Indistinguishability Obfuscation from Low-Degree Multilinear Maps and (Blockwise) Local PRGs

[Lin16b, LT17, To appear, Crypto'17]

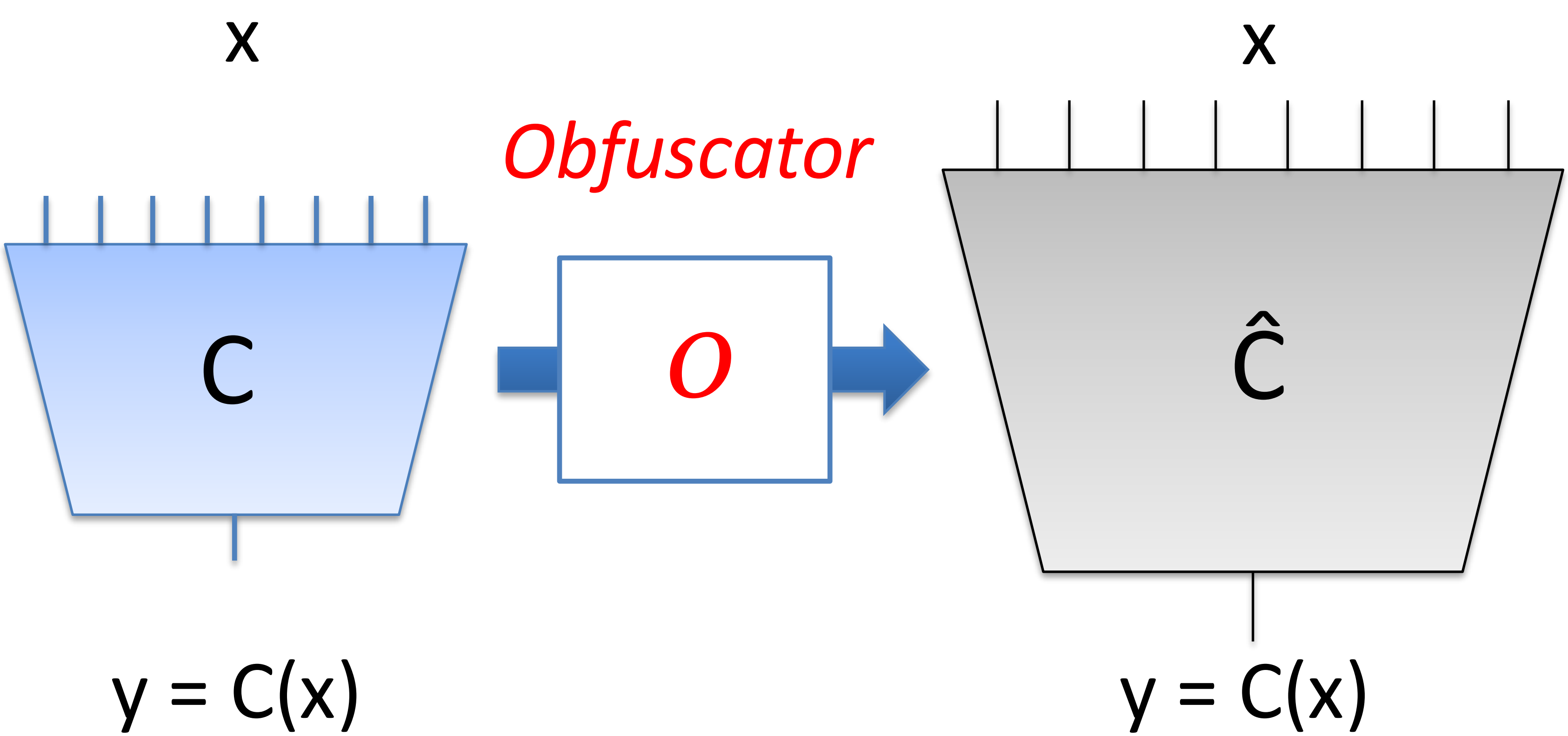
Huijia (Rachel) Lin

UCSB

Partial Joint work with Stefano Tessaro

Circuit Obfuscation

Compile a circuit C into Ĉ that *preserves functionality,* *and is unintelligible (resistant to reverse engineering)*

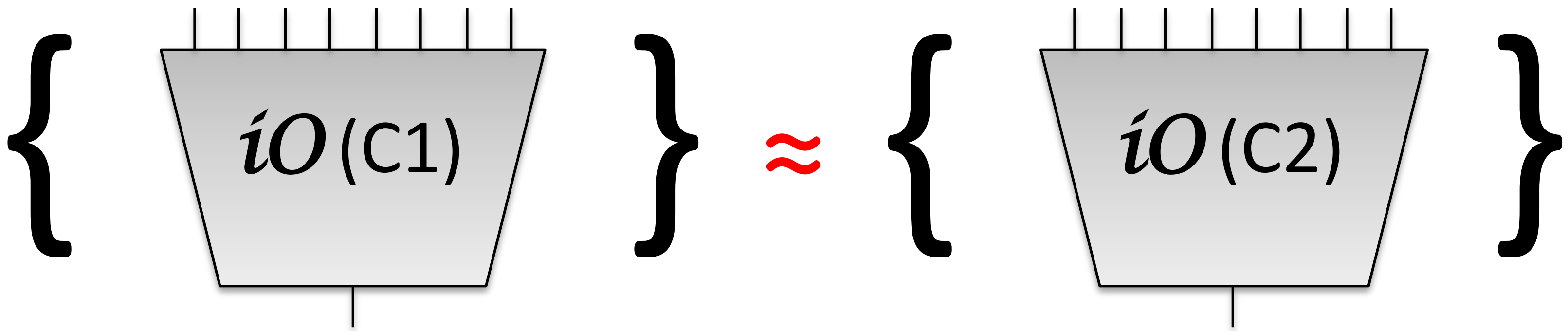


Indistinguishability Obfuscator $i\mathcal{O}$ [BGI+01]

"Which one of two equivalent circuits $C_1 \equiv C_2$ is obfuscated?"

$C_1 \equiv C_2$, meaning

- Same size $|C_1| = |C_2|$
- Same truth table $TB(C_1) = TB(C_2)$



Balancing at the border of (in)security

Candidate

iO

[GGHRSW13,BR14,BGKPS14,PST14,GLSW14,AGIS14,Zim15, AB15,GMMSSZ16,DGGM16,Lin16,LV16,AS16,Lin16b,LT17]

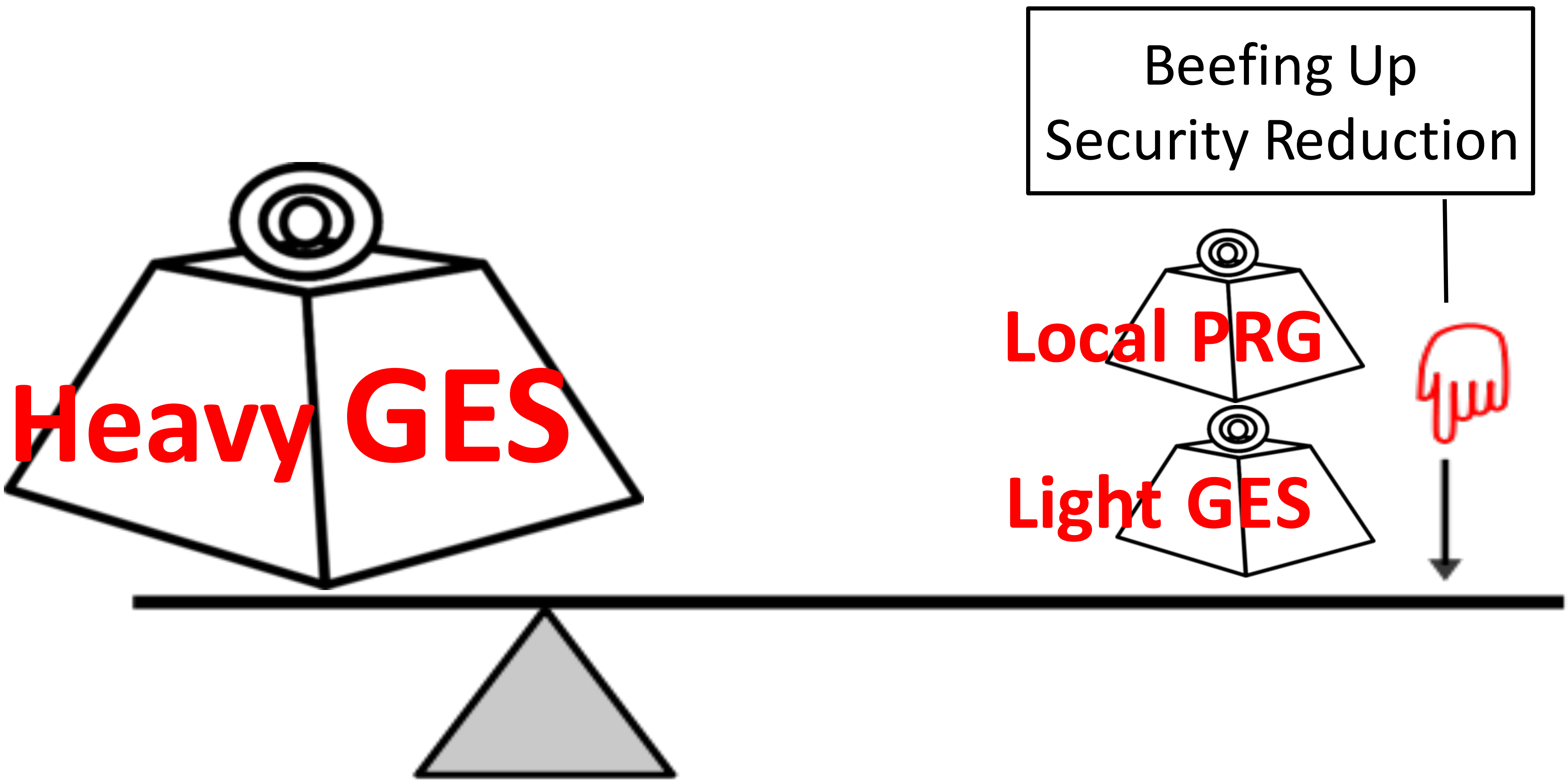
Graded Encodings [GGH13]

Direct Attack
[MSZ16,ADGM17,CGH17]

Candidates
[GGH13,CLT13, GGH15,CLT15]

Zeroizing Attacks
[GGH13,CHL15, GHMS14,BWZ14, CGH15]

What objects and assumptions imply IO?



Degree-L of GES

Support evaluation of degree-L polynomials

- More secure
- Potentially easier to find algebraic instantiation

What power do low-deg GES have?

Also, weaker functionality and security

Deg-poly GES [GGHRSW13….]

- Ideal model ~ Uber assump [PST14]
- MSEA [GLSW14,GGHZ16 + AJ15/BV15]

Deg-O(1) GES [Lin16]

- DDH-like – joint-SXDH [LV16]

Deg-5 GES [AS16]

- Close to ideal model security

Deg-5 MMap [Lin16b]

- *SXDH*

Deg-3 MMap [LT17]

- *SXDH*

Bilinear Map

Locality O(1) PRG

Locality 5 PRG

Locality 4 PRG
Impossible [MST03]

Block Locality 3 PRG

Block Locality 2 PRG
Impossible [LV17,BBKK17]

(Asymmetric) Multilinear Maps [BS03,Rot13]

Encode in group G_L: $[a]_L = g_L^a$

Multiply: $[a_1]_1 \cdots [a_i]_i \cdots [a_j]_j \cdots [a_D]_D$

$\otimes$

$[a_1 \ldots a_D]_{D+1}$

Degree d
(d = 2, Bilinear map)

Add/Sub: $[a_i]_L$ $[a_j]_L$

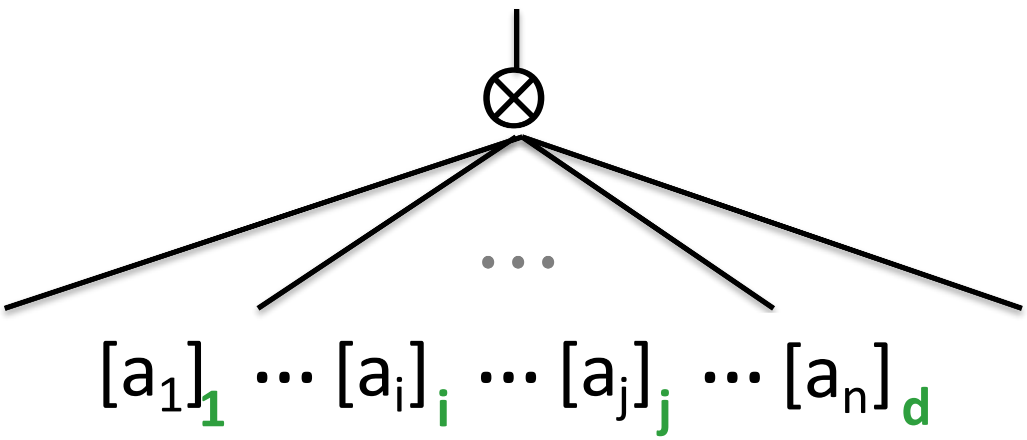
$\oplus$

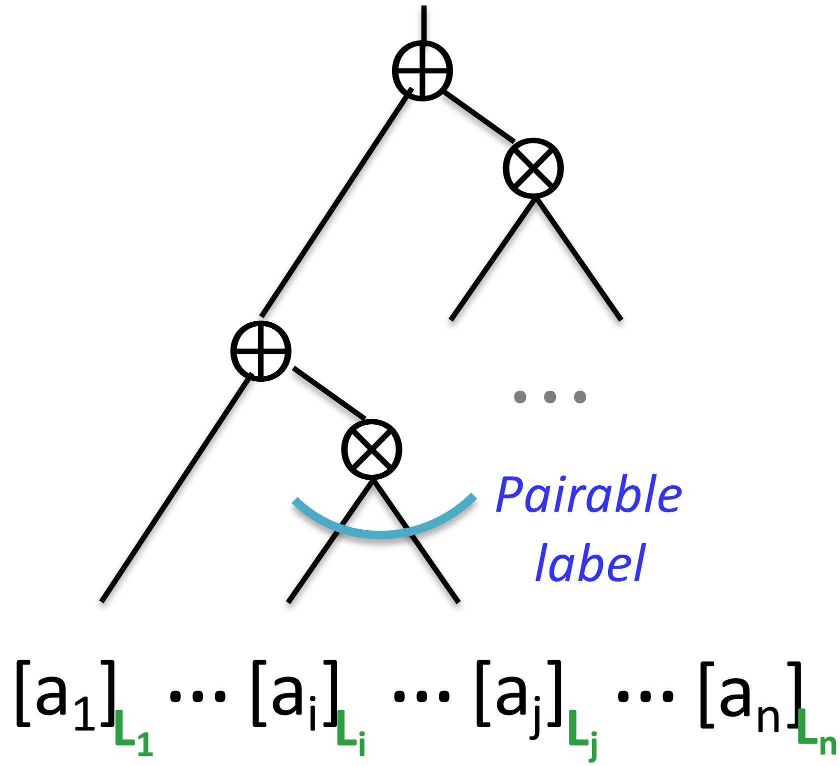
$[a_i + a_j]_L$

Zero-Test (ZT):

$ZT([a]_L) = 1$
iff $a = 0$

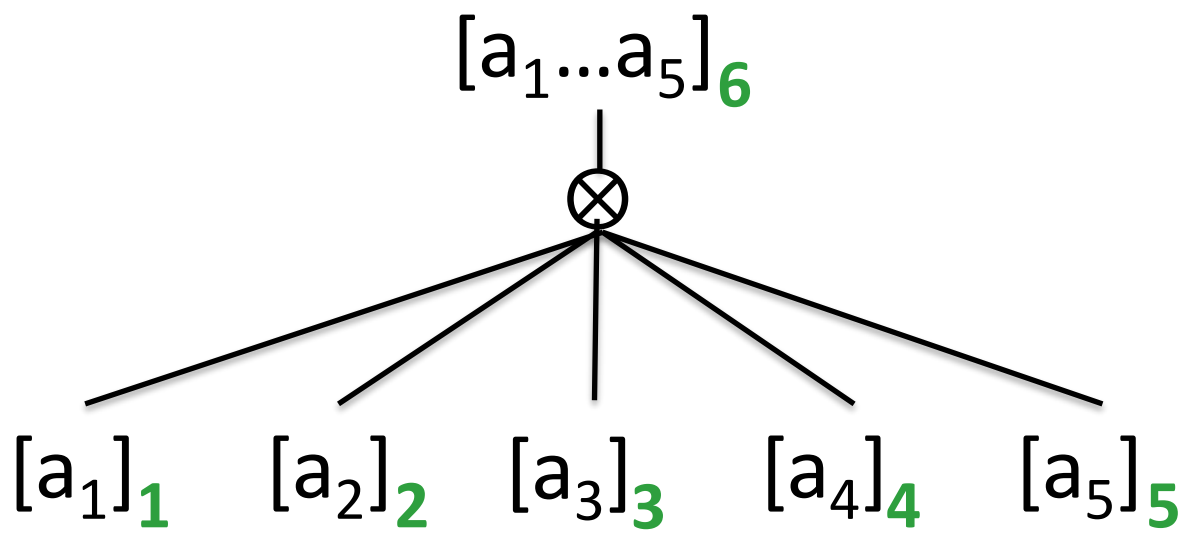
Deg-d Mmap vs Deg-d GES





GES is stronger

- Functionality-wise: Support evaluation of more polynomials
- Security-wise: More control on what polynomials are prohibited



SXDH: Classical DDH on EACH label source group

$$\{\ [a]_L \quad [b]_L \quad [ab]_L\ \}$$

$$\approx \qquad a, b, r \leftarrow R$$

$$\{\ [a]_L \quad [b]_L \quad [r]_L\ \} \quad \text{given } \{\ [1]_{L'} \}_{L' \in [5]}$$

* In the rest of the talk, implicitly assume sub-exp security

Main Theorem [Lin16b]:

∃ a construction of (sub-exp secure) IO from d-linear MMaps assuming

- sub-exp SXDH on d-linear MMaps
- sub-exp locality-d PRG
- sub-exp LWE

Block Locality d PRG [LT17]

Blockwise Local PRGs

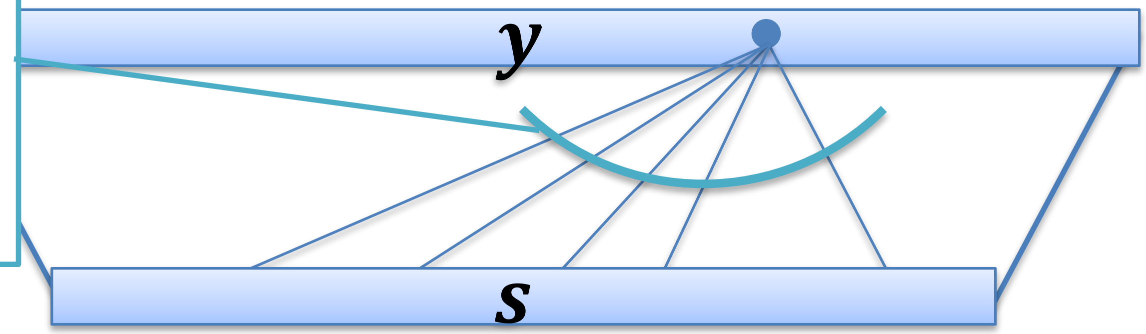
Locality L

[CEMT09, BQ12, OW14, AL16]

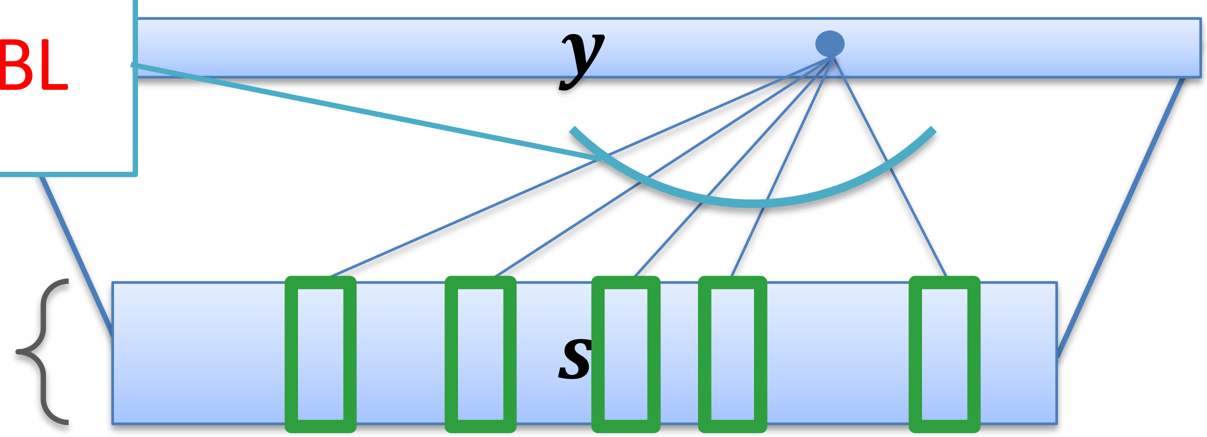
L = 5 [Gol00,MST03,OW14]

L < 5 impossible [MST03]

Polynomial stretch $|y| = |s|^{1+\alpha}$



Blockwise Locality BL



Block size $\mathrm{b} \leq \log(\lambda)$

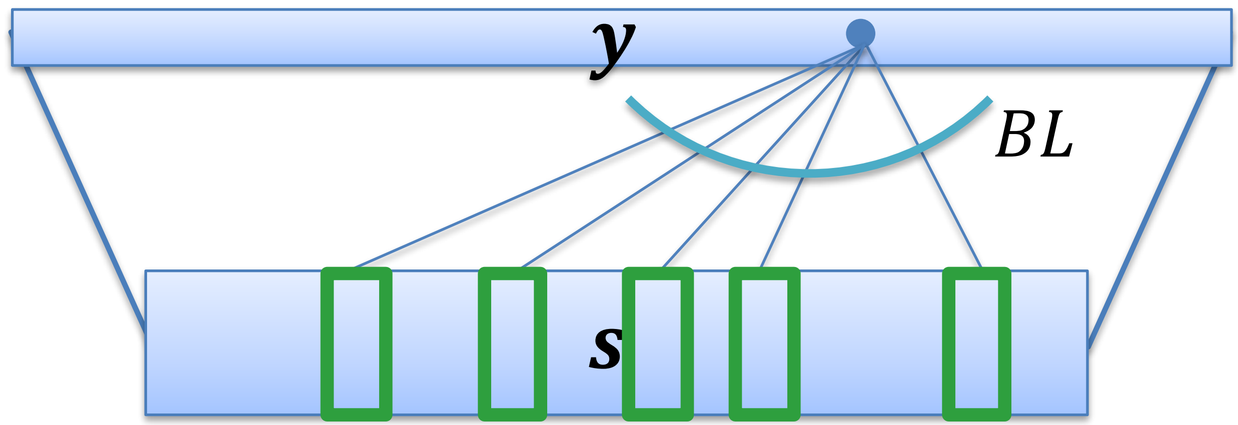
Input Blocks

Preliminary Study on Blockwise Local PRGs

Goldreich's local functions with input bits replaced by blocks

$F_{G,\vec{P},b}$:

$y_i = P_i(s_{G(i)})$



When BL = 2,

deg-BL bipartite graph, $P_i : \{0,1\}^{b \times BL} \rightarrow \{0,1\}$

graph G with good expansion (unique neighbor expansion)

Do Not Exist :(

1. ∃ Blockwise Locality-2 Small Bias Generators *

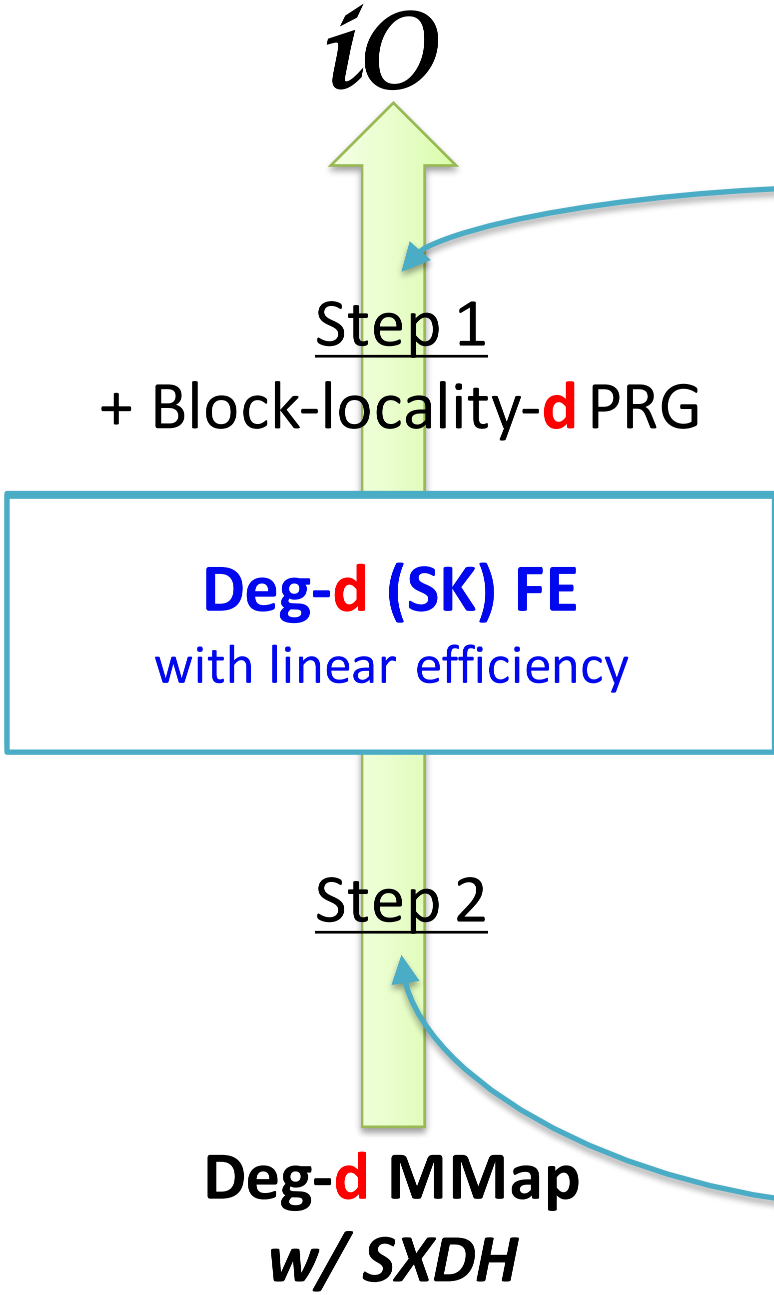
2. For large b, a good expander G, and suitable P, $F_{G,P,b}$ is a ***high-locality*** function with ***good expansion***, assumed to be OW and pseudorandom by [AR16]

Cool Attacks, next Talk

* Small window of expansion, not known to be sufficient for IO

3. Hardness amplification

Construction



Degree Preserving Bootstrapping [LT17]

- Key Idea: Preprocessing ! [Lin16b,AS16,LT17]

Degree Preserving FE Construction [Lin16b]

- Key Idea: Recursively Compose IPE [ABDP15]

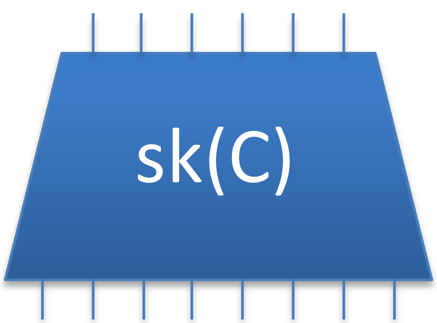
Functional Encryption

--- Encryption with partial decryption keys

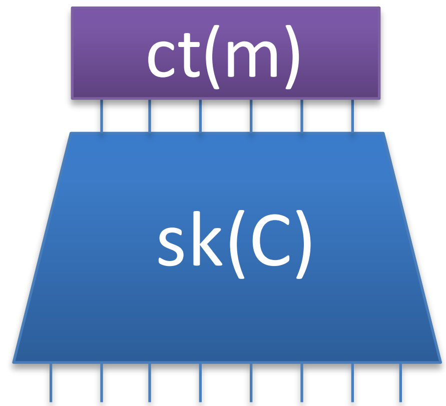
msk ← Setup(1^n)

Enc(msk, m):

Keygen(msk, C):



Dec(sk_C, ct):



y = C(m)

Semantic Security:

$\forall, \vec{m}^1, \vec{m}^2$

$\vec{C}$, s.t. $C_j(m_i^1) = C_j(m_i^2)$

$$\{ ct(m_i^1) \}_i \{ sk(C_j) \}_j$$
$$\approx \{ ct(m_i^2) \}_i \{ sk(C_j) \}_j$$

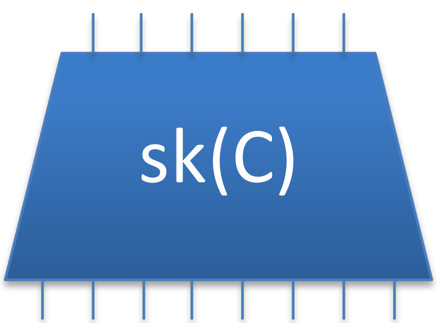
Functional Encryption

msk ← Setup(1^n)

Enc(msk, m):

ct(m)

Keygen(msk, C):



Non-Compact:

$$T_{Enc} = poly(\lambda, |m|, |C|)$$

FE for NC^1

(Weakly) Compact

$$T_{Enc} = poly(\lambda, |m|)\,|C|^{1-\varepsilon}$$

Deg-d FE

Linear efficiency

$$T_{Enc} = poly(\lambda)\,|m|$$

Bootstrapping

iO

[BNPW16]

1-key FE for **NC**^{1}
compact

+ Block-locality-**d** PRG

deg-d FE
w/ linear efficiency

1. **Deg(FE) = 3 L + 1** [Lin16,LV16]
   Bootstrapping via Randomized Encoding + local PRG

2. **Deg(FE) = L** [Lin16b,AS16]

3. **Deg(FE) = BL** [LT17]

Randomized Encodings (RE) [AIK04]

Represent
a "**complex**" computation, by a "**simpler**" computation

$$f(x) = y \qquad \mathrm{RE}_f(x;r) = \Pi$$

$\in \mathrm{NC}^1$ $\qquad$ $\in \mathrm{NC}^0_4$

$\mathrm{Deg}(\mathrm{RE_f}) = 4$

Importantly,
deg 1 in **x** and deg 3 in **r**

s.t., $\boldsymbol{\Pi}$ encodes y, and reveals nothing else

Use RE for Low-Deg Computation
[Lin16, LV16]

Goal: 1-key FE for NC^1
compact

$\mathrm{CT}(x)$

$\mathrm{SK}(f)$

Tool: Deg-**d** FE
linearly efficient

$\mathrm{CT}(x, r)$

$\mathrm{SK}(\mathrm{RE}_f)$

Compact?

$T_{\mathrm{Enc}} = \cancel{\mathrm{poly}(\lambda)}|x, r|$

$\leq \cancel{\mathrm{poly}(\lambda)}|f|^{1-\epsilon}$

Deg(FE) $= \mathrm{Deg}(\mathrm{RE_f})$
$= 4$

Need, Randomness Efficient RE $|r| < |f|^{1-\epsilon}$

Use PRG for Randomness Efficiency
[Lin16, LV16]

$r = \text{PRG}(s)$

$g(x, s) = \text{RE}_f(x\,; \text{PRG}(s))$

Compact

$s = r^{\frac{1}{1+\alpha}}$

w.l.o.g. RE has $r = O(|f|)$

➔

$T_{\text{Enc}} = |x, s| = |f|^{1-\epsilon}$

Tool: deg-**d** FE
linearly efficient

$\text{CT}(x, s)$

$\text{SK}(g)$

Deg(FE) = $\text{Deg}(g)$
$= 3\text{Deg}(\text{PRG}) + 1$
$\leq 3\,\text{L} + 1$

Bootstrapping [LT17]

iO

[BNPW16]

1-key **FE for** **NC**^{1}
compact

\+ Block-locality-**d** PRG

deg-d **FE**
w/ linear efficiency

1. **Deg(FE) = 3 L + 1** [Lin16,LV16]
   Bootstrapping via Randomized Encoding + local PRG

2. **Deg(FE) = L** [Lin16b,AS16]
   Preprocessing at Encryption time

3. **Deg(FE) = BL** [LT17]

Preprocessing [Lin16,LV16]

Goal: Decompose g into A, B s.t.

$g(x, s) = \mathrm{RE}_f(x\,; \mathrm{PRG}(s))$

$= B(\,A(x, s)\,)$

$\mathrm{CT}(A(x, s))$

$\mathrm{SK}(B)$

Compact

$\mathrm{T_{Enc}} = |A(x, s)| \leq |f|^{1-\epsilon}$

Deg(FE) = $\mathrm{Deg}(B) \leq \mathrm{L}$

Pro-process Multiplication with x [Lin16,LV16]

Recall: RE has deg 1 in x & deg 3 in r

$g(x,s) = \mathrm{RE}_f(x\,;\mathrm{PRG}(s))$

$= \sum x_i \mathrm{PRG_j}(s)\mathrm{PRG_k}(s)\mathrm{PRG_l}(s)$

Compact

$\mathrm{T_{Enc}} = |A(x,s)|$

$|x, s, x \otimes s| \le \mathrm{poly}(\lambda)|f|^{1-\epsilon}$

$\mathrm{CT}\left(\begin{matrix} A(x,s) = \\ x, s, x \otimes s \end{matrix}\right)$

$\mathrm{SK}(B)$

s.t. $B(A(x,s)) = g(x,s)$ &

Deg(FE) = $\mathrm{Deg}(B)$

$\le \mathrm{Deg}(g) - 1$

Pro-process Multiplication between s [Lin16,LV16]

Recall: RE has deg 1 in x & deg 3 in r

$g(x, s) = \mathrm{RE}_f(x\,; \mathrm{PRG}(s))$

$= \sum x_i \mathrm{PRG_j}(s)\mathrm{PRG_k}(s)\mathrm{PRG_l}(s)$

$\mathrm{CT}(\ A(x,s) = x, s, s \otimes s\ )$

$\mathrm{SK}(B)$

s.t. $B(A(x,s)) = g(x,s)$ &

Compact

$\mathrm{T_{Enc}} = |A(x,s)|$

$|x, s, s \otimes s| \le |f|^{1-\epsilon} \times |f|^{1-\epsilon}$

Deg(FE) = $\mathrm{Deg}(B)$

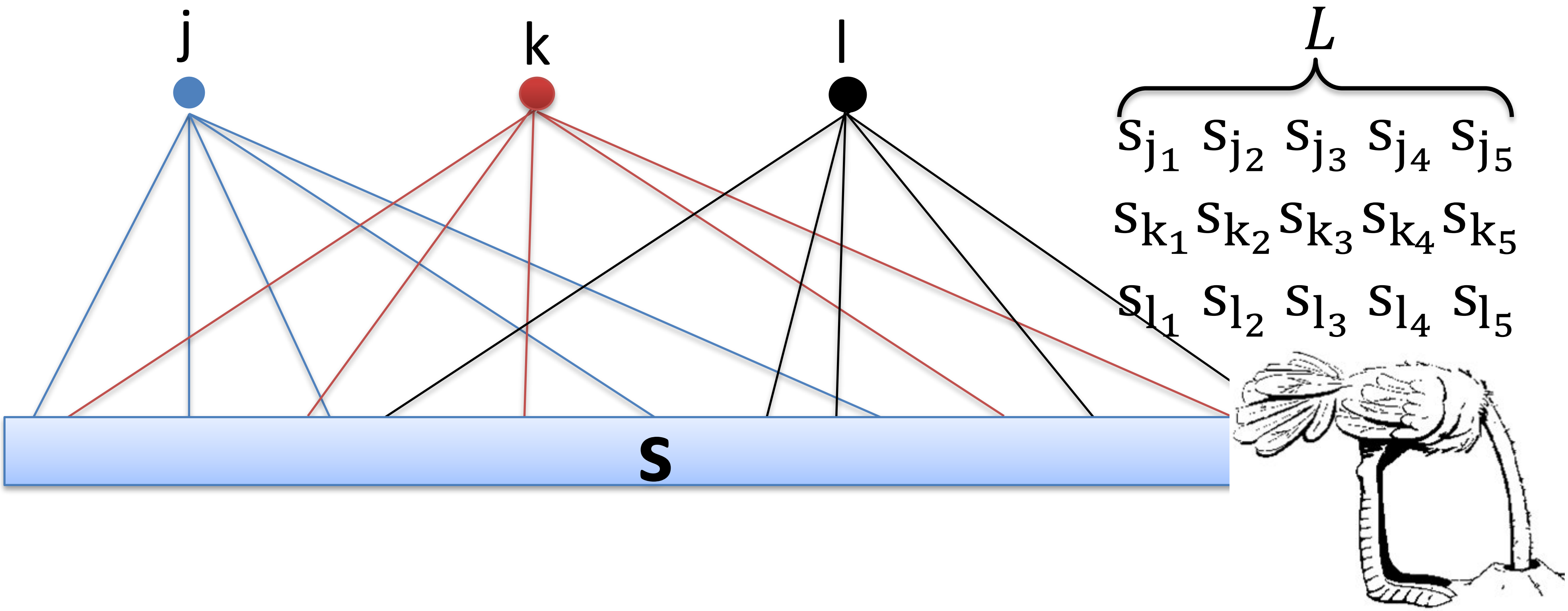
$\le \mathrm{Deg}(g) - 1$

$$g(x, s) = \mathrm{RE}_f\big(x\,;\mathrm{PRG}(s)\big)$$

$$= \sum x_i \mathrm{PRG}_{\mathrm{j}}(s)\mathrm{PRG}_{\mathrm{k}}(s)\mathrm{PRG}_{\mathrm{l}}(s)$$

✘ Multiplication b/w random edges

Hard to pre-compute compactly

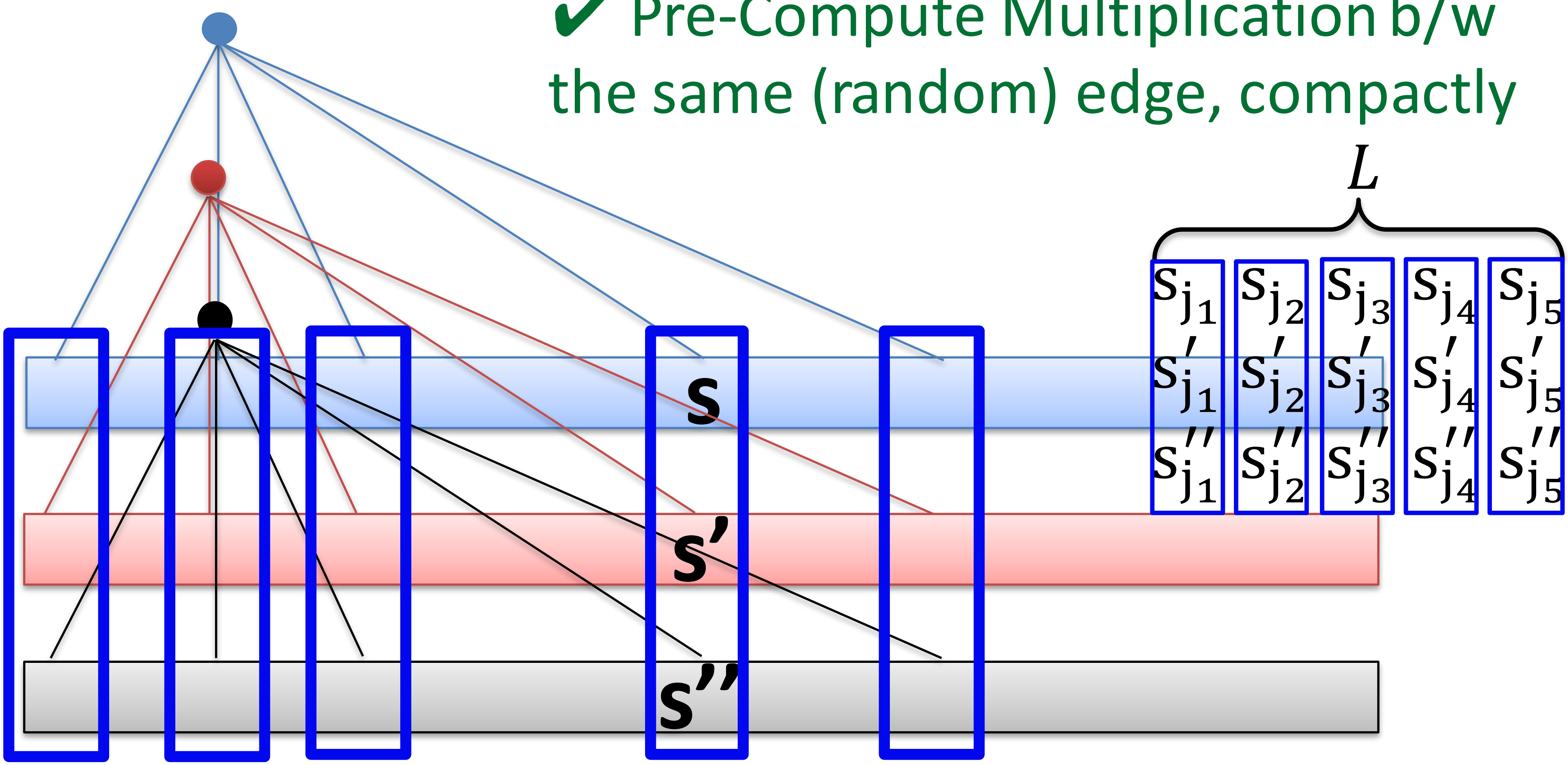


$$g(x, s) = \mathrm{RE}_f\big(x\,;\mathrm{PRG}(s)\big)$$

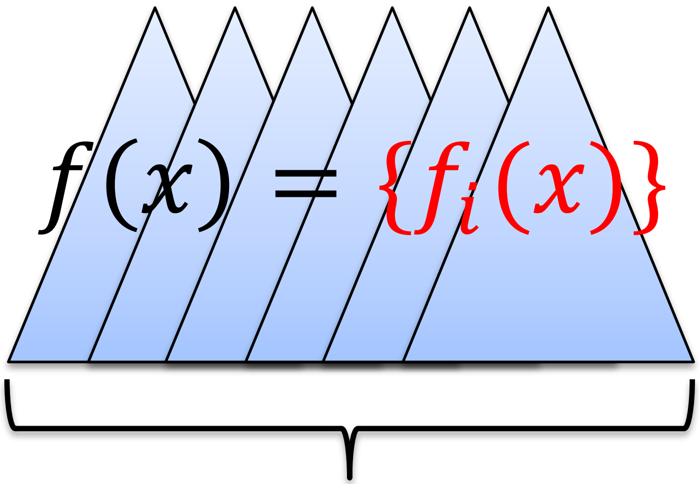
Massage g into this form

What can we pre-compute?

✔ Pre-Compute Multiplication b/w the same (random) edge, compactly

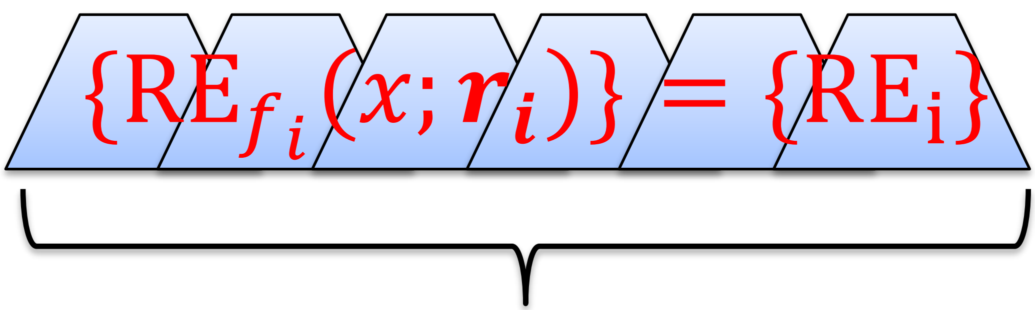


$g(x, s) =$
$\mathrm{RE}_f\big(x\,;\mathrm{PRG}(s)\big)$



w.l.o.g $\quad |f_i| = \mathrm{poly}(\lambda)$

o.w. $\quad f = \mathrm{Yao}(f, x; \mathrm{PRF}(k))$



Then $\quad |\boldsymbol{r_i}| = \mathrm{poly}(\lambda)$

& multiplication within r_i

$r_{ij} r_{ik} r_{iq}$

$$\{\mathrm{RE}_{f_i}(x; \boldsymbol{r_i})\} = \{\mathrm{RE_i}\}$$

$$m \approx |f|$$

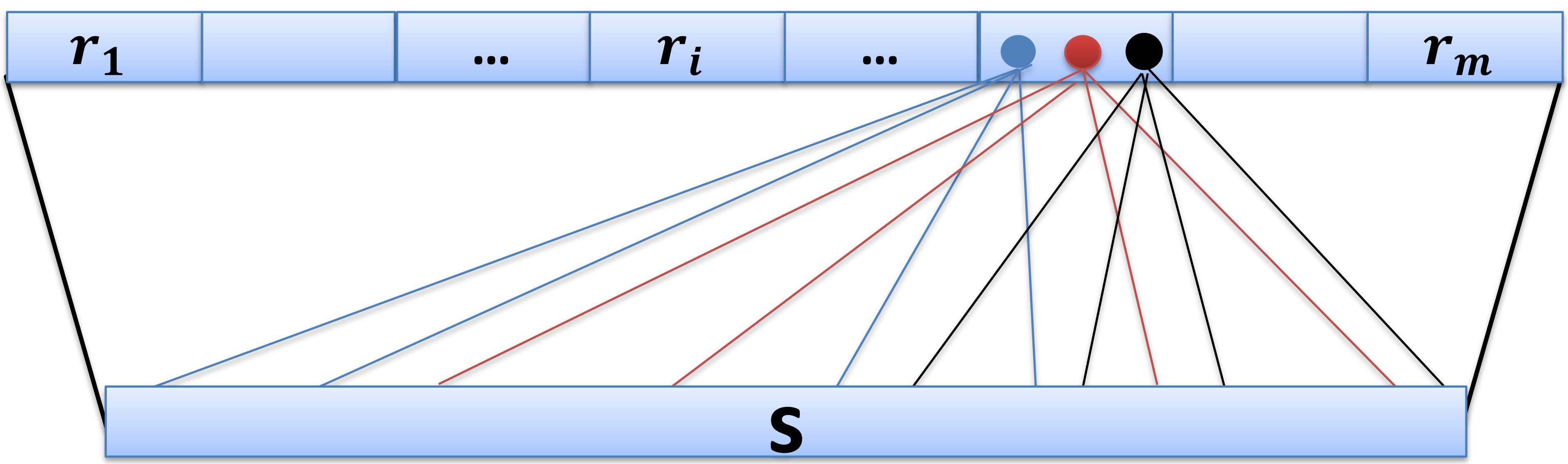
$$|\boldsymbol{r_i}| = \mathrm{poly}(\lambda)$$

multiplication within r_i

$r_{ij}r_{ik}r_{iq}$

So far, $\boldsymbol{r_i}$ = **i$^{\text{th}}$ portion of $\mathbf{PRG}(\boldsymbol{s})$**

✘ Multiplication b/w random edges



$$\{\mathrm{RE}_{f_i}(x; \boldsymbol{r_i})\} = \{\mathrm{RE}_\mathrm{i}\}$$

$$m \approx |f|$$

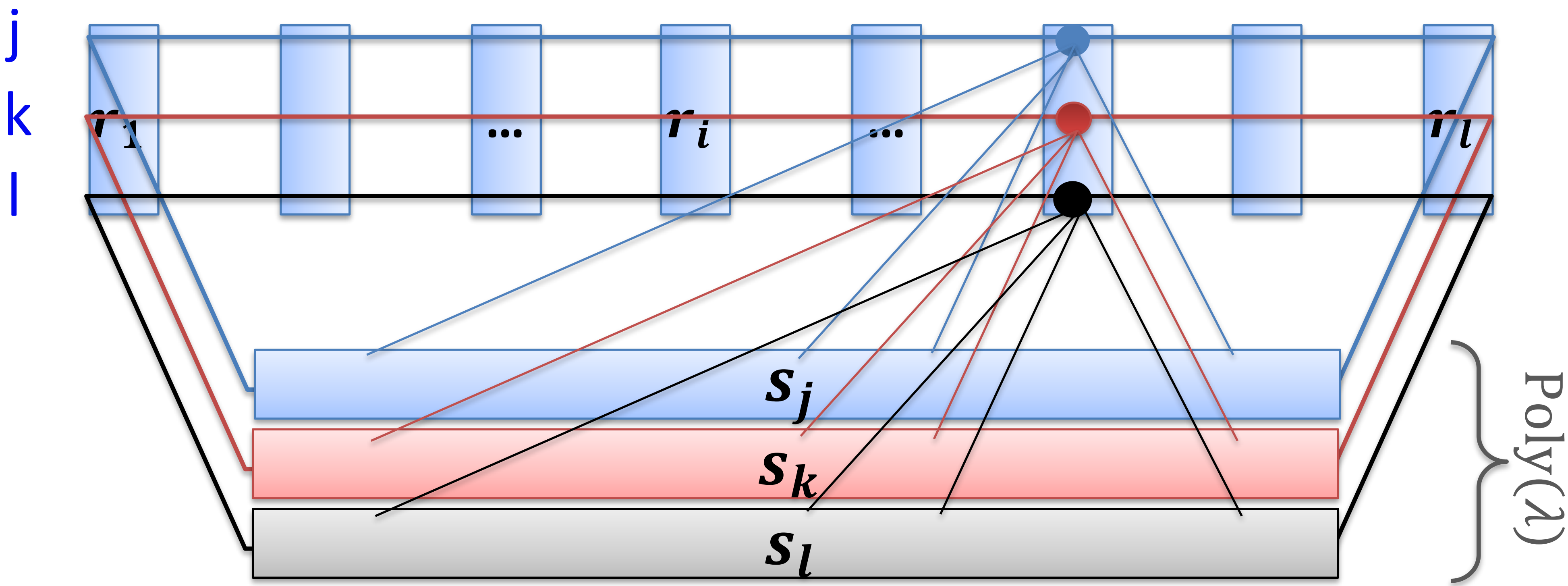
$$|\boldsymbol{r_i}| = \mathrm{poly}(\lambda)$$

multiplication within r_i

$$r_{ij} r_{ik} r_{iq}$$

Now, $\{\boldsymbol{r_{1j} \ldots r_{ij} \ldots r_{mj} = \mathrm{PRG}(s_j)}\}_j$

✔ Multiplication b/w same edges



Pre-compute deg-3 monomials over each column

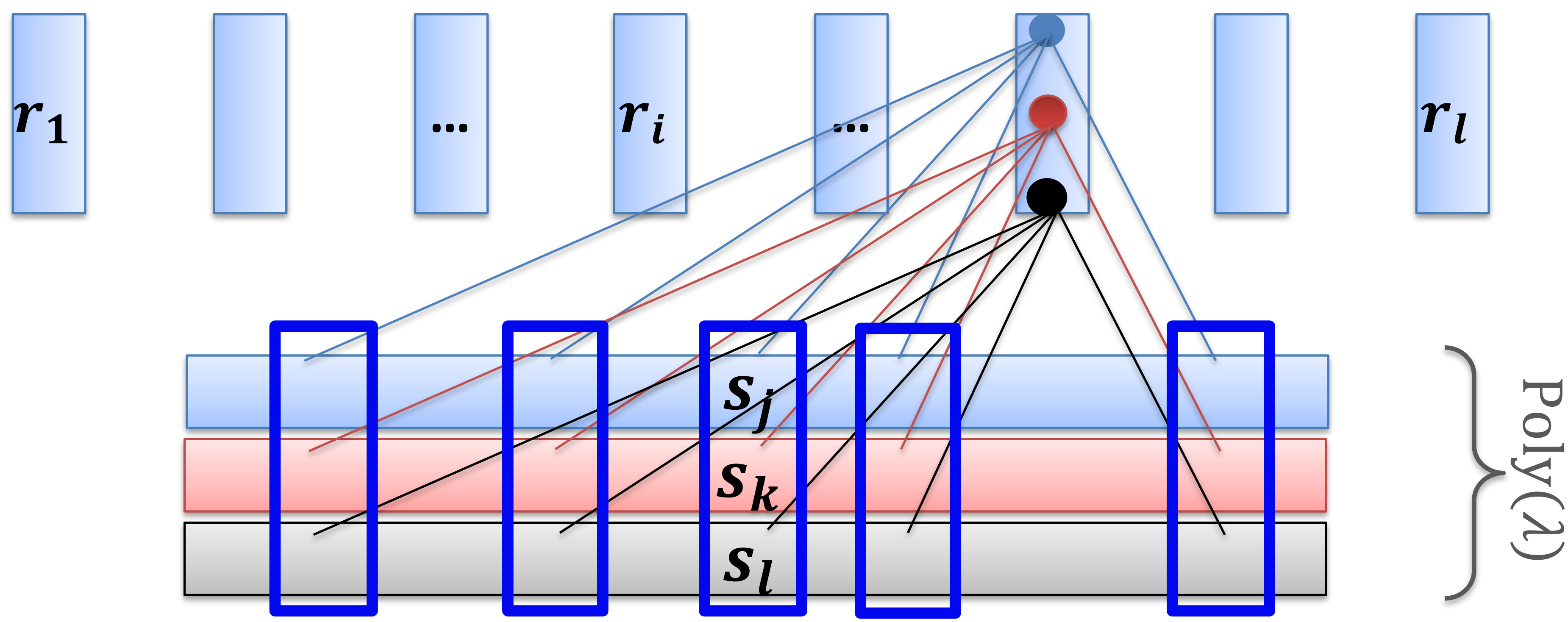
So that, $r_{ij}r_{ik}r_{iq}$ can be computed in degree L

➔ Deg(FE) = L

 # monomials = $|\text{column}|^3 \times |s_j| = \text{poly}(\lambda) f^{1-\epsilon}$

➔ Compactness

✔ Multiplication b/w same edges



Bootstrapping [LT17]

iO

[BNPW16]

1-key **FE for NC^1** compact

+ Block-locality-**d** PRG

deg-d FE w/ linear efficiency

1. **Deg(FE) = 3 L + 1** [Lin16,LV16]
   Bootstrapping via Randomized Encoding + local PRG

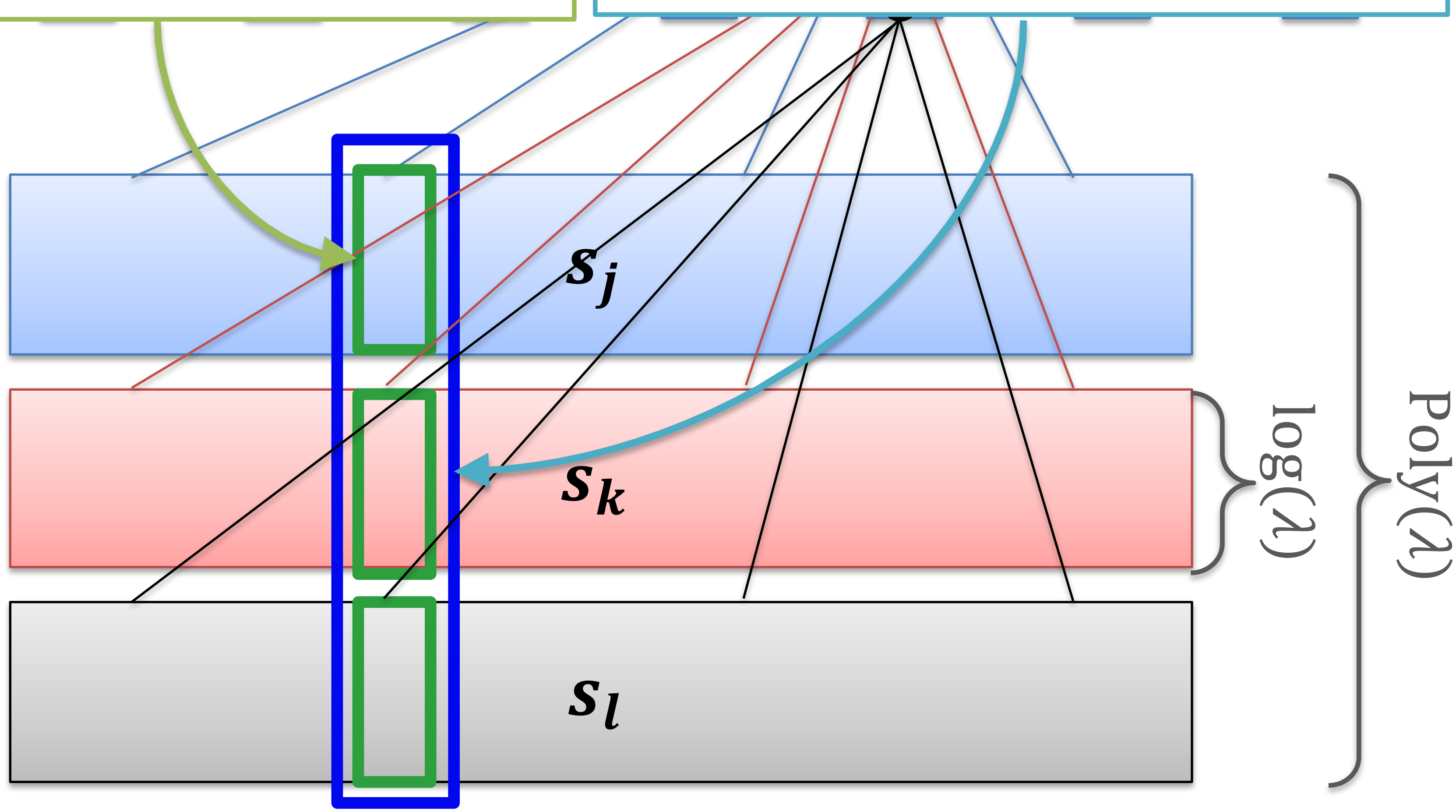
2. **Deg(FE) = L** [Lin16b,AS16]
   Preprocessing at Encryption time

3. **Deg(FE) = BL** [LT17]
   Extend Preprocessing to Block Locality

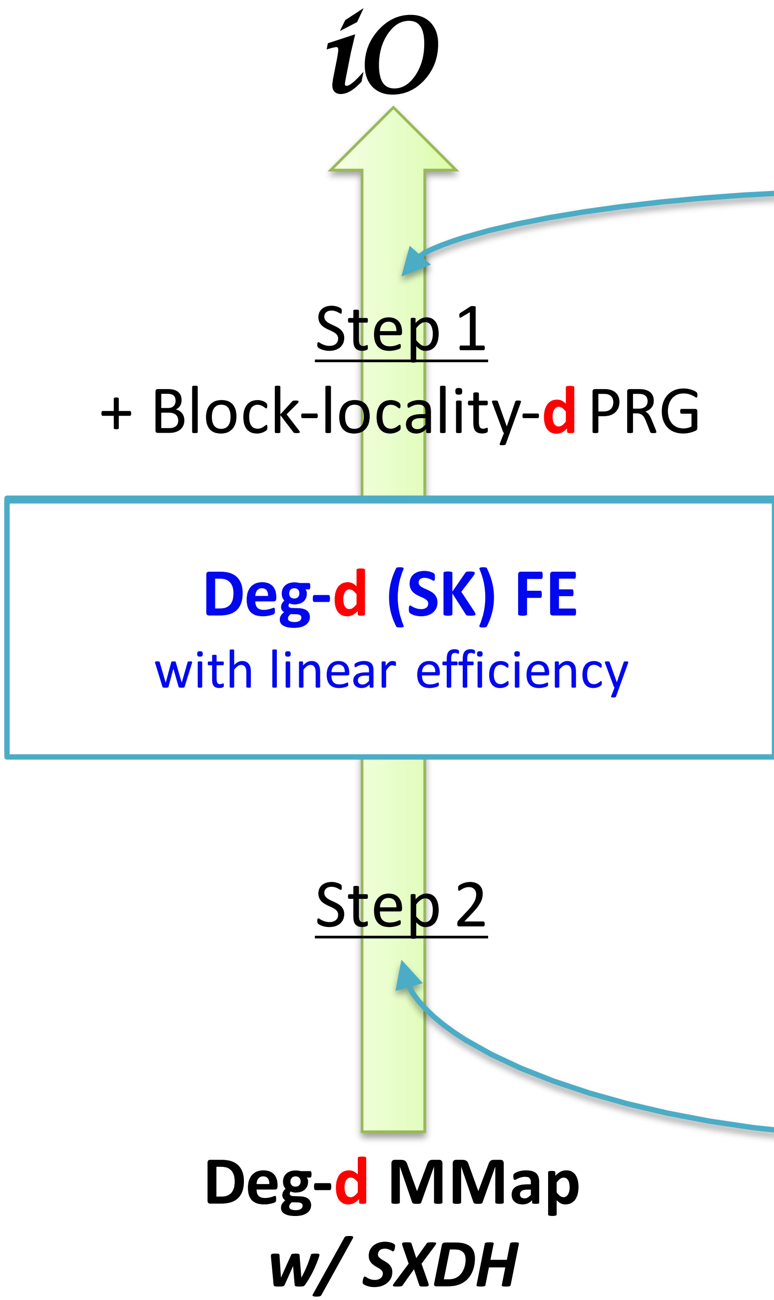
Pre-compute Mult b/w blocks in each column [LT17]

Pre-compute **all** monomials over each block, $2^{\log(\lambda)} = \lambda$, many

Pre-compute deg-3 mnmls over mnmls in each column $\mathrm{poly}(\lambda)^3$, many



Construction



Degree Preserving Bootstrapping [LT17]

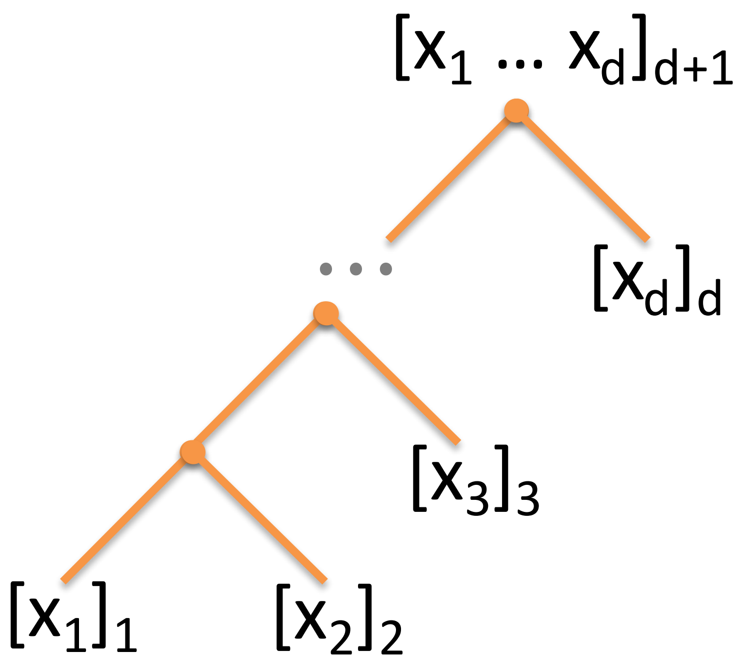
- Key Idea: Preprocessing ! [Lin16b,AS16,LT17]

Degree Preserving FE Construction [Lin16b]

- Key Idea: Compose IPE [ABDP15]

Bird's Eye View: Deg-d FE ← Deg-d Mmap

Suppose we only want to compute $x_1 \ldots x_d$ while hiding x



✔ **Functionality**

✘ **Security**

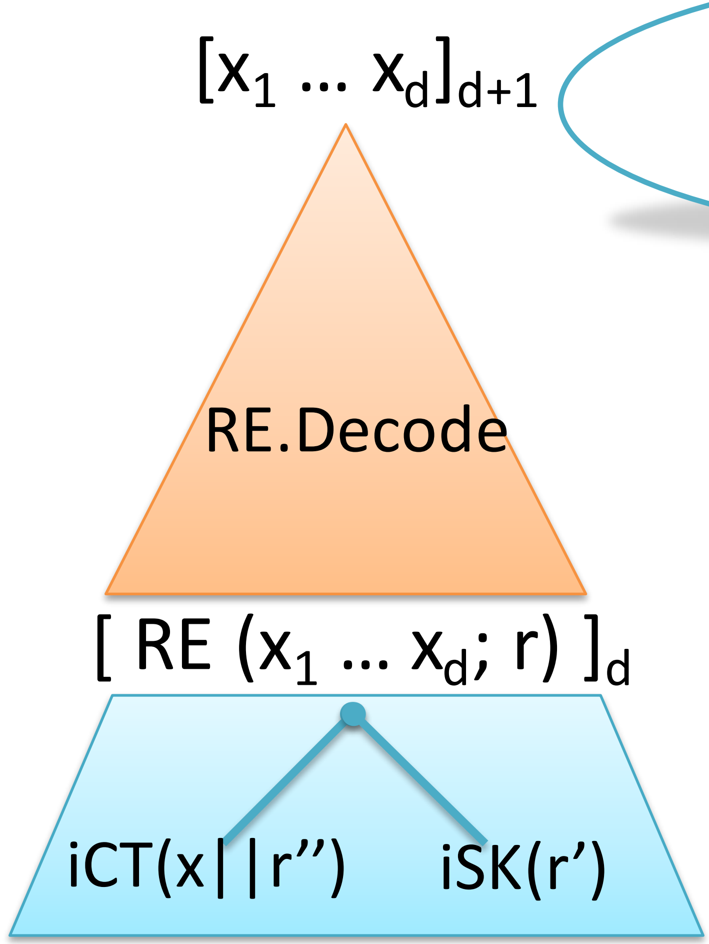
How to hide x?

Bird's Eye View: Deg-d FE ← Deg-d Mmap

Suppose we only want to compute $x_1 \ldots x_d$, while hiding x

[LV16] Security ← RE [AIK14]

+ Function Hiding IPE

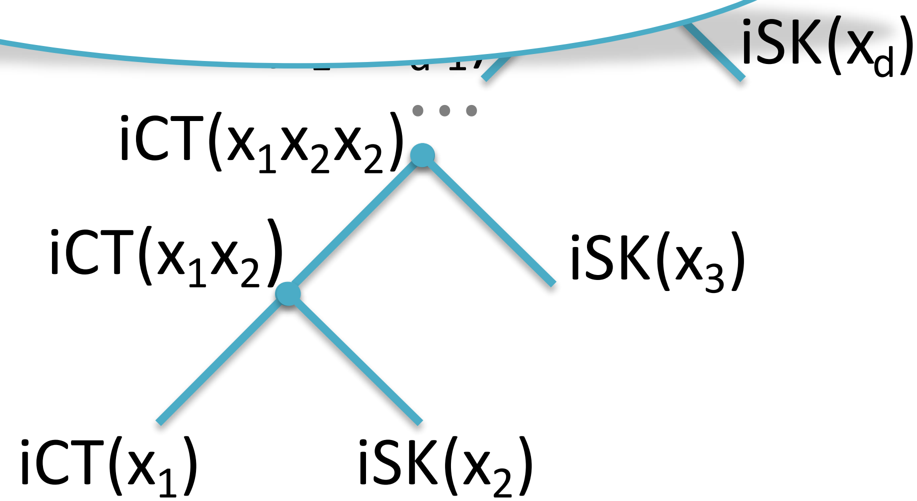


[Lin16b] Security ←

Recursively composing IPE

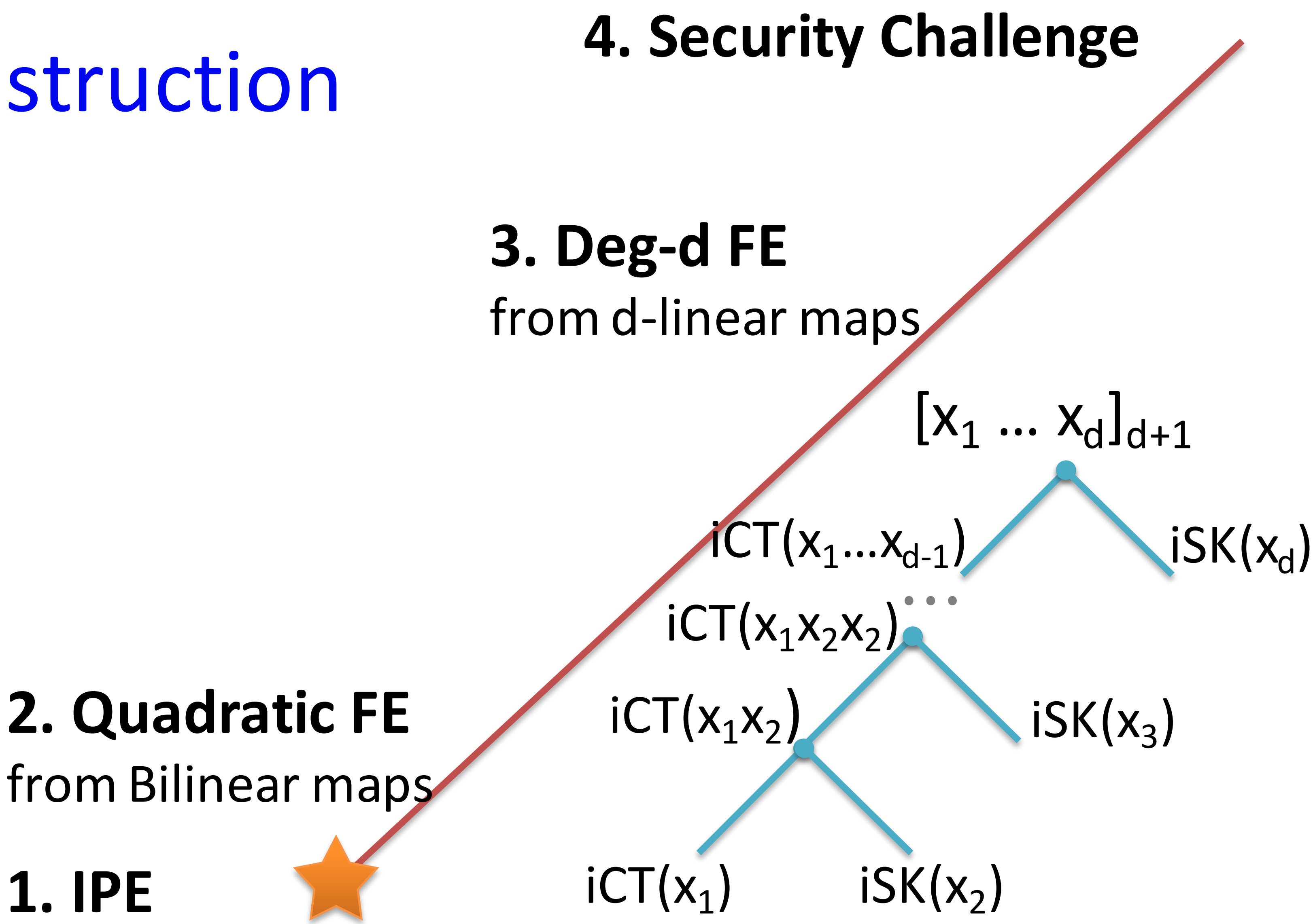
IPE: FE for inner products

← SXDH on Bilinear maps



No Waste of Degree

FE Construction



Inner Product Encryption

iSK(***u***) ─┐
├─ Dec ─ <***u***,***v***>
iCT(***v***) ─┘

Weak Functionality: Test if <*u*,*v*> is zero

- Public-key IPE

  ← *DDH or LWE or Paillier* [ABDP15,ALS16]

- Secret-key IPE, function hiding (i.e, hide both u, v)

  ← *Bilinear map* [KSW08,BJK15+LV16,DDM16]

Canonical Form
FH SK-IPE
[This work]

cSK (***u***)= $[\,U\,]_0$ ─┐
├─ Dec ─ $[\langle u,v\rangle]_2$
cCT(***v***)= $[\,V\,]_1$ ─┘

target group

different source groups

New Simple Construction

The ABDP PK-IPE Scheme from DDH

El Gamal
+ function keys

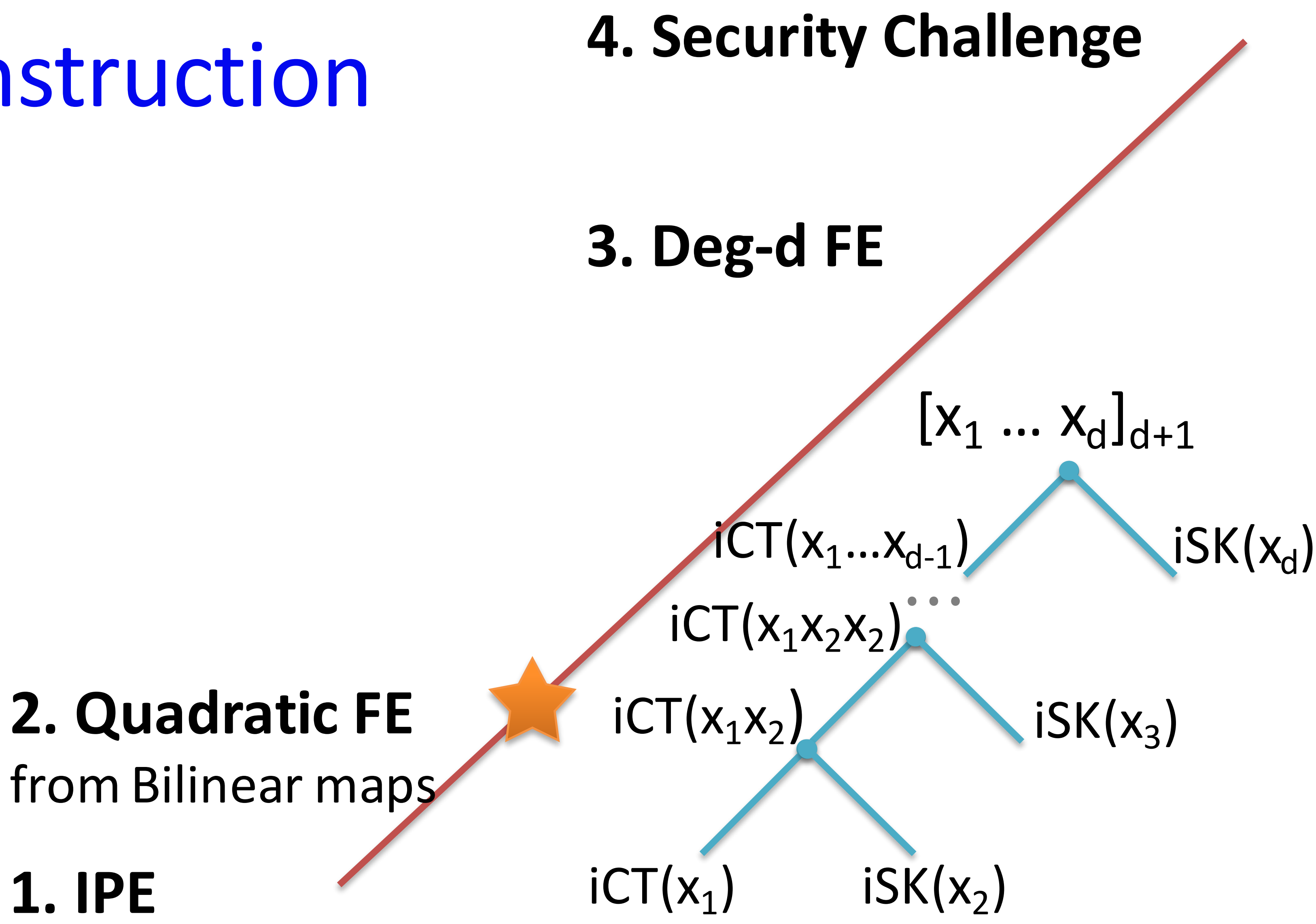
iMSK	:	$\mathbf{s} \leftarrow Z_p^n$	
iMPK	:	$[\mathbf{s}] = g^{\mathbf{s}}$	
iSK($\boldsymbol{y}$)	:	$\langle \mathbf{s}, \mathbf{y} \rangle$	$\mathbf{y}$
iCT($\boldsymbol{x}$)	:	$[-r] = g^{-r}$	$[r\mathbf{s} + \mathbf{x}] = g^{r\mathbf{s}+\mathbf{x}}$

Dec

$$\langle \text{iCT}(\boldsymbol{x}), \text{iSK}(\boldsymbol{y}) \rangle = [-r \langle \mathbf{s}, \mathbf{y} \rangle] + [\langle r\mathbf{s}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{y} \rangle]$$

$$= [\langle \mathbf{x}, \mathbf{y} \rangle]$$

FE Construction



qFE: SK-FE for Quadratic Poly

Starting Point: Compute quadratic polynomials using inner products

$$f(x) = \Sigma C_{ij} x_i x_j = \langle \mathbf{C}, \mathbf{x} \otimes \mathbf{x} \rangle$$

But, $|qCT| = n^2$

qMSK	:	$\mathbf{S} \leftarrow Z_p^{n^2}$	
qSK(f) = iSK(**C**)	:	$\langle \mathbf{S}, \mathbf{C} \rangle$	$\mathbf{C}$
qCT(**x**) = iCT($\mathbf{x} \otimes \mathbf{x}$)	:	[-r]	$[r\mathbf{S} + \mathbf{x} \otimes \mathbf{x}]$

qFE: SK-FE for Quadratic Poly

$f(x) = \sum C_{ij} x_i x_j = < \mathbf{C}, \mathbf{x} \otimes \mathbf{x} >$

Idea: Compress Ciphertext

qMSK : $\mathbf{S} \leftarrow Z_p^{n^2}$

But, $|S| = n^2$

qCT is imcompressble

qSK(f) = iSK(**C**) : <**S**, **C**>

C

qCT(**x**) = iCT($\mathbf{x} \otimes \mathbf{x}$) : [-r]

[r**S** + $\mathbf{x} \otimes \mathbf{x}$]

qFE: SK-FE for Quadratic Poly

$f(x) = \sum C_{ij} x_i x_j = \langle \mathbf{C}, \mathbf{x} \otimes \mathbf{x} \rangle$

Idea: Compress Ciphertext

1. Replace S with $\mathbf{s^1} \otimes \mathbf{s^2}$

Now, iCT depends on $|(r, s^1, s^2, x)| = O(n)$

qMSK	:	$\mathbf{s^1}\ \mathbf{s^2} \leftarrow Z_p^n$	
qSK(f) = iSK(**C**)	:	$\langle \mathbf{s^1} \otimes \mathbf{s^2}, \mathbf{C} \rangle$	**C**
qCT(**x**) = iCT($\mathbf{x} \otimes \mathbf{x}$)	:	[-r]	$[r\ \mathbf{s^1} \otimes \mathbf{s^2} + \mathbf{x} \otimes \mathbf{x}]$

qFE: SK-FE for Quadratic Poly

$f(x) = \sum C_{ij} x_i x_j = \langle \mathbf{C}, \mathbf{x} \otimes \mathbf{x} \rangle$

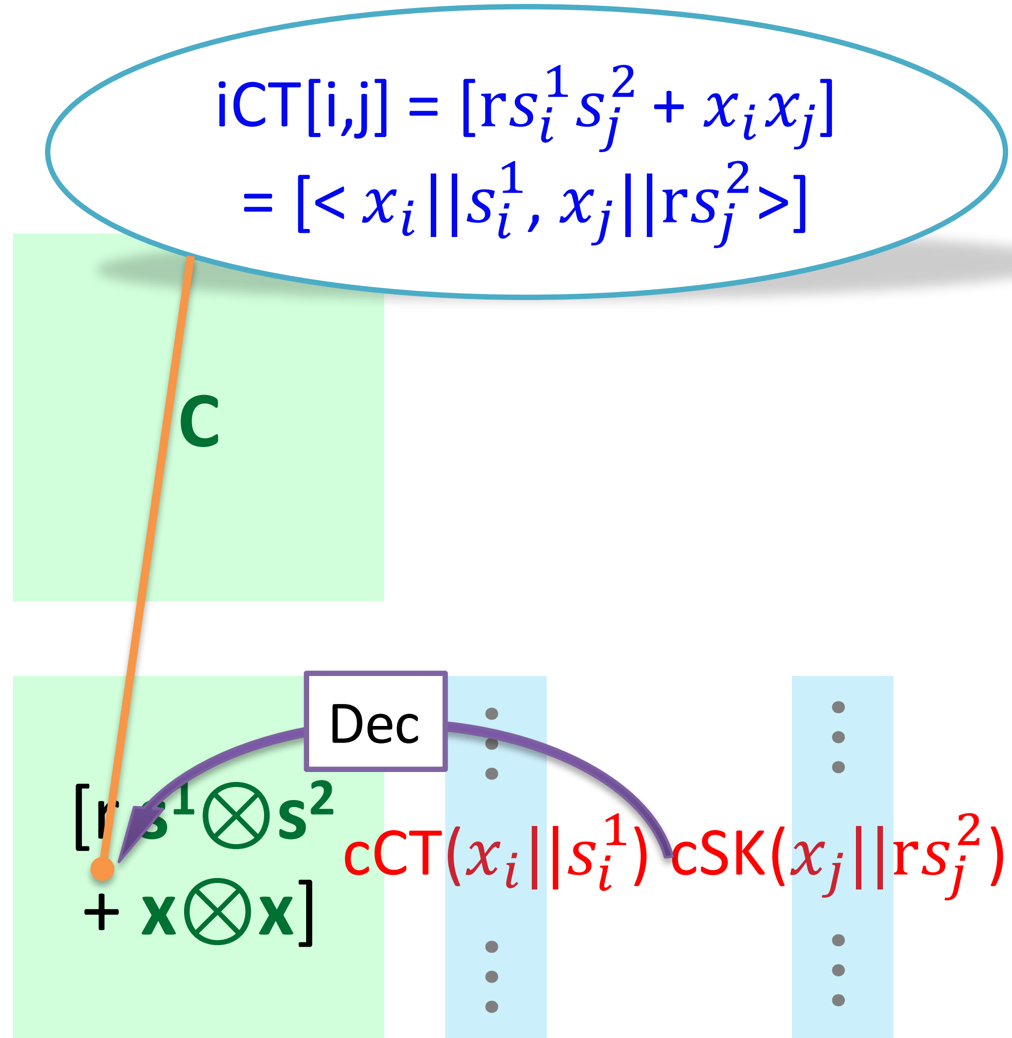
Idea: Compress Ciphertext

1. Replace S with $\mathbf{s^1} \otimes \mathbf{s^2}$
2. Generate iCT using canonical FH IPE

qMSK : $\mathbf{s^1}\ \mathbf{s^2} \leftarrow Z_p^n$

qSK(f) = iSK(**C**) : $\langle \mathbf{s^1} \otimes \mathbf{s^2}, \mathbf{C} \rangle$

qCT(**x**) = iCT($\mathbf{x} \otimes \mathbf{x}$) : [-r]



qFE: SK-FE for Quadratic Poly

$f(x) = \sum C_{ij} x_i x_j = \langle \mathbf{C}, \mathbf{x} \otimes \mathbf{x} \rangle$

Idea: Compress Ciphertext

1. Replace S with $\mathbf{s}^1 \otimes \mathbf{s}^2$
2. Generate iCT using canonical FH IPE

qMSK : $\mathbf{s}^1\ \mathbf{s}^2 \leftarrow Z_p^n$

qSK(f) = iSK(**C**) : $\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{C} \rangle$

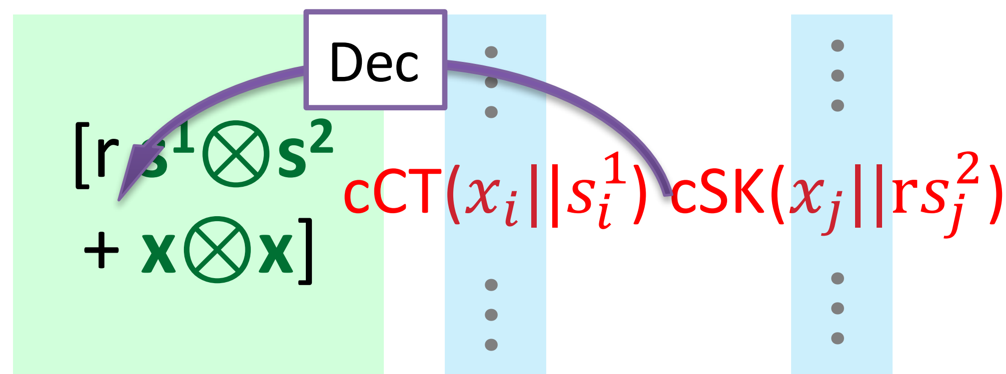
qCT(**x**) : [-r]

✔ Linear Efficiency

|qCT| = n(|cCT|+|cSK|)=O(n)

✔ Security Hope

- ~~FH IPE reveals only iCT~~
- ~~PK IPE Sec → iCT hides x~~



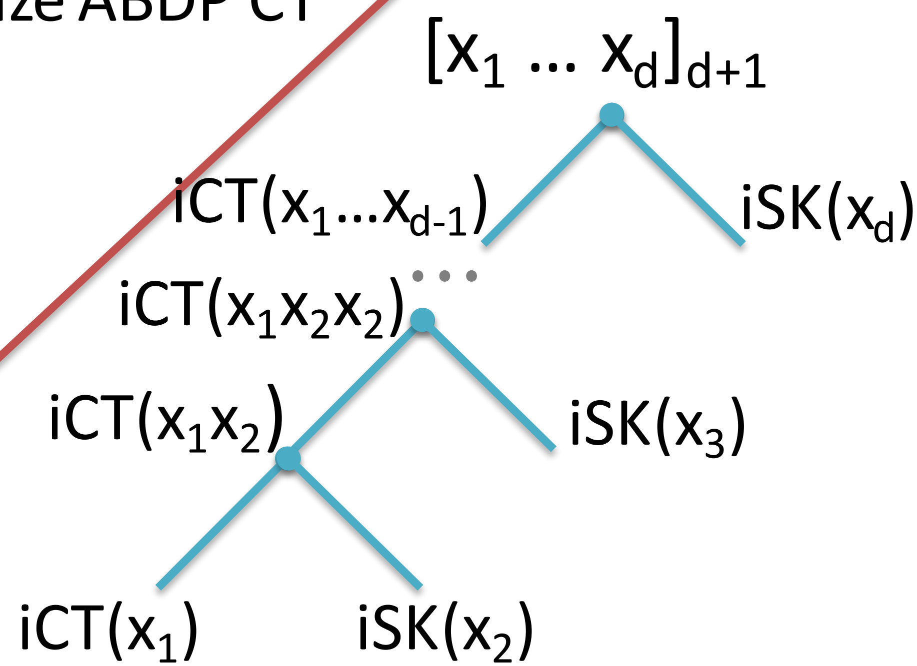
FE Construction

4. Security Challenge

3. Deg-d FE

Generalizing IPE to high degree, deg-d HIPE

Use HIPE to compress N^d-size ABDP CT



2. Quadratic FE

from Bilinear maps

1. IPE

dFE: SK-FE for Deg-d Poly

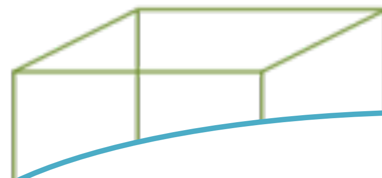
Starting Point: Compute deg-d polynomials using inner products

$$f(x) = \Sigma\, C_{I_1 \dots I_d}\, x_{I_1} \dots x_{I_d} = \langle \mathbf{C}, \otimes\, \mathbf{x}^{\mathbf{d}} \rangle$$

$= \mathbf{x} \otimes \cdots \otimes \mathbf{x}$

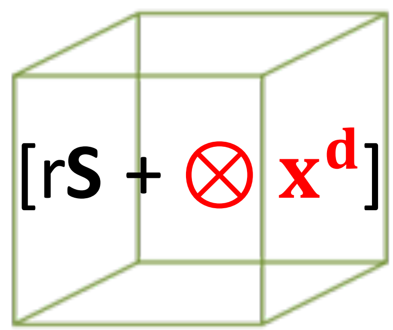
MSK : $\mathbf{S} \leftarrow Z_p^{n^d}$

SK(f) = iSK(**C**) : <**S**, **C**>



But, FH IPE only compresses |CT| to $n^{d/2}$

CT(**x**) = iCT($\otimes\, \mathbf{x}^{\mathbf{d}}$) : [-r]



$[r\mathbf{S} + \otimes\, \mathbf{x}^{\mathbf{d}}]$

New Tool: High-degree IPE (HIPE)

--- Multi-input FE for computing *high-degree inner product*

Deg-d HIPE

hCT^1($\boldsymbol{x}^1$) ... hCT^{d-1}($\boldsymbol{x}^{d-1}$) hSK($\boldsymbol{x}^d$)

Dec

Deg-d Inner Product

$$< \boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^{d-1}, \boldsymbol{x}^d > \;=\; \sum_{i\in[n]} \prod_{j\in[d]} x_i^j$$

Weak Functionality: Test if inner product is zero

Function hiding: Hide $\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^{d-1}, \boldsymbol{x}^d$

New Tool: High-degree IPE (HIPE)

--- Multi-input FE for computing *high-degree inner product*

Canonical

Deg-d HIPE

hCT^1(x^1) = $[X^1]_1$ … hCT^{d-1}(x^{d-1}) = $[X^{d-1}]_{d-1}$ hSK(x^d) = $[X^d]_d$

Dec

Deg-d Inner Product

$$\left[\langle x^1, x^2, \cdots, x^{d-1}, x^d \rangle = \sum_{i\in[n]} \prod_{j\in[d]} x_i^j \right]_{d+1}$$

This Work: Canonical Deg-d HIPE ← SXDH on Deg-d Mmaps

dFE: SK-FE for Deg-d Poly

$f(x) = < \mathbf{C}, \otimes \mathbf{x}^d >$

Idea: Compress Ciphertext

1. Replace S with

$$\otimes \mathbf{s}^{\leq d} = \mathbf{s}^1 \otimes \mathbf{s}^2 \cdots \otimes \mathbf{s}^d$$

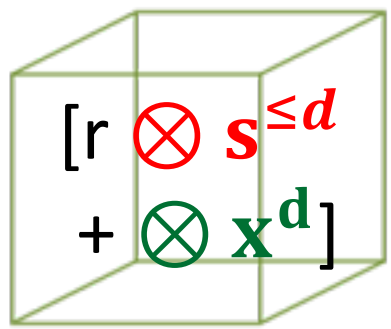
MSK : $\mathbf{s}^1, \mathbf{s}^2, \cdots, \mathbf{s}^d \leftarrow \{0,1\}^n$

SK(f) = iSK(**C**) : $< \otimes \mathbf{s}^{\leq d}, \mathbf{C} >$

iCT depends on

$|(r, s^1, \ldots, s^d, x)| = O(n)$

CT(**x**) = iCT($\otimes \mathbf{x}^d$) : [-r]



dFE: SK-FE for Deg-d Poly

$f(x) = < \mathbf{C}, \otimes \mathbf{x}^d >$

Idea: Compress Ciphertext

1. Replace S with

$$\otimes\, \mathbf{s}^{\le d} = \mathbf{s}^1 \otimes \mathbf{s}^2 \cdots \otimes \mathbf{s}^d$$

2. Generate iCT using canonical FH **HIPE**

MSK : $\mathbf{s}^1 \quad ^2$

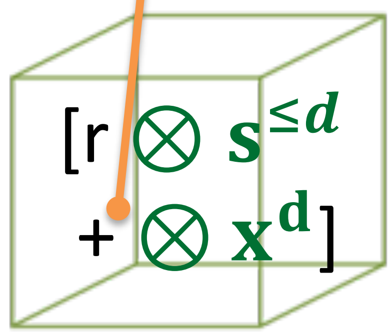
$\mathrm{iCT}[i_1,\cdots,i_d]$

$$= [\, r s^1_{i_1} \cdots s^{d-1}_{i_{d-1}}\, s^d_{i_d} + x_{i_1} \cdots x_{i_{d-1}}\, x_{i_d} \,]$$

$$= [\, < x_{i_1} || s^1_{i_1} \ \cdots \ x_{i_{d-1}} || s^{d-1}_{i_{d-1}},\ x_{i_d} || r s^d_{i_d} > \,]$$

SK(f)
= iSK(**C**)

CT(**x**)
= iCT($\otimes\, \mathbf{x}^{\mathbf{d}}$) : [-r]



dFE: SK-FE for Deg-d Poly

$f(x) = \langle \mathbf{C}, \otimes \mathbf{x}^d \rangle$

Idea: Compress Ciphertext

1. Replace S with

$$\otimes \mathbf{s}^{\leq d} = \mathbf{s}^1 \otimes \mathbf{s}^2 \cdots \otimes \mathbf{s}^d$$

2. Generate iCT using canonical FH **HIPE**

MSK : $\mathbf{s}^1, \mathbf{s}^2$

iCT$[i_1,\cdots,i_d]$

$= [\, r s^1_{i_1} \cdots s^{d-1}_{i_{d-1}} s^d_{i_d} + x_{i_1} \cdots x_{i_{d-1}} x_{i_d} \,]$

$= [\, \langle x_{i_1} || s^1_{i_1} \ \cdots \ x_{i_{d-1}} || s^{d-1}_{i_{d-1}},\ x_{i_d} || r s^d_{i_d} \rangle \,]$

SK(f) = iSK(**C**)

Dec

CT(**x**) : [-r]

$\text{hCT}^1(x_{i_1} || s^1_{i_1}) \cdots \text{hCT}^{d-1}(x_{i_{d-1}} || r s^{d-1}_{i_{d-1}})\ \text{hSK}(x_{i_d} || r s^d_{i_d})$

This work:

Canonical FH Deg-d HIPE ← SXDH on Deg-d MMaps

Recursion: *Canonical* **Deg-d HIPE ←** *Canonical* **Deg-d-1 HIPE + ** *Canonical* **FH IPE**

Deg-d Inner Product

$$\langle x^1, x^2, \cdots, x^{d-1}, x^d \rangle = \sum_{i\in[n]} \prod_{j\in[d]} x_i^j$$

$$= \langle x^1 \times x^2 \times \cdots \times x^{d-1} \;,\; x^d \rangle$$

coordinate-wise multiplication

Compute using Deg-(d-1) HIPE

Compute using FH IPE

Idea: *Canonical* Deg-d HIPE ← *Canonical* Deg-d-1 HIPE + *Canonical* FH IPE

Use deg-d-1 HIPE to compute deg-d-1 product

hCT¹(x_1^1) … hCT^d-2^(x_1^{d-2}) hSK(x_1^{d-1})	Dec	$[x_1^1 \cdots x_1^{d-1}]$
hCT¹(x_i^1) … hCT^d-2^(x_i^{d-2}) hSK(x_i^{d-1})	Dec	$[x_i^1 \cdots x_i^{d-1}]$
hCT¹(x_n^1) … hCT^d-2^(x_n^{d-2}) hSK(x_n^{d-1})	Dec	$[x_n^1 \cdots x_n^{d-1}]$

=

$[\mathbf{x}^1 \times \mathbf{x}^2 \times \cdots \times \mathbf{x}^{d-1}]$

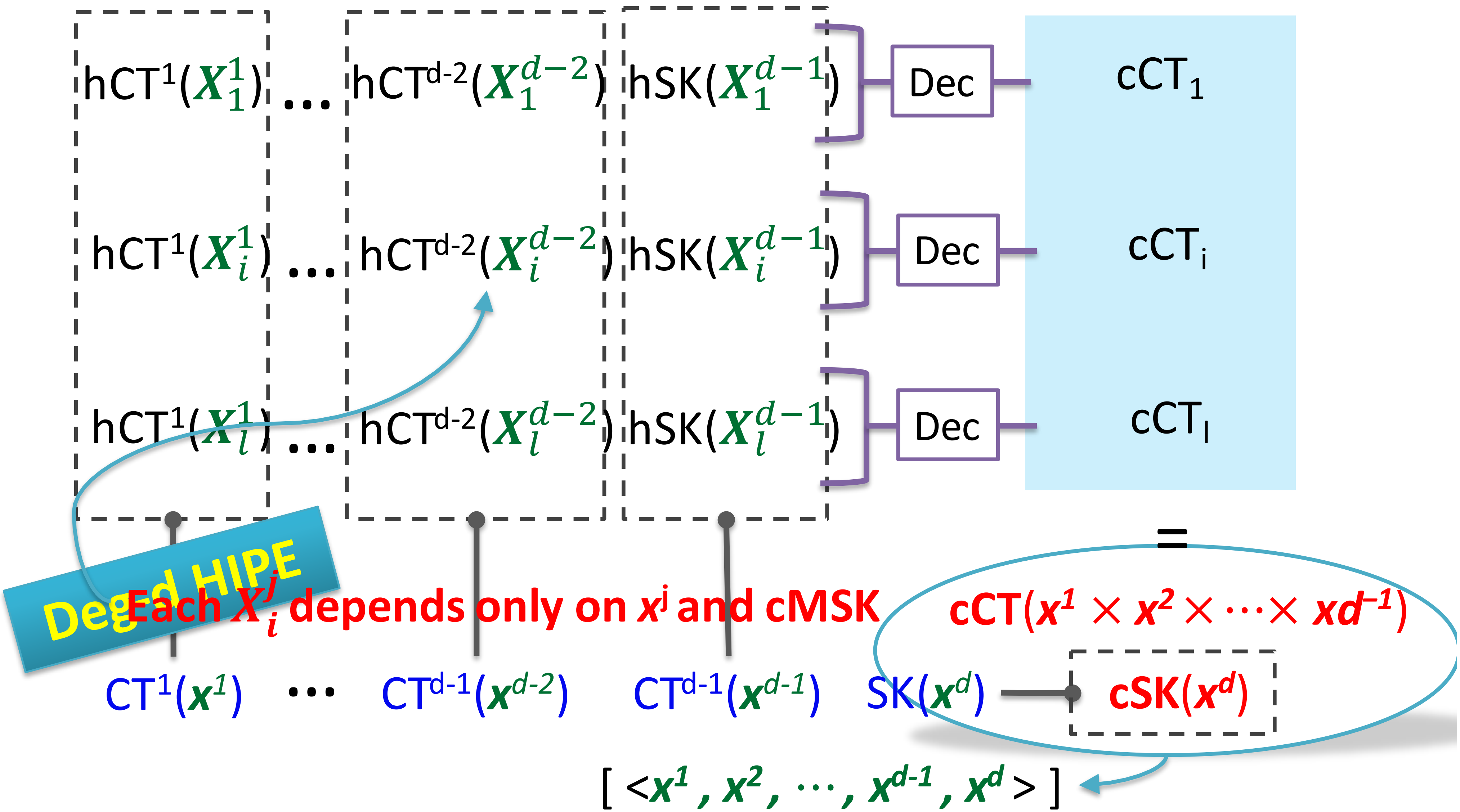
Encrypt $\mathbf{x}^1, \cdots \mathbf{x}^{d-2}, \mathbf{x}^{d-1}$ **one by one** using deg-d HIPE

- Each row i with an **independent $hMSK_i$**

Canonical *Canonical* *Canonical*

Idea: Deg-d HIPE ← Deg-d-1 HIPE + FH IPE

Use deg-d-1 HIPE to compute cCT(deg-d-1 product)



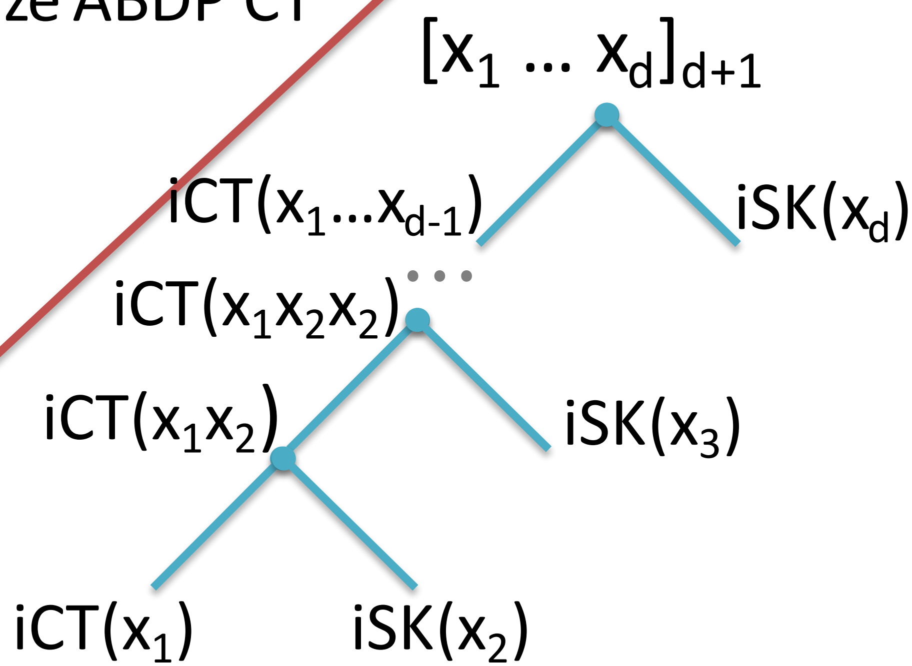
FE Construction

4. Security Challenge

3. Deg-d FE

Generalizing IPE to high degree, deg-d HIPE

Use HIPE to compress N^d-size ABDP CT

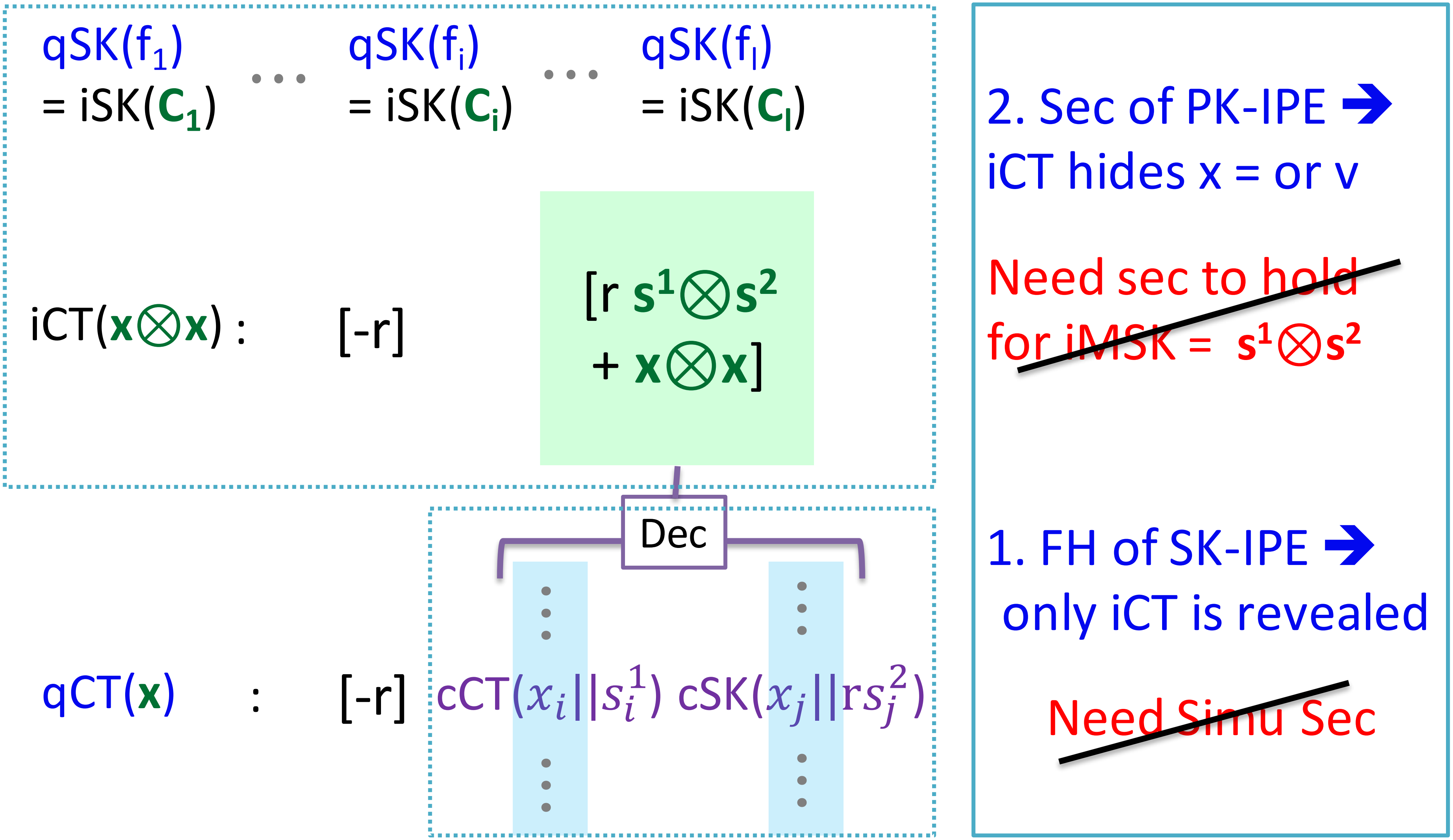


2. Quadratic FE

from Bilinear maps

1. IPE

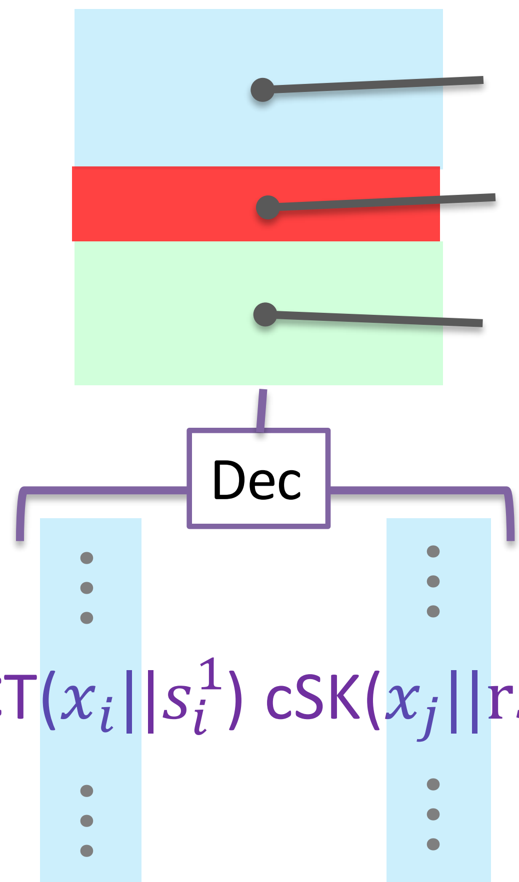
Want: qCT(u) ≈ qCT(v), if f_i(u) = f_i(v) for all i



Want: qCT(u) ≈ qCT(v), if $f_i(u) = f_i(v)$ for all i

$qSK(f_1) = iSK(\mathbf{C}_1)$ … $qSK(f_i) = iSK(\mathbf{C}_i)$ … $qSK(f_l) = iSK(\mathbf{C}_l)$

iCT($\mathbf{x} \otimes \mathbf{x}$) : [-r]



$[r\ \mathbf{s}^1_{<i}\mathbf{s}^2 + \mathbf{v}_{<i}\mathbf{v}]$

$[r\ s^1_i\mathbf{s}^2 + u_i\mathbf{u}\ /\ v_i\mathbf{v}]$

$[r\ \mathbf{s}^1_{>i}\mathbf{s}^2 + \mathbf{u}_{>i}\mathbf{u}]$

qCT($\mathbf{x}$) : [-r] cCT($x_i || s^1_i$) cSK($x_j || r s^2_j$)

1. Change matrix $x \otimes x$ row by row

2. Use Ind-FH to emulate Sim-FH

3. Add offset in qSK to keep outputs same

4. SXDH ➔ $s^1_i\mathbf{s}^2 \approx \mathbf{U}_n$

Lin16b:

IO ← SXDH on DEG-5 Multi-linear Maps
+ locality-5 PRG + LWE

LT17:

IO ← SXDH on Trilinear Maps
+ Block-locality-3 PRG + LWE

Bilinear Map

谢

Thank you!