Privately Constraining and Programming PRFs, the LWE Way

$\boxed{\text{Chris Peikert}}$ Sina Shiehian

PKC 2018

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm Eval(msk, x).

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm Eval(msk, x).
2. For any constraint $C \in \mathcal{C}$, can generate a constrained key sk_C (using msk).

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm Eval(msk, x).
2. For any constraint $C \in \mathcal{C}$, can generate a constrained key sk_C (using msk).
3. Constrained evaluation algorithm CEval(sk_C, x).

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm Eval(msk, x).
2. For any constraint $C \in \mathcal{C}$, can generate a constrained key sk_C (using msk).
3. Constrained evaluation algorithm CEval(sk_C, x).

Correctness

- If $C(x) = 0$ ("authorized") then CEval$(sk_C, x) =$ Eval(msk, x).

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk, x)$.
2. For any constraint $C \in \mathcal{C}$, can generate a constrained key sk_C (using msk).
3. Constrained evaluation algorithm $\mathsf{CEval}(sk_C, x)$.

Correctness

- If $C(x) = 0$ ("authorized") then $\mathsf{CEval}(sk_C, x) = \mathsf{Eval}(msk, x)$.

Security

- If $C(x) = 1$ ("unauth") then $\mathsf{Eval}(msk, x) \overset{c}{\approx}$ random (even w/ sk_C).

Constrained Pseudorandom Functions [KPTZ'13,BW'13,BGI'14]

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk, x)$.
2. For any constraint $C \in \mathcal{C}$, can generate a constrained key sk_C (using msk).
3. Constrained evaluation algorithm $\mathsf{CEval}(sk_C, x)$.

Correctness

- If $C(x) = 0$ ("authorized") then $\mathsf{CEval}(sk_C, x) = \mathsf{Eval}(msk, x)$.

Security

- If $C(x) = 1$ ("unauth") then $\mathsf{Eval}(msk, x) \overset{c}{\approx}$ random (even w/sk_C).

- Applications: uses of *iO* [SW'14], ID-based key exchange, broadcast encryption, . . .

Privacy and Programmability [BonehLewiWu'17]

- Ordinarily, a constrained key sk_C may reveal C.
  (It hides only the *PRF output* at unauthorized x.)

Privacy and Programmability [BonehLewiWu'17]

- Ordinarily, a constrained key sk_C may reveal C.
  (It hides only the *PRF output* at unauthorized x.)

Privacy (a.k.a. Constraint Hiding)

- Constrained key sk_C reveals nothing about C.
  In particular, it hides whether x is (un)authorized.

Privacy and Programmability [BonehLewiWu'17]

- Ordinarily, a constrained key sk_C may reveal C.
  (It hides only the *PRF output* at unauthorized x.)

Privacy (a.k.a. Constraint Hiding)

- Constrained key sk_C reveals nothing about C.
  In particular, it hides whether x is (un)authorized.
- Applications: searchable encryption, function secret sharing [BGI'15].

Privacy and Programmability [BonehLewiWu'17]

- Ordinarily, a constrained key sk_C may reveal C.
  (It hides only the *PRF output* at unauthorized x.)

Privacy (a.k.a. Constraint Hiding)

- Constrained key sk_C reveals nothing about C.
  In particular, it hides whether x is (un)authorized.
- Applications: searchable encryption, function secret sharing [BGI'15].

Programmability

- Can program sk_C to produce a desired value at some unauthorized x^*.
  (Nontrivial only if unauthorized x are hidden.)

Privacy and Programmability [BonehLewiWu'17]

- Ordinarily, a constrained key sk_C may reveal C.
  (It hides only the *PRF output* at unauthorized x.)

Privacy (a.k.a. Constraint Hiding)

- Constrained key sk_C reveals nothing about C.
  In particular, it hides whether x is (un)authorized.
- Applications: searchable encryption, function secret sharing [BGI'15].

Programmability

- Can program sk_C to produce a desired value at some unauthorized x^*.
  (Nontrivial only if unauthorized x are hidden.)
- Applications: watermarking PRFs, ???.

Prior Results

BLW'17 Private constrained PRFs for all functions, & programmable PRFs, from *iO*.

Prior Results

BLW'17 Private constrained PRFs for all functions, & programmable PRFs, from *iO*.

BKM'17 Private constrained PRFs for point functions, from LWE.

Prior Results

BLW'17	Private constrained PRFs for all functions, & programmable PRFs, from *iO*.
BKM'17	Private constrained PRFs for point functions, from LWE.
CC'17	Private constrained PRFs for NC^1 circuits, from LWE.

Prior Results

BLW'17 Private constrained PRFs for all functions, & programmable PRFs, from *iO*.

BKM'17 Private constrained PRFs for point functions, from LWE.

CC'17 Private constrained PRFs for NC^1 circuits, from LWE.

BTVW'17 Private constrained PRFs for all circuits, from LWE.

Prior Results

BLW'17 Private constrained PRFs for all functions, & programmable PRFs, from *iO*.

BKM'17 Private constrained PRFs for point functions, from LWE.

CC'17 Private constrained PRFs for NC^1 circuits, from LWE.

BTVW'17 Private constrained PRFs for all circuits, from LWE.

Caveat! Limited to one constrained key.
Two keys for arbitrary circuits $\Rightarrow$ *iO* [CC'17]

Prior Results

BLW'17	Private constrained PRFs for all functions, & programmable PRFs, from *iO*.
BKM'17	Private constrained PRFs for point functions, from LWE.
CC'17	Private constrained PRFs for NC^1 circuits, from LWE.
BTVW'17	Private constrained PRFs for all circuits, from LWE.
Caveat!	Limited to one constrained key. Two keys for arbitrary circuits $\Rightarrow$ *iO* [CC'17]
Open	Programmable PRFs from a non-*iO* assumption.

Our Results

Main Message

- A unified approach to private constrained and programmable PRFs from LWE: shift-hiding functions.
- Simple, modular constructions via the 'right' choice of shift function.

Our Results

Main Message

- A unified approach to private constrained and programmable PRFs from LWE: shift-hiding functions.
- Simple, modular constructions via the ‘right’ choice of shift function.

Constructions

1. Shift-hiding functions from LWE by standard FHE/ABE/PE tech [GSW’13,BGG+’14,GVW’15]

Our Results

Main Message

- A unified approach to private constrained and programmable PRFs from LWE: shift-hiding functions.
- Simple, modular constructions via the 'right' choice of shift function.

Constructions

1. Shift-hiding functions from LWE by standard FHE/ABE/PE tech [GSW'13,BGG+'14,GVW'15]
2. Private constrained & programmable PRFs, simply by letting

$$\text{shift} = \text{constraint} \times \text{(pseudo)random function}$$

Our Results

Main Message

- A unified approach to private constrained and programmable PRFs from LWE: shift-hiding functions.
- Simple, modular constructions via the ‘right’ choice of shift function.

Constructions

1. Shift-hiding functions from LWE by standard FHE/ABE/PE tech [GSW'13,BGG+'14,GVW'15]
2. Private constrained & programmable PRFs, simply by letting
   shift = constraint × (pseudo)random function

In particular, the first programmable PRFs from non-*iO* assumptions.

Our Results

Main Message

- A unified approach to private constrained and programmable PRFs from LWE: shift-hiding functions.
- Simple, modular constructions via the 'right' choice of shift function.

Constructions

1. Shift-hiding functions from LWE by standard FHE/ABE/PE tech [GSW'13,BGG+'14,GVW'15]
2. Private constrained & programmable PRFs, simply by letting

$$\text{shift} = \text{constraint} \times \text{(pseudo)random function}$$

In particular, the first programmable PRFs from non-*iO* assumptions.

Selectively simulation-secure, for *a priori* bounded-size functions.

Shift-Hiding Functions

$\Downarrow$

Private/Programmable PRFs

Main Tool: Shift-Hiding Functions

1. Ordinary evaluation algorithm Eval$(msk, \cdot)\colon \mathcal{X} \to \mathbb{Z}_q$.

Main Tool: Shift-Hiding Functions

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk, \cdot): \mathcal{X} \to \mathbb{Z}_q$.
2. Shifting algorithm $sk_H \leftarrow \mathsf{Shift}(msk, H)$ for shift fct $H: \mathcal{X} \to \mathbb{Z}_q$.

Main Tool: Shift-Hiding Functions

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk,\,\cdot\,)\colon \mathcal{X} \to \mathbb{Z}_q$.
2. Shifting algorithm $sk_H \leftarrow \mathsf{Shift}(msk, H)$ for shift fct $H\colon \mathcal{X} \to \mathbb{Z}_q$.
3. Shifted evaluation algorithm $\mathsf{SEval}(sk_H,\,\cdot\,)\colon \mathcal{X} \to \mathbb{Z}_q$.

Main Tool: Shift-Hiding Functions

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk, \cdot\,)\colon \mathcal{X} \to \mathbb{Z}_q$.
2. Shifting algorithm $sk_H \leftarrow \mathsf{Shift}(msk, H)$ for shift fct $H\colon \mathcal{X} \to \mathbb{Z}_q$.
3. Shifted evaluation algorithm $\mathsf{SEval}(sk_H, \cdot\,)\colon \mathcal{X} \to \mathbb{Z}_q$.

Shifting

- For every shift function H and every $x \in \mathcal{X}$:

$$\mathsf{SEval}(sk_H, x) \approx \mathsf{Eval}(msk, x) + H(x) \pmod q$$

Main Tool: Shift-Hiding Functions

1. Ordinary evaluation algorithm $\mathsf{Eval}(msk, \cdot\,) \colon \mathcal{X} \to \mathbb{Z}_q$.
2. Shifting algorithm $sk_H \leftarrow \mathsf{Shift}(msk, H)$ for shift fct $H \colon \mathcal{X} \to \mathbb{Z}_q$.
3. Shifted evaluation algorithm $\mathsf{SEval}(sk_H, \cdot\,) \colon \mathcal{X} \to \mathbb{Z}_q$.

Shifting

- For every shift function H and every $x \in \mathcal{X}$:

$$\mathsf{SEval}(sk_H, x) \approx \mathsf{Eval}(msk, x) + H(x) \pmod q$$

Hiding

- sk_H reveals nothing about H.

Shift-Hiding Functions $\Rightarrow$ Private Constrained PRFs

- $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$ "rounds off."

Shift-Hiding Functions $\Rightarrow$ Private Constrained PRFs

- $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$ "rounds off."
- To generate a constrained key for circuit C, define shift function

$$H(x) = C(x) \cdot \mathsf{PRF}_k(x)$$

and output $sk_C \leftarrow \mathsf{Shift}(msk, H)$. This hides H, hence C (and k).

Shift-Hiding Functions $\Rightarrow$ Private Constrained PRFs

- $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$ "rounds off."
- To generate a constrained key for circuit C, define shift function

  $$H(x) = C(x) \cdot \mathsf{PRF}_k(x)$$

  and output $sk_C \leftarrow \mathsf{Shift}(msk, H)$. This hides H, hence C (and k).
- Constrained evaluation: $\lfloor \mathsf{SEval}(sk_C, x) \rceil$. By shifting property, this is

  $$\lfloor \mathsf{Eval}(msk, x) + H(x) \rceil = \begin{cases} F(msk, x) & \text{if } C(x) = 0 \\ \overset{c}{\approx} \text{ random} & \text{if } C(x) = 1. \end{cases}$$

Shift-Hiding Functions $\Rightarrow$ Programmable PRFs

- As before, $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$.

Shift-Hiding Functions $\Rightarrow$ Programmable PRFs

- As before, $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$.
- To generate a programmed key that maps x^* to $y^* \in \{0, 1\}$:
  Choose random $z^* \in \mathbb{Z}_q$ s.t. $\lfloor z^* \rceil = y^*$, define shift function

$$H(x) = \begin{cases} y^* - \mathsf{Eval}(msk, x^*) & \text{if } x = x^* \\ 0 & \text{otherwise,} \end{cases}$$

  and output $sk_C = \mathsf{Shift}(sk, H)$. This hides H, hence x^*.

Shift-Hiding Functions $\Rightarrow$ Programmable PRFs

- As before, $F(msk, x) := \lfloor \mathsf{Eval}(msk, x) \rceil$, where $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_2$.
- To generate a programmed key that maps x^* to $y^* \in \{0, 1\}$: Choose random $z^* \in \mathbb{Z}_q$ s.t. $\lfloor z^* \rceil = y^*$, define shift function

$$H(x) = \begin{cases} y^* - \mathsf{Eval}(msk, x^*) & \text{if } x = x^* \\ 0 & \text{otherwise,} \end{cases}$$

  and output $sk_C = \mathsf{Shift}(sk, H)$. This hides H, hence x^*.
- As before, constrained evaluation is $\lfloor \mathsf{SEval}(sk_C, x) \rceil$. This is

$$\lfloor \mathsf{Eval}(msk, x) + H(x) \rceil = \begin{cases} \lfloor z^* \rceil = y^* & \text{if } x = x^* \\ F(msk, x) & \text{otherwise.} \end{cases}$$

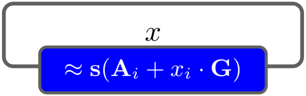
Construction Shift-Hiding Functions

Gadget Homomorphisms [MP'12,GSW'13,BGG+'14,GVW'15]

- Fixed 'gadget' matrix $\mathbf{G}$, public random matrices $\mathbf{A}_i$ over $\mathbb{Z}_q$.

Gadget Homomorphisms [MP'12,GSW'13,BGG+'14,GVW'15]

- Fixed 'gadget' matrix $\mathbf{G}$, public random matrices $\mathbf{A}_i$ over $\mathbb{Z}_q$.
- 'Embed' $x_i \in \{0,1\}$ or $\mathbb{Z}_q$ w.r.t. $\mathbf{A}_i$ as $\approx \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G})$



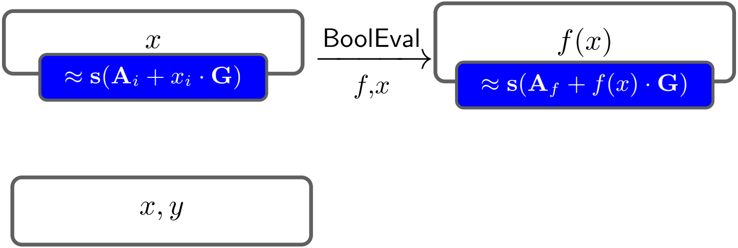
Gadget Homomorphisms [MP'12,GSW'13,BGG+'14,GVW'15]

- Fixed 'gadget' matrix $\mathbf{G}$, public random matrices $\mathbf{A}_i$ over $\mathbb{Z}_q$.
- 'Embed' $x_i \in \{0,1\}$ or $\mathbb{Z}_q$ w.r.t. $\mathbf{A}_i$ as $\approx \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G})$
- Can compute embedded $f(x)$ w.r.t. $\mathbf{A}_f$, knowing x (but not $\mathbf{s}$) ...

$$\boxed{\begin{array}{c} x \\ \boxed{\approx \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G})} \end{array}} \xrightarrow[f,x]{\text{BoolEval}} \boxed{\begin{array}{c} f(x) \\ \boxed{\approx \mathbf{s}(\mathbf{A}_f + f(x) \cdot \mathbf{G})} \end{array}}$$

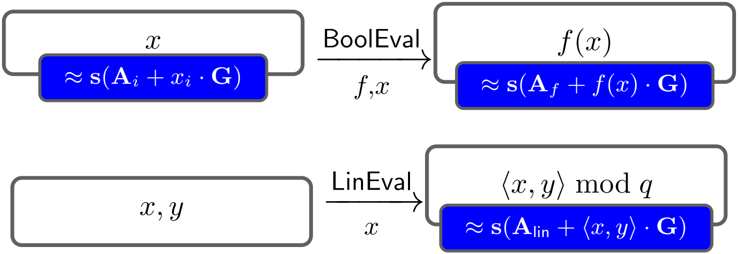
Gadget Homomorphisms [MP'12,GSW'13,BGG+'14,GVW'15]

- Fixed 'gadget' matrix $\mathbf{G}$, public random matrices $\mathbf{A}_i$ over $\mathbb{Z}_q$.
- 'Embed' $x_i \in \{0,1\}$ or $\mathbb{Z}_q$ w.r.t. $\mathbf{A}_i$ as $\approx \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G})$
- Can compute embedded $f(x)$ w.r.t. $\mathbf{A}_f$, knowing x (but not $\mathbf{s}$) ...
- ...and embedded $\langle x, y\rangle \bmod q$ w.r.t. $\mathbf{A}_{\mathsf{lin}}$, knowing x (but not $y, \mathbf{s}$).



Gadget Homomorphisms [MP'12,GSW'13,BGG+'14,GVW'15]

- Fixed 'gadget' matrix $\mathbf{G}$, public random matrices $\mathbf{A}_i$ over $\mathbb{Z}_q$.
- 'Embed' $x_i \in \{0,1\}$ or $\mathbb{Z}_q$ w.r.t. $\mathbf{A}_i$ as $\approx \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G})$
- Can compute embedded $f(x)$ w.r.t. $\mathbf{A}_f$, knowing x (but not $\mathbf{s}$) ...
- ... and embedded $\langle x, y \rangle \bmod q$ w.r.t. $\mathbf{A}_{\mathsf{lin}}$, knowing x (but not $y, \mathbf{s}$).



Input Privacy [GorbunovVaikuntanathanWee'15]

- Goal: in gadget embedding, compute many known f on one *private* x.

Input Privacy [GorbunovVaikuntanathanWee'15]

- Goal: in gadget embedding, compute many known f on one *private* x.

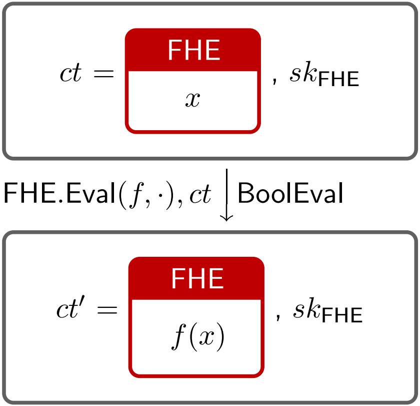
1. Encrypt x under FHE and embed $ct = \text{FHE}(x), sk_{\text{FHE}}$.

$$ct = \boxed{\begin{array}{c}\text{FHE}\\ x\end{array}}\,,\ sk_{\text{FHE}}$$

Input Privacy [GorbunovVaikuntanathanWee'15]

- Goal: in gadget embedding, compute many known f on one *private* x.

1. Encrypt x under FHE and embed $ct = \text{FHE}(x), sk_{\text{FHE}}$.
2. BoolEval $\Rightarrow$ embedding of $ct' = \text{FHE.Eval}(f, ct) = \text{FHE}(f(x))$.



Input Privacy [GorbunovVaikuntanathanWee'15]

- Goal: in gadget embedding, compute many known f on one *private* x.

1. Encrypt x under FHE and embed $ct = \mathsf{FHE}(x), sk_{\mathsf{FHE}}$.
2. BoolEval $\Rightarrow$ embedding of $ct' = \mathsf{FHE.Eval}(f, ct) = \mathsf{FHE}(f(x))$.
3. LinEval $\Rightarrow$ embedding of $\langle ct', sk_{\mathsf{FHE}} \rangle \approx f(x) \bmod q$.

$$ct = \boxed{\begin{array}{c}\mathsf{FHE}\\ x\end{array}}\,,\ sk_{\mathsf{FHE}}$$

$$\Big\downarrow \ \mathsf{FHE.Eval}(f,\cdot), ct \quad \mathsf{BoolEval}$$

$$ct' = \boxed{\begin{array}{c}\mathsf{FHE}\\ f(x)\end{array}}\,,\ sk_{\mathsf{FHE}} \quad \xrightarrow[ct']{\mathsf{LinEval}} \quad \boxed{\begin{array}{c}\approx\ f(x)\\ \approx \mathbf{s}(\mathbf{A}_f + (\approx f(x)) \cdot \mathbf{G})\end{array}}$$

Switching Function and Input [BTVW'17,this work]

- Goal: in embedding, compute one *private* H on many known x.

Switching Function and Input [BTVW'17,this work]

- Goal: in embedding, compute one *private* H on many known x.

1. Encrypt H under FHE and embed ct, sk_{FHE}.

$$ct = \boxed{\begin{array}{c}\textsf{FHE}\\ \hline H\end{array}}, \; sk_{\mathsf{FHE}}$$

Switching Function and Input [BTVW'17,this work]

- Goal: in embedding, compute one *private* H on many known x.

1. Encrypt H under FHE and embed ct, sk_{FHE}.
2. For input x, FHE-evaluate universal circuit $U_x(H) = H(x)$.

$$ct = \boxed{\begin{array}{c}\mathsf{FHE}\\ H\end{array}},\ sk_{\mathsf{FHE}}$$

$$\Big\downarrow\ \mathsf{FHE.Eval}(U_x, \cdot), ct \quad \mathsf{BoolEval}$$

$$ct' = \boxed{\begin{array}{c}\mathsf{FHE}\\ H(x)\end{array}},\ sk_{\mathsf{FHE}}$$

Switching Function and Input [BTVW'17,this work]

- Goal: in embedding, compute one *private* H on many known x.

1. Encrypt H under FHE and embed ct, sk_{FHE}.
2. For input x, FHE-evaluate universal circuit $U_x(H) = H(x)$.

$ct = $ FHE[H] , sk_{FHE}

$$\Big\downarrow \; \text{FHE.Eval}(U_x, \cdot), ct \;\; \text{BoolEval}$$

$ct' = $ FHE[$H(x)$] , sk_{FHE} $\xrightarrow[ct']{\text{LinEval}}$ $\approx \; H(x)$

$$\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G})$$

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.
- Shifted key $sk_H :=$ $\boxed{ct = \boxed{\begin{array}{c}\text{FHE}\\ \hline H\end{array}},\ sk_{\mathsf{FHE}}}$

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.
- Shifted key $sk_H := \boxed{ct = \boxed{\begin{array}{c}\text{FHE}\\ H\end{array}},\ sk_{\mathsf{FHE}}}$
- $\mathsf{SEval}(sk_H, x) :=$ first entry of $\boxed{\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G})}$

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.
- Shifted key $sk_H :=$ $\boxed{ct = \boxed{\begin{array}{c}\text{FHE} \\ H\end{array}}, sk_{\mathsf{FHE}}}$
- SEval$(sk_H, x) :=$ first entry of $\boxed{\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G})}$
- Eval$(msk, x) :=$ first entry of $\mathbf{s}\mathbf{A}_x$.

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.
- Shifted key $sk_H :=$ $\boxed{ct = \boxed{\begin{array}{c}\textsf{FHE}\\ H\end{array}}, sk_{\textsf{FHE}}}$
- $\textsf{SEval}(sk_H, x) :=$ first entry of $\boxed{\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G})}$
- $\textsf{Eval}(msk, x) :=$ first entry of $\mathbf{s}\mathbf{A}_x$.

Shift Correctness

$$\begin{aligned}\textsf{SEval}(sk_H, x) &\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G}) \cdot \mathbf{e}_1 \\ &= \textsf{Eval}(msk, x) + \mathbf{s} \cdot (\approx H(x)) \cdot \mathbf{G} \cdot \mathbf{e}_1\end{aligned}$$

Constructing Shift-Hiding Functions

- Master secret key $msk :=$ LWE secret $\mathbf{s}$, with $s_1 = 1$.
- Shifted key $sk_H := \boxed{ct = \boxed{\begin{array}{c}\text{FHE}\\ H\end{array}}, sk_{\text{FHE}}}$
- SEval$(sk_H, x) :=$ first entry of $\boxed{\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G})}$
- Eval$(msk, x) :=$ first entry of $\mathbf{s}\mathbf{A}_x$.

Shift Correctness

$$\begin{aligned} \text{SEval}(sk_H, x) &\approx \mathbf{s}(\mathbf{A}_x + (\approx H(x)) \cdot \mathbf{G}) \cdot \mathbf{e}_1 \\ &= \text{Eval}(msk, x) + \mathbf{s} \cdot (\approx H(x)) \cdot \mathbf{G} \cdot \mathbf{e}_1 \\ &\approx \text{Eval}(msk, x) + H(x). \end{aligned}$$

Open Problems

1. Better modulus-to-noise ratio?

   (Currently exponential in size of shift function H.)

Open Problems

1. Better modulus-to-noise ratio?
   (Currently exponential in size of shift function H.)
2. Adaptive security? (Currently selective in choice of H.)

Open Problems

1. Better modulus-to-noise ratio?
   (Currently exponential in size of shift function H.)
2. Adaptive security? (Currently selective in choice of H.)
3. One construction for all circuit sizes?
   (Currently 'leveled'; related to bootstrapping ABE.)

Open Problems

1. Better modulus-to-noise ratio?
   (Currently exponential in size of shift function H.)
2. Adaptive security? (Currently selective in choice of H.)
3. One construction for all circuit sizes?
   (Currently 'leveled'; related to bootstrapping ABE.)
4. Programming superpolynomially many inputs?
   (Currently limited to *a priori* polynomial.)

Open Problems

1. Better modulus-to-noise ratio?
   (Currently exponential in size of shift function H.)
2. Adaptive security? (Currently selective in choice of H.)
3. One construction for all circuit sizes?
   (Currently 'leveled'; related to bootstrapping ABE.)
4. Programming superpolynomially many inputs?
   (Currently limited to *a priori* polynomial.)

Thanks!