c_i + c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $c_i \cdot c_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot c_j - \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i)\\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (c_i \cdot c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Private Constrained PRFs [BKM17, BTVW17]

Hide punctured point? Recall...

Addition: To compute $c_i + c_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (c_i + c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $c_i \cdot c_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot c_j - \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i) \\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (c_i \cdot c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Private Puncturable PRFs [BKM17, BTVW17]

Hide punctured point? Recall...

Addition: To compute $c_i + c_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (c_i + c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $c_i \cdot c_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot c_j - \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i) \\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (c_i \cdot c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Punctured key must include $\mathbf{c}$ in order to compute!

Private Puncturable PRFs [BKM17, BTVW17]

Hide punctured point? Recall...

Addition: To compute $c_i + c_j$:

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] + \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \\ &= \mathbf{s}^T(\mathbf{A}_{(+)} + (c_i + c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Multiplication: To compute $c_i \cdot c_j$

$$\begin{aligned}&\left[\mathbf{s}^T(\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^T\right] \cdot c_j - \left[\mathbf{s}^T(\mathbf{A}_j + c_j \cdot \mathbf{G}) + \mathbf{e}_j^T\right] \cdot \mathbf{G}^{-1}(\mathbf{A}_i) \\ &= \mathbf{s}^T(\mathbf{A}_{(\times)} + (c_i \cdot c_j) \cdot \mathbf{G}) + \text{noise}\end{aligned}$$

Punctured key must include $\mathbf{c}$ in order to compute!

Idea: Use FHE to hide encoded values [GVW15]

Fully Homomorphic Encryption

For message $x_1, \ldots, x_\ell$ and circuit $f : \{0,1\}^\ell \to \{0,1\}$, compute only on ciphertext

$$\left(\ \mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_\ell) \ , \ f \ \right) \Longrightarrow \mathsf{Enc}(f(\mathbf{x}))$$

Fully Homomorphic Encryption

For message $x_1, \ldots, x_\ell$ and circuit $f : \{0,1\}^\ell \to \{0,1\}$, compute only on ciphertext

$$\left(\ \mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_\ell) \ , \ f \ \right) \Longrightarrow \mathsf{Enc}(f(\mathbf{x}))$$

For LWE-based constructions, $\mathsf{Enc}(f(x)) \in \mathbb{Z}_q^m$, $\mathsf{sk} \in \mathbb{Z}_q^m$, and decryption is an inner product

$$\left\langle \ \mathsf{Enc}(f(\mathbf{x})) \ , \ \mathsf{sk} \ \right\rangle = \frac{q}{2} \cdot f(\mathbf{x}) + \text{noise}.$$

Fully Homomorphic Encryption

For message $x_1, \ldots, x_\ell$ and circuit $f : \{0,1\}^\ell \to \{0,1\}$, compute only on ciphertext

$$\Big(\ \mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_\ell) \ , \ f \ \Big) \Longrightarrow \mathsf{Enc}(f(\mathbf{x}))$$

For LWE-based constructions, $\mathsf{Enc}(f(x)) \in \mathbb{Z}_q^m$, $\mathsf{sk} \in \mathbb{Z}_q^m$, and decryption is an inner product

$$\Big\langle \ \mathsf{Enc}(f(\mathbf{x})) \ , \ \mathsf{sk} \ \Big\rangle = f(\mathbf{x}).$$

(cheating for simplicity)

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1$$

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{ccc} \mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},1} + \tilde{\mathsf{ct}}_1 \cdot \mathbf{G}) + \mathsf{noise} \\ \vdots & \overset{\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}}{\Longrightarrow} & \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},\ell} + \tilde{\mathsf{ct}}_m \cdot \mathbf{G}) + \mathsf{noise} \end{array}$$

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{ccc} \mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},1} + \tilde{\mathsf{ct}}_1 \cdot \mathbf{G}) + \mathsf{noise} \\ \vdots & \overset{\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}}{\Longrightarrow} & \vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},\ell} + \tilde{\mathsf{ct}}_m \cdot \mathbf{G}) + \mathsf{noise} \end{array}$$

We have: $\tilde{\mathsf{ct}} = \mathsf{Enc}(\mathsf{eq}_{\mathbf{x}}(\mathbf{c}))$

How do we **extract** this value?

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{ccc}
\mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},1} + \tilde{\mathsf{ct}}_1 \cdot \mathbf{G}) + \mathsf{noise} \\
\vdots & \overset{\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}}{\Longrightarrow} & \vdots \\
\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1 & & \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x},\ell} + \tilde{\mathsf{ct}}_m \cdot \mathbf{G}) + \mathsf{noise}
\end{array}$$

$$\begin{array}{c}
\mathbf{s}^T(\mathbf{B}_1 + \mathsf{sk}_1 \cdot \mathbf{G}) + \mathbf{e}'_1 \\
\vdots \\
\mathbf{s}^T(\mathbf{B}_\ell + \mathsf{sk}_m \cdot \mathbf{G}) + \mathbf{e}'_m
\end{array}$$

Give out ct, but not $\mathsf{sk}_1, \ldots, \mathsf{sk}_m$!

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1$$

$$\xrightarrow{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x}} + \langle \mathsf{sk}, \tilde{\mathsf{ct}} \rangle \cdot \mathbf{G}) + \text{noise}$$

$$\mathbf{s}^T(\mathbf{A}_1 + \mathsf{sk}_1 \cdot \mathbf{G}) + \mathbf{e}'_1$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{sk}_m \cdot \mathbf{G}) + \mathbf{e}'_m$$

Give out ct, but not $\mathsf{sk}_1, \ldots, \mathsf{sk}_m$!

Private Puncturable PRF [BKM17, BTVW17]

First encrypt $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then define punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1$$

$$\xRightarrow{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} \mathbf{s}^T(\mathbf{A}^*_{\mathbf{x}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}) \cdot \mathbf{G}) + \mathsf{noise}$$

$$\mathbf{s}^T(\mathbf{A}_1 + \mathsf{sk}_1 \cdot \mathbf{G}) + \mathbf{e}'_1$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_\ell + \mathsf{sk}_m \cdot \mathbf{G}) + \mathbf{e}'_m$$

Give out ct, but not $\mathsf{sk}_1, \ldots, \mathsf{sk}_m$!

Private Puncturable PRF [BKM17, BTVW17]

First $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{c})$.

Then punctured key $\mathsf{sk}_{\mathbf{c}}$:

$$\begin{array}{l} \mathbf{s}^T(\mathbf{A}_1 + \mathsf{ct}_1 \cdot \mathbf{G}) + \mathbf{e}_1 \\ \qquad\vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + \mathsf{ct}_\ell \cdot \mathbf{G}) + \mathbf{e}_1 \end{array} \quad \overset{\mathsf{IP}\circ\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}}{\Longrightarrow} \quad \mathbf{s}^T(\mathbf{A}_{\mathsf{IP}\circ\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}) \cdot \mathbf{G}) + \mathsf{noise}$$

$$\begin{array}{l} \mathbf{s}^T(\mathbf{A}_1 + \mathsf{sk}_1 \cdot \mathbf{G}) + \mathbf{e}'_1 \\ \qquad\vdots \\ \mathbf{s}^T(\mathbf{A}_\ell + \mathsf{sk}_m \cdot \mathbf{G}) + \mathbf{e}'_m \end{array}$$

Define PRF with respect to $\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}$: $\mathsf{PRF}(k, \mathbf{x}) = \left\lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{IP}\circ\mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} \right\rceil_p$

Programming

Encode punctured point $\mathbf{c}$ (under FHE)

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}) \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $\mathbf{c}$ (under FHE)

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for some $w \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for $w \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for $w \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + \mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for $w \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + w \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for $w \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\underbrace{\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + w \cdot \mathbf{G}}) + \text{noise}$$

There is some programmability here!

We can choose w at key generation.

Programming

Encode punctured point $(\mathbf{c}, w)$ (under FHE) for $w_i \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + w \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w_1, \ldots, w_N)$ (under FHE) for $w_i \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x}}(\mathbf{c}, w) = \begin{cases} w & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}}}} + w \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w_1, \ldots, w_N)$ (under FHE) for $w_i \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x},i}(\mathbf{c}, w_i) = \begin{cases} w_i & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_\mathbf{x}}} + w \cdot \mathbf{G}) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w_1, \ldots, w_N)$ (under FHE) for $w_i \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x},i}(\mathbf{c}, w_i) = \begin{cases} w_i & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x},1}}} + w_1 \cdot \mathbf{G}_1) + \text{noise}$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x},N}}} + w_N \cdot \mathbf{G}_N) + \text{noise}$$

Programming

Encode punctured point $(\mathbf{c}, w_1, \ldots, w_N)$ (under FHE) for $w_i \in \mathbb{Z}_q$

Define PRF with respect to equality circuit:

$$\mathsf{eq}_{\mathbf{x},i}(\mathbf{c}, w_i) = \begin{cases} w_i & \mathbf{x} = \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Constrained key evaluation at $\mathbf{c}$:

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x},1}}} + w_1 \cdot \mathbf{G}_1) + \text{noise}$$

$$\vdots$$

$$\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x},N}}} + w_N \cdot \mathbf{G}_N) + \text{noise}$$

Choose $w_1, \ldots, w_N$ such that

$$\mathbf{D} = \sum_{i \in [N]} \mathbf{A}_{\mathsf{IP} \circ \mathsf{Eval}_{\mathsf{eq}_{\mathbf{x}_i}}} + \sum_{i \in [N]} w_i \cdot \mathbf{G}_i$$

Programming

Constrained key evaluation at $\mathbf{c}$:

$$\left\lfloor \mathbf{s}^T \mathbf{D} \right\rceil_p$$

Watermark key includes short vector $\mathbf{w}$ such that

$$\mathbf{D} \cdot \mathbf{w} = \mathbf{0}$$

To Conclude...

Open Problems:

- Achieve stronger security guarantees?
- Watermark other primitives?

To Conclude...

Open Problems:

- Achieve stronger security guarantees?
- Watermark other primitives?

Thanks!