Lattice-based PRFs and Constrained PRFs

Xinyu Mao

November 13, 2021

Shanghai Jiao Tong Uninversity

Definition of Pseudorandom Functions (PRFs)

Definition 1 (Keyed function)

Let κ be a security parameter. A *keyed function* with domain $\mathcal{D} := \{\mathcal{D}_\kappa\}_{\kappa \in \mathbb{N}}$ and range $\mathcal{R} := \{\mathcal{R}_\kappa\}_{\kappa \in \mathbb{N}}$ is a pair of PPT algorithms $(\mathsf{Gen}, \mathsf{Eval})$ where

- $\mathsf{Gen}(1^\kappa) \mapsto K \in \{0,1\}^\kappa$.
- $\mathsf{Eval}(K, x) \mapsto y \in \mathcal{R}_\kappa$: The evaluation algorithm takes as input $x \in \mathcal{D}_\kappa$ and outputs $y \in \mathcal{R}_\kappa$.

Definition 2 (PRF)

A keyed function $\Pi := (\mathsf{Gen}, \mathsf{Eval})$ is a *PRF* if for every PPT adversary $\mathcal{A}$, the following quantity is negligible:

$$\left| \Pr_{K \leftarrow \mathsf{Gen}(1^\kappa)} \left[\mathcal{A}^{\mathsf{Eval}(K,\cdot)}(1^\kappa) = 1 \right] - \Pr_{f \xleftarrow{\$} \mathcal{F}} \left[\mathcal{A}^{f(\cdot)}(1^\kappa) = 1 \right] \right|,$$

where $\mathcal{F}$ is the set of all functions from $\mathcal{D}_\kappa$ to $\mathcal{R}_\kappa$.

The Construction in [BPR12]

Construction 1

- *Public parameters: moduli* $q > p$.
- $\mathcal{D} := \{0,1\}^{\ell}, \mathcal{R} := \mathbb{Z}_p^n$.
- $\mathsf{Gen}(1^{\kappa}) \mapsto K$: *Sample* $\mathsf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ *and* $\mathsf{S}_i \leftarrow \chi^{n \times n}$ *for each* $i \in \ell$. *Output* $K := \left(\mathsf{a}, \{\mathsf{S_i}\}_{i \in [\ell]}\right)$.
- $\mathsf{Eval}(K, x) \mapsto y$: *Parse* $K := \left(\mathsf{a}, \{\mathsf{S_i}\}_{i \in [\ell]}\right)$ *and output*

$$F_{\mathsf{a},\mathsf{s}_1,\ldots,\mathsf{s}_\ell}(x) := \left\lfloor \mathsf{a}^{\top} \cdot \prod_{i=1}^{\ell} \mathsf{S}_i^{x_i} \right\rceil_p \in \mathbb{Z}_p^n.$$

Proof Outline

- Replace $F_{\mathbf{a},\mathbf{S}_1,\ldots,\mathbf{S}_\ell}(x)$ with

$$\widetilde{F}_{\mathbf{a},\mathbf{S}_1,\ldots,\mathbf{S}_\ell}(x) := \left\lfloor (\mathbf{a}^\top \mathbf{S}_1^{x_1} + x_1 \cdot \mathbf{e}_{x_1}^\top) \cdot \prod_{i=2}^{\ell} \mathbf{S}_i^{x_i} \right\rceil_p$$

$$= \left\lfloor \mathbf{a}^\top \prod_{i=1}^{\ell} \mathbf{S}_1^{x_i} + x_1 \cdot \mathbf{e}_{x_1}^\top \cdot \prod_{i=2}^{\ell} \mathbf{S}_i^{x_i} \right\rceil_p .$$

- Since the error term is small, after rounding, $\widetilde{F}(x) = F(x)$ on all queries w.h.p..
- Replace $(\mathbf{a}, \mathbf{a}^\top \mathbf{S}_1 + \mathbf{e}_{x_1}^\top)$ with uniform $(\mathbf{u}_0, \mathbf{u}_1)$. That is, we now output

$$F'_{\mathbf{a},\mathbf{S}_1,\ldots,\mathbf{S}_\ell}(x) := \left\lfloor \mathbf{u}_{x_1} \cdot \prod_{i=2}^{\ell} \mathbf{S}_i^{x_i} \right\rceil_p .$$

- Repeat for $\mathbf{S}_2, \ldots, \mathbf{S}_\ell$, we get $F''''(x) = \lfloor \mathbf{u}_x \rceil_p$, which is a uniformly random function.

Key-Homomorphic Construction [BLMR13]

Construction 2

- *Public parameters:* $\mathsf{B}_0, \mathsf{B}_1 \xleftarrow{\$} \{0,1\}^{m \times m}$ *and moduli* $q > p$.
- $\mathcal{D} := \{0,1\}^{\ell}, \mathcal{R} := \mathbb{Z}_p^m$.
- $\mathsf{Gen}(1^{\kappa}) \mapsto K \in \mathbb{Z}_q^m$: *Sample* $\mathsf{s} \xleftarrow{\$} \mathbb{Z}_q^m$ *and output* $K := \mathsf{s}$.
- $\mathsf{Eval}(\mathsf{s}, x \in \{0,1\}^{\ell})$: *Output*

$$F_{\mathsf{s}}(x) := \left\lfloor \mathsf{s}^{\top} \prod_{i=1}^{\ell} \mathsf{B}_{x_i} \right\rceil_p \in \mathbb{Z}_p^m.$$

- Almost key-homomorphic:

$$F_{\mathsf{s}_1+\mathsf{s}_2}(x) = F_{\mathsf{s}_1}(x) + F_{\mathsf{s}_2}(x) + \{-1,0,1\}^m .$$

- The proof strategy is similar to [BPR12]: introduce short errors that vanishes after rounding.

Proof Outline [BLMR13]

$$F_{\mathbf{s}}(x) := \left\lfloor \mathbf{s}^\top \prod_{i=1}^{\ell} \mathbf{B}_{x_i} \right\rfloor_p \approx_s \left\lfloor (\mathbf{s}^\top \mathbf{B}_{x_1} + \mathbf{e}_{x_1}) \cdot \prod_{i=2}^{\ell} \mathbf{B}_{x_i} \right\rfloor_p$$

$$\approx_c \left\lfloor \mathbf{u}_{x_1} \cdot \prod_{i=2}^{\ell} \mathbf{B}_{x_i} \right\rfloor_p \approx_c \cdots \approx_c \lfloor \mathbf{u}_x \rceil_p = U(x).$$

- Note that the public matrix $\mathbf{B}_0, \mathbf{B}_1$ is sampled from $\{0,1\}^{m \times m}$ (not $\mathbb{Z}_q^{n \times n}$). This guarantees the error we introduced will not be amplified when multiplied by $\mathbf{B}_i$.
- By setting $m \approx n \log q$, this can be reduced to the standard LWE with dimension n.

✗ LWE approx factor α grows exponentially in input length ℓ.

Gadget Trapdoors, Rewind

Recall that the *gadget matrix* is defined as

$$\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g} \in \mathbb{Z}_q^{n \times n\ell},$$

where $\ell = \lceil \log q \rceil$ and $\mathbf{g} := (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{\ell}$.

- If $\mathbf{x} \in \{0,1\}^{\ell}$ is the binary decomposition of $u \in \mathbb{Z}_q$, we have $\langle \mathbf{g}, \mathbf{x} \rangle = u$.
- View $\mathbf{x} \in \{0,1\}^{n\ell}$ as n blocks: $\mathbf{x} = (\mathbf{x}_{\{1\}}, \ldots, \mathbf{x}_{\{n\}})$, where each block has length ℓ, i.e., $\mathbf{x}_{\{i\}} \in \{0,1\}^{\ell}$. Then $\mathbf{G}\mathbf{x} = \mathbf{u} \in \mathbb{Z}_q^n$ simply says: $\mathbf{x}_{\{i\}}$ is the binary decomposition of $\mathbf{u}_i$.
- $\mathbf{G}^{-1}$ is the "decomposition" function defined as:

$$\begin{aligned} \mathbf{G}^{-1} : \mathbb{Z}_q^n &\to \mathbb{Z}^{n\ell} \\ \mathbf{u} &\mapsto \text{a short } \mathbf{x} \text{ such that } \mathbf{G}\mathbf{x} = \mathbf{u}. \end{aligned}$$

[BP14]: A Tree Enjoys Better Parameter :)

Construction 3

- *Public parameters:* $\mathsf{A}_0, \mathsf{A}_1 \in \mathbb{Z}_q^{n \times n\ell}$, *a binary tree T, and a moduli* $q \geq p$.
- $\mathcal{D} := \{0,1\}^{|T|}, \mathcal{R} := \mathbb{Z}_p^{n\ell}$, *where* $|T|$:= *number of leaves in T.*
- $\mathsf{Gen}(1^\kappa) \to K \in \mathbb{Z}_q^n$: *Sample* $\mathsf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ *and output* s.
- $\mathsf{Eval}(\mathsf{s}, x) \to y$: *Output*

$$\left\lfloor \mathsf{s}^\top \cdot \mathsf{A}_T(x) \right\rceil \in \mathbb{Z}_p^{n\ell}.$$

$\mathsf{A}_T : \{0,1\}^{|T|} \to \mathbb{Z}_q^{n \times n\ell}$ is defined recursively as

$$\mathsf{A}_T(a) := \begin{cases} \mathsf{A}_x & \text{if } |T| = 1, \\ \mathsf{A}_{T.l}(x.l) \cdot \mathsf{G}^{-1}(\mathsf{A}_{T.r}(x.r)), & \text{otherwise,} \end{cases}$$

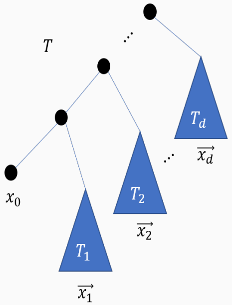
where we parse $x := x.l \| x.r$ for $x.l \in \{0,1\}^{|T.l|}, x.r \in \{0,1\}^{|T.r|}$.

$F_{\mathsf{s}}(x) := \left\lfloor \mathsf{s}^\top \cdot \mathsf{A}_T(x) \right\rceil \in \mathbb{Z}_p^{n\ell}$ where

$$\mathsf{A}_T(a) := \begin{cases} \mathsf{A}_x & \text{if } |T| = 1, \\ \mathsf{A}_{T.l}(x.l) \cdot \mathsf{G}^{-1}(\mathsf{A}_{T.r}(x.r)), & \text{otherwise.} \end{cases}$$

- Sequentiality $s(T)$ (the “right depth” of T): Circuit depth of PRF is proportional to $s(T)$.
- Expansion $e(T)$ (the “left depth” of T): LWE approx factor is exponential in $e(T)$.
- Max input length = max number of leaves = $\binom{e+s}{e}$.

Proof Idea



Consider the leftmost path:

$$
\begin{aligned}
F_{\mathbf{s}}(x) &= \left\lfloor \mathbf{s}^\top \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{T_1}(\overrightarrow{x_1}))\cdots \right\rceil_p \\
&\approx_s \left\lfloor (\mathbf{s}^\top \mathbf{A}_{x_0} + \mathbf{e}_{x_0}) \cdot \mathbf{G}^{-1}(\mathbf{A}_{T_1}(\overrightarrow{x_1}))\cdots \right\rceil_p \\
&\approx_c \left\lfloor \mathbf{u}_{x_0}^\top \cdot \mathbf{G}^{-1}(\mathbf{A}_{T_1}(\overrightarrow{x_1}))\cdots \right\rceil_p . \ (*)
\end{aligned}
$$

- Problem: $\{\mathbf{A}_{T_1}(\overrightarrow{x_1})\}_{\overrightarrow{x_1}\in\{0,1\}^w}$ is not independent unless $w := |\overrightarrow{x_1}| = 1$.
- A wishful thinking: if $\mathbf{u}_{x_0}^\top = \mathbf{t}_{x_0}^\top \mathbf{G}$, then $(*) = \left\lfloor \mathbf{t}_{x_0}^\top \cdot \mathbf{A}_{T_1}(\overrightarrow{x_1})\cdots \right\rceil_p$.
- However, a uniformly random u is highly likely to be very far from any vector of the form $\mathbf{t}^\top \mathbf{G}$.

Proof Idea

Solution: Write $\mathbf{u}^\top = \mathbf{t}^\top \mathbf{G} + \mathbf{v}^\top$, where $\mathbf{v} \in \mathcal{P}(\mathbf{G})$ and $\mathbf{t}$ are uniform and independent.

$F_{\mathbf{s}}(x)$ is indistinguishable from

$$F'_{\mathbf{u}_0,\mathbf{u}_1,\mathbf{v}_0,\mathbf{v}_1}(x) = \left\lfloor \mathbf{t}_{x_0}^\top \cdot \mathbf{A}_{T'}(x_2\|\cdots\|x_\ell) + \mathbf{v}_{x_0}^\top \cdot \mathbf{G}^{-1}(\mathbf{A}_{T_1}(\overrightarrow{x_1}))\cdots\right\rceil_p ,$$

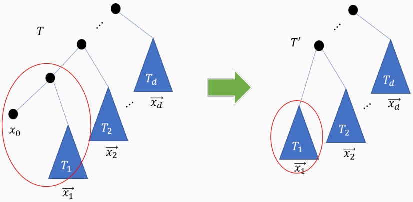


Figure 1: T' is the tree obtained from T by removing its leftmost leaf z and promoting z's sibling subtree T_1 to replace their parent.

Summary

The common idea in [BLMR13] and [BP14]

- Generate some matrices $\left\{\mathsf{A}_i \in \mathbb{Z}_q^{n\times m}\right\}_{i\in[k]}$ as public parameters.
- The key of the PRF is a vector $\mathbf{s} \in \mathbb{Z}_q^n$.
- To evaluate on the point $\mathbf{x} \in \{0,1\}^\ell$, one first compute a matrix $\mathsf{A}_x \in \mathbb{Z}_q^{n\times m}$ publicly, and output $F_{\mathbf{s}}(x) := \left\lfloor \mathbf{s}^\top \mathsf{A}_x \right\rceil_p$.

[BLMR13] can be view as a special case of [BP14] in the following sense:

- The [BLMR13] construction works as long as the public matrices $\mathsf{B}_0, \mathsf{B}_1$ is somewhat "short". Hence, we may generate $\mathsf{B}_0, \mathsf{B}_1$ as follows:

$$\text{for } i = 1,2\text{: } \mathsf{B}_i := \mathsf{G}^{-1}(\mathsf{A}_i), \text{ where } \mathsf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times m}.$$

- This coincides with [BP14] construction by letting T be a spline-shaped tree, i.e., $s(T) = 1$.

Syntax of Constrained PRF

- Let $\mathcal{R} = \{\mathcal{R}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{D} = \{\mathcal{D}_\kappa\}_{\kappa \in \mathbb{N}}$ be families of sets representing the range and domain of the PRF respectively.
- Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of circuits, where $\mathcal{C}_\kappa$ is a set of circuits with domain $\mathcal{D}_\kappa$ and range $\{0, 1\}$.

Definition 3 (Syntax of CPRF)

A *constrained pseudorandom function for* $\mathcal{C}$ is defined by the five PPT algorithms $\Pi := (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ where:

- $\mathsf{Setup}(1^\kappa) \mapsto pp$.
- $\mathsf{Gen}(pp) \mapsto K$: K is referred to as *master key*.
- $\mathsf{Eval}(pp, K, x \in \mathcal{D}) \mapsto y \in \mathcal{R}$.
- $\mathsf{Constrain}(K, C \in \mathcal{C}) \mapsto K_C$: K_C is referred to as *constrained key*.
- $\mathsf{CEval}(pp, K_C, x) \mapsto y$: CEval takes as input a public parameter pp, a constrained key K_C, and an input $x \in \mathcal{D}$ and outputs $y \in \mathcal{R}$.

Pseudorandom on Constrained Points

The Game **PRoCP**

The game **PRoCP** between challenger $\mathbb{C}$ and adversary $\mathbb{A}$ has five stages:

- **Setup.** $\mathbb{C}$ runs $pp \leftarrow \mathsf{Setup}(1^\kappa)$, $K \leftarrow \mathsf{Gen}(pp)$, and set $S_{eval} = S_{con} = \emptyset$. $\mathbb{C}$ sends pp to $\mathbb{A}$.
- **Query.** $\mathbb{A}$ can *adaptively* make the two types of queries:
  - **Evaluation Query.** $\mathbb{A}$ queries $x \in \mathcal{D}$, and $\mathbb{C}$ returns $y \leftarrow \mathsf{Eval}(pp, K, x)$. $\mathbb{C}$ updates $S_{eval} := S_{eval} \cup \{x\}$.
  - **Constrained Key Query.** $\mathbb{A}$ queries $C \in \mathcal{C}$, and $\mathbb{C}$ returns $K_C \leftarrow \mathsf{Constrain}(K, C)$. $\mathbb{C}$ updates $S_{con} := S_{con} \cup \{C\}$.
- **Challenge.** $\mathbb{A}$ chooses $x^* \in \mathcal{D}$ s.t. $x^* \notin S_{eval}$ and $C(x^*) = 0$ for all $C \in S_{con}$. $\mathbb{C}$ toss a coin $b \xleftarrow{\$} \{0,1\}$; if $b = 0$, let $y^* \xleftarrow{\$} \mathcal{R}$, otherwise, $y* \leftarrow \mathrm{Eval(pp, K, x^*)}$. ;$\mathbb{C}$ returns $y*$ to $\mathbb{A}$.
- **Query.** Any query except for those $C \in \mathcal{C}$ with $C(x^*) = 0$.
- **Guess.** $\mathbb{A}$ guess $b' \in \{0,1\}$.

We say $\mathbb{A}$ *wins* iff $b = b'$.

Definition 4

A CPRF Π is said to be *(adaptively) pseudorandom on constrained points* if for all PPT adversary $\mathbb{A}$, it holds that $\left|\Pr\left[\mathbb{A} wins\right] - \frac{1}{2}\right| = \text{negl}(\kappa)$.

The CPRF is *selectively pseudorandom* if the constraint queries must be query at the begin of the stage 2.

Definition 5 (Collusion Resistance)

In the game **PRoCP**, if we can tolerate up to Q constrained key queries, we say the CPRF is *Q-collusion resistance*.

Definition 6

A *trapdoor* for a parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is any sufficiently "short" integer matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times n\ell}$ such that

$$\mathbf{AR} = \mathbf{HG},$$

for some invertible $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, called the *tag* of the trapdoor.

Trapdoor Generation

Sample $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$, a short $\bar{\mathbf{R}} \in \mathbb{Z}_q^{\bar{m} \times n\ell}$, and an invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. Set $\mathbf{A} := \left[\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\bar{\mathbf{R}}\right]$. Then $\mathbf{R} := \left[\begin{smallmatrix} \bar{\mathbf{R}} \\ \mathbf{I} \end{smallmatrix}\right]$ is a trapdoor for A with tag $\mathbf{H}$.

Let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ and define

$$\mathbf{A}_i := \bar{\mathbf{A}}\mathbf{R}_i - x_i \mathbf{G}, i = 1, 2.$$

That is, $\begin{bmatrix} \mathbf{R}_i \\ \mathbf{I} \end{bmatrix}$ is a trapdoor of $[\bar{\mathbf{A}} \mid \mathbf{A}_i]$ with tag $x_i \mathbf{I}$.

It holds that

$$\mathbf{A}_+ := \mathbf{A}_1 + \mathbf{A}_2 = \bar{\mathbf{A}}(\underbrace{\mathbf{R}_1 + \mathbf{R}_2}_{:=\mathbf{R}_+}) - (x_1 + x_2)\mathbf{G},$$

and

$$\begin{aligned} \mathbf{A}_\times := -\mathbf{A}_1 \cdot \mathbf{G}^{-1}(\mathbf{A}_2) &= -(\bar{\mathbf{A}}\mathbf{R}_1 - x_1\mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{A}_2) \\ &= -\bar{\mathbf{A}} \cdot \mathbf{R}_1\mathbf{G}^{-1}(\mathbf{A}_2) + x_1\mathbf{A}_2 \\ &= \bar{\mathbf{A}}(\underbrace{x_1\mathbf{R}_2 - \mathbf{R}_1\mathbf{G}^{-1}(\mathbf{A}_2)}_{:=\mathbf{R}_\times}) - x_1 x_2 \mathbf{G}. \end{aligned}$$

In the latter case, we need x_1 to be a *small* integer in order to get a good-quality trapdoor.

Homomorphic Evaluation of LWE Ciphertexts

Let $\mathbf{s} \in \mathbb{Z}_q^n$ and for $i = 1, 2$, let

$$\mathbf{u}_i^\top := \mathbf{s}^\top(\mathbf{A}_i + x_i\mathbf{G}) + \mathbf{e}_i^\top,$$

where $\mathbf{e}_i \leftarrow \chi^m$. Then

$$\mathbf{u}_+^\top := \mathbf{u}_1^\top + \mathbf{u}_2^\top = \mathbf{s}^\top((\underbrace{\mathbf{A}_1 + \mathbf{A}_2}_{\mathbf{A}_+}) + (x_1 + x_2)\mathbf{G}) + \underbrace{\mathbf{e}_1^\top + \mathbf{e}_2^\top}_{\mathbf{e}_+^\top},$$

and

$$\begin{aligned}
\mathbf{u}_\times^\top &:= x_1\mathbf{u}_2^\top - \mathbf{u}_1^\top\mathbf{G}^{-1}(\mathbf{A}_2) \\
&= x_1\left(\mathbf{s}^\top(\mathbf{A}_2 + x_2\mathbf{G}) + \mathbf{e}_2\right) - \left(\mathbf{s}^\top(\mathbf{A}_1 + x_1\mathbf{G}) + \mathbf{e}_1\right)\mathbf{G}^{-1}(\mathbf{A}_2) \\
&= \mathbf{s}^\top(\underbrace{-\mathbf{A}_1 \cdot \mathbf{G}^{-1}(\mathbf{A}_2)}_{\mathbf{A}_\times} + x_1x_2\mathbf{G}) + \underbrace{\mathbf{e}_1^\top\mathbf{G}^{-1}(\mathbf{A}_2) - x_1\mathbf{e}_2^\top}_{\mathbf{e}_\times^\top}.
\end{aligned}$$

Homomorphic Evaluation [BGG+14]

"Embed" bits $x_1, \ldots, x_k$ into matrices $\mathsf{A}_1, \ldots, \mathsf{A}_k \in \mathbb{Z}_q^{n \times m}$ and compute a circuit $C : \{0,1\}^k \to \{0,1\}$ on these matrices.

Homomorphic Evaluation

We have a pair of algorithms (ComputeA, ComputeC) satisfying the following properties:

- $\mathsf{ComputeA}(C, \mathsf{A}_1, \ldots, \mathsf{A}_k) \mapsto \mathsf{A}_C \in \mathbb{Z}_q^{n \times m}$.
- $\mathsf{ComputeC}(C, \{\mathsf{A}_i, x_i, \mathsf{u}_i\}_{i \in [k]}) \mapsto \mathsf{u}_C \in \mathbb{Z}_q^m$. If $\mathsf{u_i} = \mathsf{s}^\top(\mathsf{A}_i + x_i\mathsf{G}) + \mathsf{e}_i$, then

$$\mathsf{u}_C = \mathsf{s}^\top(\mathsf{A}_C + C(\mathsf{x})\mathsf{G}) + \mathsf{e}_C,$$

  where $\|\mathsf{e}_C\|_\infty \leq (1+m)^d \cdot \mathsf{max}_{i \in [k]} \|\mathsf{e}_i\|_\infty$.

- What we can do: Embed $\mathbf{x}$ into some matrices, and compute something about $C(\mathbf{x})$ when given circuit C.
- Goal: With the constrained key K_C for circuit C, we want to evaluate a function on some point $\mathbf{x}$ somehow related to $C(\mathbf{x})$.

Universal Circuit

Suppose that our circuits $\mathcal{C} := \left\{C : \{0,1\}^k \to \{0,1\}\right\}$ can be described by a string in $\{0,1\}^z$. There exists a *universal circuit* $\mathcal{U}_k : \{0,1\}^z \times \{0,1\}^k \to \{0,1\}$ such that

$$\mathcal{U}_k(C, x) = C(x), \forall C \in \mathcal{C}, \forall x \in \{0,1\}^k.$$

CPRF: First Attmept

- $\mathsf{Gen}(1^\kappa, 1^z) \mapsto (pp, K)$: Output

$$pp := (\underbrace{\mathbf{A}_0, \mathbf{A}_1}_{\text{for input } \mathbf{x}}, \underbrace{\mathbf{B}_1, \ldots, \mathbf{B}_z}_{\text{for circuit } C}), K := \mathbf{s},$$

  where $\mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_1, \ldots, \mathbf{B}_z \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$.
- $\mathsf{Eval}(pp, K = \mathbf{s}, \mathbf{x} \in \{0,1\}^k)$: Compute

$$\mathbf{B}_{\mathcal{U},\mathbf{x}} := \mathsf{ComputeA}\left(\mathcal{U}_k, \mathbf{B}_1, \ldots, \mathbf{B}_z, \mathbf{A}_{x_1}, \ldots, \mathbf{A}_{x_k}\right),$$

  and output $F_{\mathbf{s}}(\mathbf{x}) = \left\lfloor s^\top \mathbf{B}_{\mathcal{U},\mathbf{x}} \right\rceil_p$.
- $\mathsf{Constrain}(pp, \mathbf{s}, C)$: Compute for $b \in \{0,1\}, i \in [z]$:

$$\mathbf{a}_b := \mathbf{s}^\top(\mathbf{A}_b + b \cdot \mathbf{G}) + \mathbf{e}_{1,b}^\top \in \mathbb{Z}_q^m, \quad \mathbf{b}_i := \mathbf{s}^\top(\mathbf{B}_i + C_i \cdot \mathbf{G}) + \mathbf{e}_{2,i}^\top \in \mathbb{Z}_q^m,$$

  where $\mathbf{e} \leftarrow \chi$. Output $K_C := (\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_1, \ldots, \mathbf{b}_z)$.
- $\mathsf{CEval}(pp, K_C, \mathbf{x})$: Compute

$$\mathbf{b}_{\mathcal{U},\mathbf{x}} := \mathsf{ComputeC}\left(\mathcal{U}, \left(\mathbf{b}_1, \ldots, \mathbf{b}_z, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_k}\right), \left(C_1, \ldots, C_z, x_1, \ldots, x_k\right)\right).$$

  Output $\left\lfloor \mathbf{b}_{\mathcal{U},\mathbf{x}} \right\rceil_p$.

Correctness

✓ $\mathsf{b}_{\mathcal{U},x} = \mathsf{s}^\top(\mathsf{B}_{\mathcal{U},x} + C(\mathsf{x})\mathsf{G}) + \text{noise}.$

But what if $\lfloor\cdot\rceil_p$ errs? This kind of event can be used to solve the following 1D-SIS problem.

Definition 7 (The One-Dimensional Short Integer Solution problem ID-SIS$_{q,m,t}$)

Given a uniformly distributed vector $\mathsf{v} \in \mathbb{Z}_q^m$, find $\mathbf{z} \in \mathbb{Z}^m$ such that

$$\|\mathbf{z}\| \leq t \text{ and } \langle \mathsf{v}, \mathbf{z} \rangle \in [-t, t] + q\mathbb{Z}.$$

Theorem 8 ([GPV07])

Let $n \in \mathbb{N}$ and $q = \prod_{i\in[n]} p_i$, where all $p_1 < p_2 < \cdots < p_n$ are co-prime. Let $m \geq c \cdot n \log q$ (for some universal constant c). Assuming that $p_1 \geq t\omega(\sqrt{mnlogn})$, 1D-SIS$_{q,m,t}$ is at least as hard as SIVP$_{t\cdot\tilde{O}(\sqrt{mn})}$ and GapSVP$_{t\cdot\tilde{O}(\sqrt{mn})}$.

Achieving Pseudorandomess

✗ Pseudorandom on unauthorized points: if $C(\mathbf{x}) = 1$, it is indeed hard to compute $F_{\mathbf{s}}(\mathbf{x})$, but *not* pseudorandom.

Solution

Introduce a new independent LWE matrix $\mathbf{D}$ in *pp* and

$$\text{Eval}(pp, \mathbf{s}, \mathbf{x}) \text{ outputs } \left\lfloor s^\top \mathbf{B}_{\mathcal{U},\mathrm{x}} \cdot \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p .$$

Now we have

$$s^\top \mathbf{B}_{\mathcal{U},\mathrm{x}} \cdot \mathbf{G}^{-1}(\mathbf{D}) \approx \mathbf{s}^\top \left((\mathbf{B}_{\mathcal{U},\mathrm{x}} - C(\mathbf{x})\mathbf{G}) + \text{noise} \right) \cdot \mathbf{G}^{-1}(\mathbf{D}) + C(\mathbf{x}) \left(\mathbf{s}^\top \mathbf{D} + \text{noise} \right).$$

✓ When $C(\mathbf{x}) = 1$, the blue part randomizes the expression.

✓ Correctness still holds since $\mathbf{G}^{-1}(\mathbf{D})$ has low norm.

Coup de Grace

$$F_{\mathsf{s}}(\mathsf{x}) := \left\lfloor s^\top \mathbf{B}_{\mathcal{U},\mathsf{x}} \cdot \mathbf{G}^{-1}(\mathbf{D}) \right\rceil_p.$$

✗ Only for *single query*, since the randomness from $\mathbf{D}$ can only use once.

Solution

Use *admissible hash* to deal with the challenge query x^* differently.

Now this is exactly the construction in [BV15]!

1-Key Privacy (or Constraint-Hinding)

The Game CH

The game **CH** between challenger $\mathbb{C}$ and adversary $\mathbb{A}$ has three stages:

- **Setup.** $\mathbb{C}$ runs $pp \leftarrow \mathsf{Setup}(1^\kappa)$, $K \leftarrow \mathsf{Gen}(pp)$, and set $S_{eval} = S_{con} = \emptyset$. $\mathbb{C}$ sends pp to $\mathbb{A}$.
- **Constraind Key Query.**
  - $\mathbb{A}$ send two circuits $C_0, C_1 \in \mathcal{C}$ to $\mathbb{C}$
  - $\mathbb{C}$ toss a coin $b \xleftarrow{\$} \{0,1\}$ and sends $K_b \leftarrow \mathsf{Constrain}(K, C_b)$ to $\mathbb{A}$.
- **Guess.** $\mathbb{A}$ guesses $b' \in \{0,1\}$.

$\mathbb{A}$ wins iff $b' = b$.

Definition 9

A CPRF Π is said to satisfy *1-key privacy* if for all PPT adversary $\mathbb{A}$, it holds that $\left|\Pr[\mathbb{A} wins] - \frac{1}{2}\right| = \mathsf{negl}(\kappa)$.

State of Art

Table 2: List of existing constructions of CPRFs along with their functionality and the assumptions required.

	Adaptive	Collusion-resistance	Privacy	Predicate	Assumption
[BW13]	×	poly	$0^{\dagger}$	$\text{Prefix}^{\ddagger}$	OWF
	✓	poly	poly	LR	BDDH & ROM
	×	poly	0	BF	MLDDH
	×	poly	0	P/poly	MLDDH
[KPTZ13]	×	poly	$0^{\dagger}$	$\text{Prefix}^{\ddagger}$	OWF
[BGI14]	×	poly	$0^{\dagger}$	$\text{Prefix}^{\ddagger}$	OWF
[BZ14]	×	poly	0	P/poly	IO
[HKKW19]	✓	poly	0	P/poly	IO & ROM
[BFP+15]	×	poly	0	Prefix	LWE
[BV15]	×	1	0	P/poly	LWE
[HKW15]	✓	poly	0	Puncturing	SGH & IO
[BLW17]	×	poly	1 (weak)	Puncturing	MLDDH
	×	poly	1 (weak)	BF	MLDDH
	×	poly	poly	P/poly	IO
[BTVW17]	×	1	1	P/poly	LWE
[CC17]	×	1	1	BF	LWE
	×	1	1	NC^1	LWE
[AMN+18]	×	1	1	BF	DDH
	×	1	0	NC^1	L-DDHI
	✓	1	1	BF	ROM
	✓	1	0	NC^1	L-DDHI & ROM
[CVW18]	×	1	1	NC^1	LWE
[PS18]	×	1	1	P/poly	LWE
[AMN+19]	✓	1	0	NC^1	SGH & IO
Section 4	✓	$O(1)$	1 (weak)	t-CNF ($\supseteq$ BF)	OWF
Section 5	✓	1	1 (weak)	IP	LWE
Section 6	✓	$O(1)$	0	P/poly	LWE & IO

Figure 2: Taken from [DKN+20]

Discussion

- Can we support the following functionality?

$$\mathsf{AddConstraint}(pp, K_C, C') \mapsto K_{C \wedge C'}.$$

- Support more collusion.
- Achieving adaptive security.
- CPRF from other assumptions?

- D. Boneh, Craig Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and Dhinakaran Vinayagamurthy, *Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits*, EUROCRYPT, 2014.
- D. Boneh, Kevin Lewi, H. Montgomery, and A. Raghunathan, *Key homomorphic prfs and their applications*, CRYPTO, 2013.
- Abhishek Banerjee and Chris Peikert, *New and improved key-homomorphic pseudorandom functions*, Annual Cryptology Conference, Springer, 2014, pp. 353–370.
- Abhishek Banerjee, Chris Peikert, and Alon Rosen, *Pseudorandom functions and lattices*, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 719–737.

References ii

- Zvika Brakerski and V. Vaikuntanathan, *Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your prf*, TCC, 2015.
- Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa, *Adaptively secure constrained pseudorandom functions in the standard model*, CRYPTO, 2020.
- Craig Gentry, Chris Peikert, and V. Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, Proceedings of the fortieth annual ACM symposium on Theory of computing (2007).