Function Secret Sharing

Elette Boyle

IDC Herzliya (Reichman University)

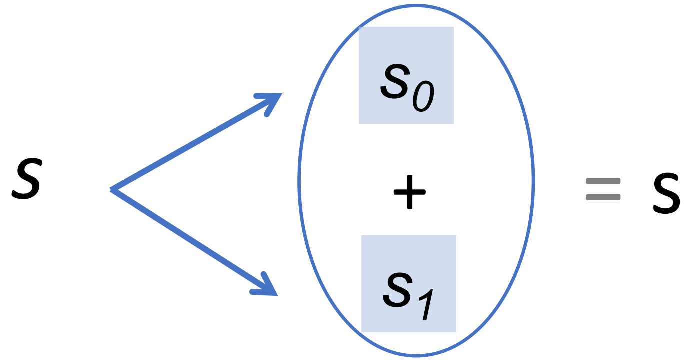
Based predominantly on joint works with Niv Gilboa and Yuval Ishai

In the Coming Days…

- **Function Secret Sharing**
- Prio +
- Oblivious RAM
- Vector OLE
- Pseudorandom Correlation Generators
- Private Set Intersection
- Signatures

Additive Secret Sharing

Elements in Abelian group $\mathbb{G}$



- **Secrecy**: s_b hides s
- **Reconstruction**: $s_0 + s_1 = s$ (in $\mathbb{G}$)

Function Secret Sharing (FSS) [BGI15]

Secret *function*

$f \; : \; \{0,1\}^n \rightarrow \mathbb{G}$

i.e., 2^n *correlated* secrets in $\mathbb{G}$

f_0

$+$

f_1

$= f$

Secrets have a
compact representation (via f)…

Can we secret share them ALL
in a compact way?

FSS: The 3-Hour Adventure

Definition & Discussion & Highlights

Core Constructions

Extensions & Applications

FSS: Definition & Discussion

Function Secret Sharing (FSS) for $\mathcal{F}$

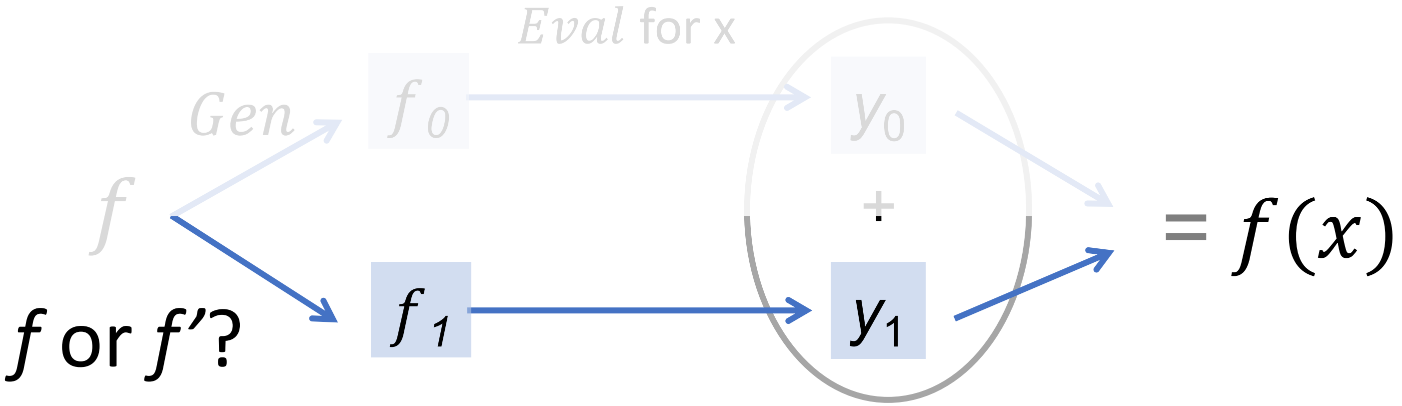
For this lecture:
Focus on 2 shares

Definition [BGI15]: **FSS** scheme for class $\mathcal{F}$ is $(\mathrm{Gen}, \mathrm{Eval})$ st:

- $\mathrm{Gen}(1^\lambda, f)$ for $f \in \mathcal{F}$ → (f_0, f_1) sometimes k_0, k_1 "function keys"
- $\mathrm{Eval}(b, k_b, x)$ for $x \in \mathrm{Domain}(f)$ → y_b output share

satisfying…

- Secrecy: "Semantic security": $\forall f, f' \in \mathcal{F}$, $\{ k_b \text{ from } f \} \approx \{ k_b \text{ from } f' \}$
- Reconstruction: $y_0 + y_1 = f(x)$



Alternative Notion of Security

- “**Semantic security**”:

  $\forall f, f' \in \mathcal{F}$,

  $$\{ k_b \text{ from } f \} \approx \{ k_b \text{ from } f' \}$$

- “**Simulation security**” wrt leakage function L:

  $\exists$ Sim st $\forall f \in \mathcal{F}$,

  $$\{ k_b \text{ from } f \} \approx \{ \text{Sim}(\text{L}(f)) \}$$

Allows fine-grained hiding/revealing

(Semantic security) ≡ (Simulation security wrt L= $\mathcal{F}$)

Remarks

- This talk: Split into **2** shares

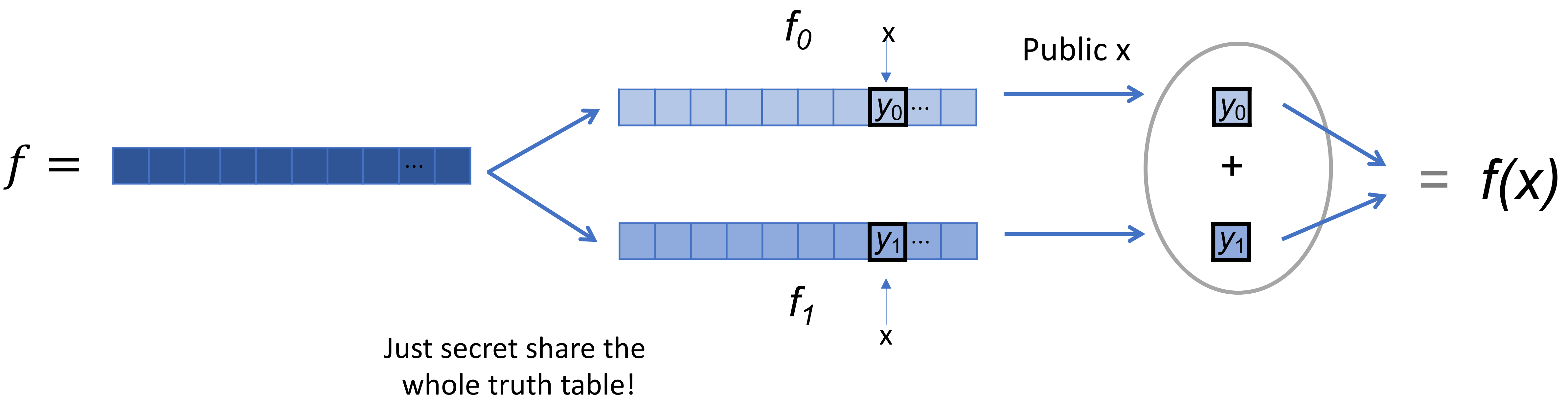
$$\begin{tikzcd} & f_0 \\ f \arrow[ur] \arrow[dr] & \\ & f_1 \end{tikzcd}$$

- This talk: **Semi-honest** parties (wait for Henry's talk!)

- This talk: **Additive** reconstruction

  Why additive? Hold that thought…

Example: FSS for All Functions (Truth Table)



- Share size = |Truth Table| ~ $O(2^n)$

Example: Linear Functions [Ben86]

$f(x) = ax + b$

Over ring R

$a_0 \quad b_0$

$a_1 \quad b_1$

Additively share

a, b over R

Public x

$a_0 x + b_0$

$a_1 x + b_1$

+

$ax + b$

$= f(x)$

Example: Polynomials

$f(x) =$

$ax^2 + bx + c$

Over ring R

$\boxed{a_0}\ \boxed{b_0}\ \boxed{c_0}$

$\boxed{a_1}\ \boxed{b_1}\ \boxed{c_1}$

Public x

$\boxed{a_0}x^2 + \boxed{b_0}\,x + \boxed{c_0}$

$+$

$\boxed{a_1}x^2 + \boxed{b_1}\,x + \boxed{c_1}$

$= ax^2 + bx + c$

$= f(x)$

More generally: **Secret** linear combination of **public** functions of x

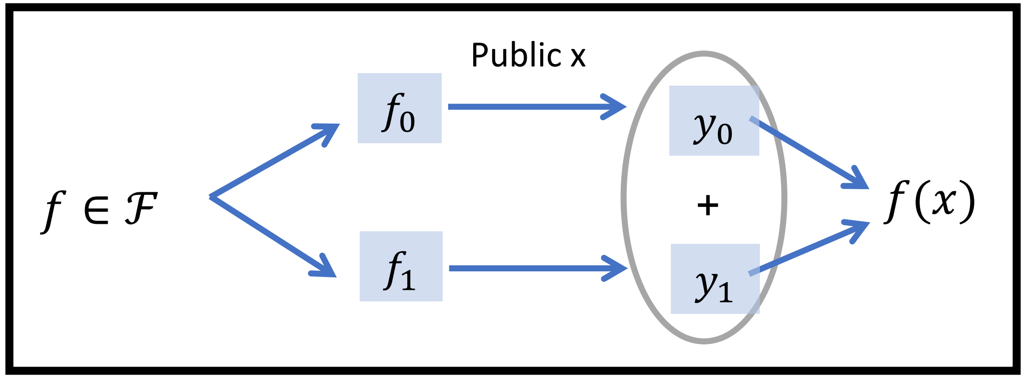
$a \ , \ b \ , \ c$ $\qquad$ $x^2 , \ x , \ 1$

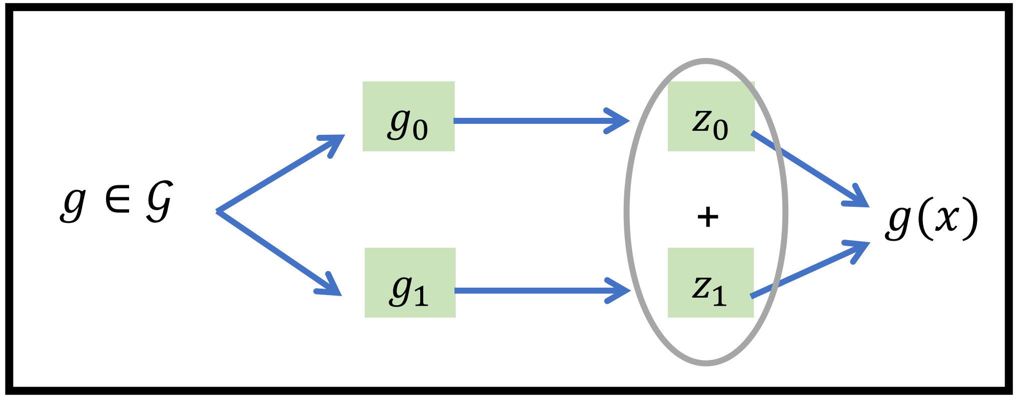
Note: Sum of FSS

FSS for $\mathcal{F}$

$+$

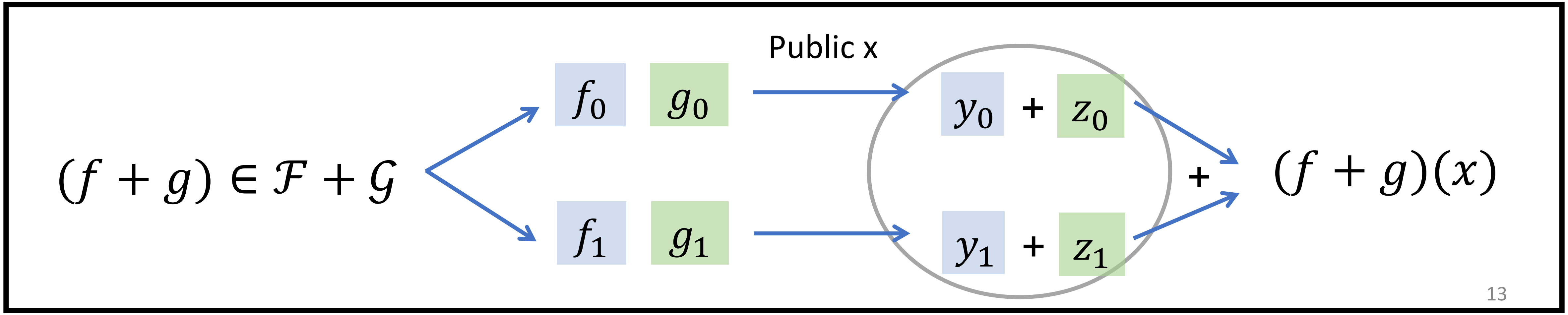
FSS for $\mathcal{G}$





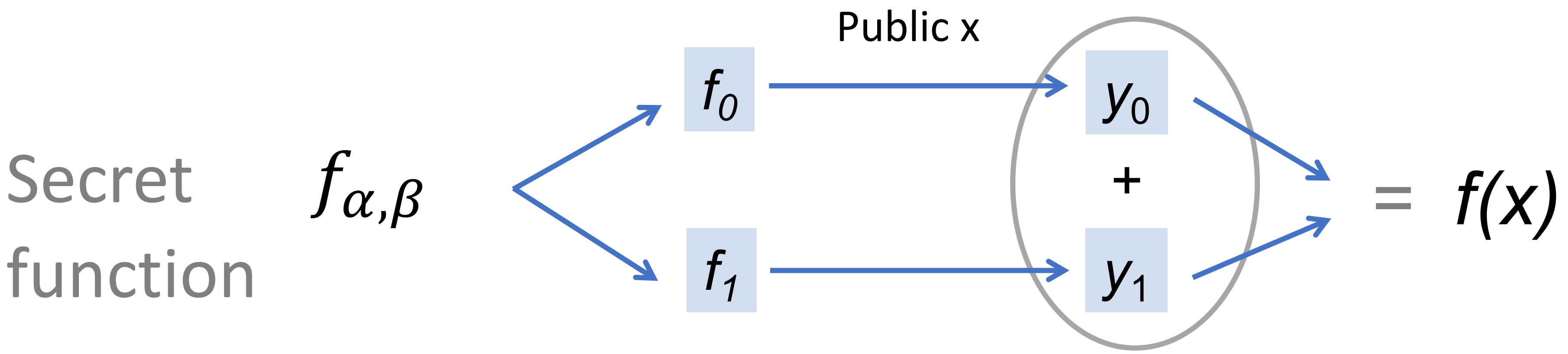
$\Rightarrow$ FSS for

$\mathcal{F} + \mathcal{G} = \{f + g : f \in \mathcal{F}, g \in \mathcal{G}\}$



A Little Different…

Useful Example: FSS for **Point Functions** [GI14]

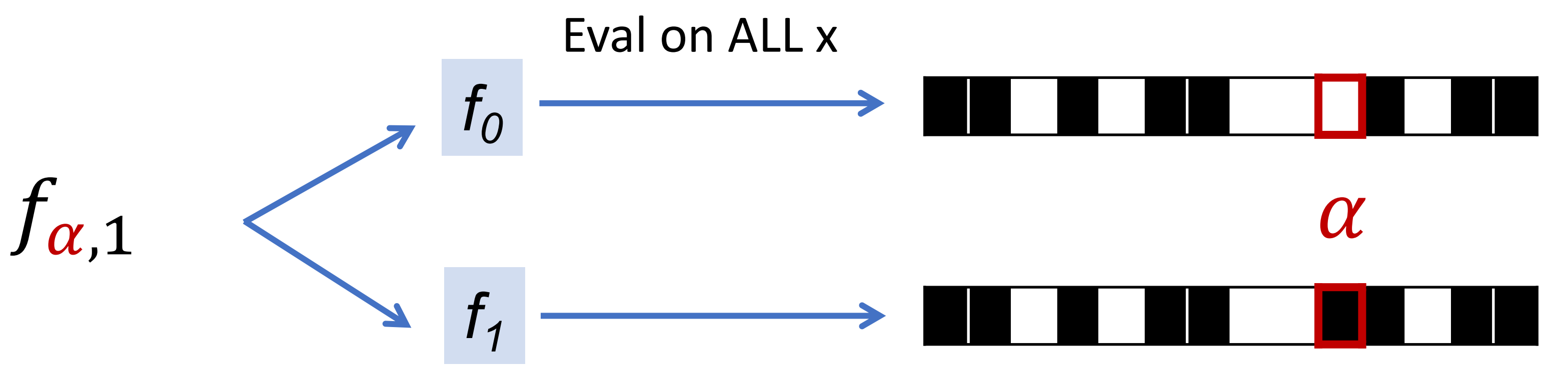


- Point function class $\mathcal{F}$: $\quad f_{\alpha,\beta} : \{0,1\}^n \to \mathbb{G}$

Eg: $\mathbb{G} = \mathbb{Z}_2$ or $\mathbb{Z}_N$

$$f_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0 & else \end{cases}$$

Useful Example: FSS for **Point Functions** [GI14]



= "Distributed Point Functions (DPF)"

Construction: In Part 2!

Coming Up Next…

- **Sample Application**: Private Data Manipulation

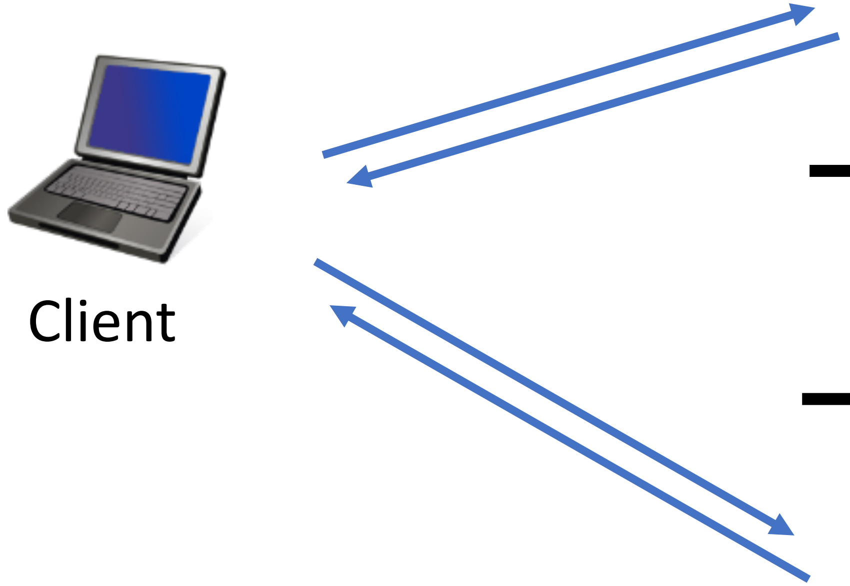
- **Overview:** What is Known

Sample Application

Private Data Manipulation

Goal: Private Queries to Public DB

Query DB, hiding query from servers



#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

⋮

#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

Public DB

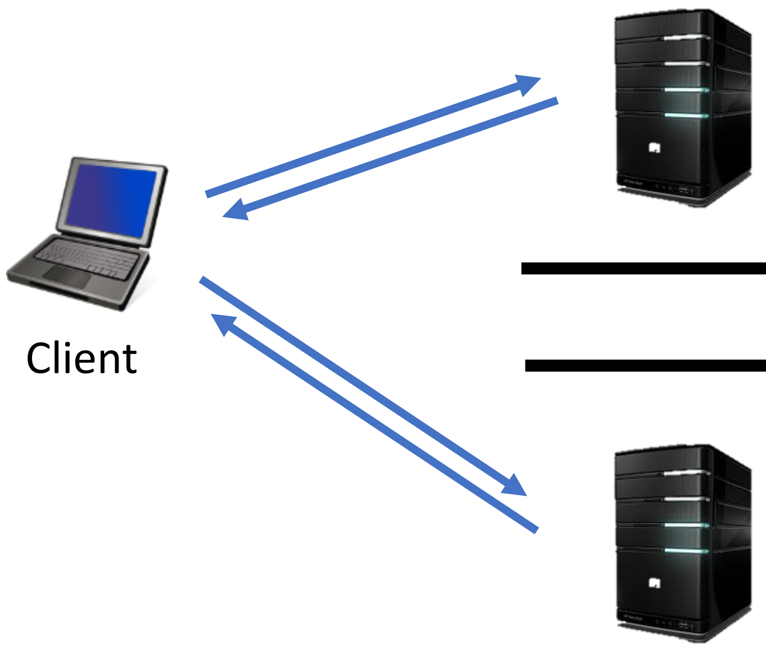
Held by $s \geq 1$ (non-colluding) servers

Examples: Stock quotes, map search, …

Note: Client does not own DB (many-client setting)

Special Case: Private Information Retrieval (PIR)

[CGKS98, KO00]



#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

⋮

#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

Public DB

Held by $s \geq 1$ (non-colluding) servers

Private Query: "Retrieve item i" (while hiding index i)

Private Information Retrieval

Suppose DB = n entries, each 1 bit

<u>Statistical Privacy</u>

- **2+ servers**:

  slightly $n^{o(1)}$

  [Yekhanin07,Efremenko09, Dvir-Gopi 15]

- **1 server**: Impossible.

  Requires public-key crypto

  [Di Crescenzo-Malkin-Ostrovsky 00]

<u>Computational Privacy</u> (λ = sec param)

- **2+ servers**: (one-way functions)

  $(\lambda + 2) \log n$ [Boyle-Gilboa-Ishai 16b]

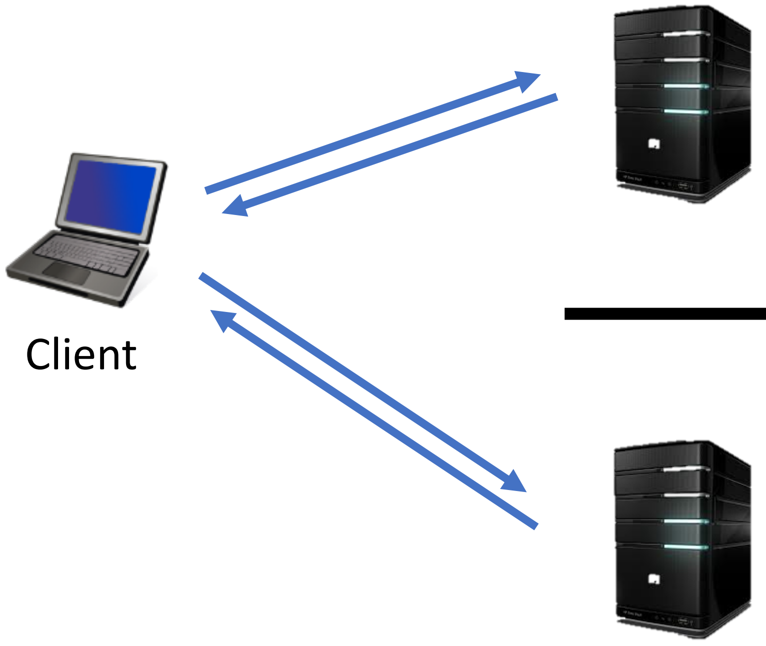
- **1 server**: (structured PKE assumptions)

  $\mathrm{poly}(\lambda) \log^2 n$ [Kushilevitz-Ostrovsky00,…]

Thus: A Motivated Setting

- 2 non-colluding servers
- Lightweight crypto

Non-collusion: Eg, different providers / subpoena jurisdictions…



#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

Public DB

Held by $s \geq 2$ (non-colluding) servers

#	Value 1	Value 2
1	100101	Jane
2	100100	Daniel
3	011011	Peter
4	101010	Elliott
5	001001	Maya
6	110101	Samuel

For this talk: passive adversary (honest-but-curious server)

FSS for Point Functions ⇒ 2-Server PIR

To access item $i \in [n]$:

Define point function $f_{i,1} : [n] \to \mathbb{Z}_2$

$$f_{i,1}(j) := \begin{cases} 1 \text{ if } j = i \\ 0 \text{ else} \end{cases}$$

$$f_{i,1} \to f_A, \quad f_{i,1} \to f_B$$

$$y_A + y_B = 1 \cdot val[i] + \sum_{\{j \neq i\}} 0$$

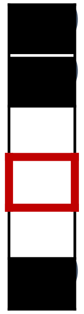
$f_A \rightarrow$

$y_A \leftarrow$

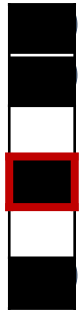
$f_B \rightarrow$

$y_B \leftarrow$

$y_A = \Sigma f_A(j)\, val[j]$



#	Value
1	10000101
2	10010100
3	01100111
4	10101010
5	00101101
6	11010101



#	Value
1	10000101
2	10010100
3	01100111
4	10101010
5	00101101
6	11010101

Communication $= |f_A| + |\text{val}[i]|$ | $\sim \mathbf{log}\, \boldsymbol{n} + |\text{val}[i]|$ for FSS from PRGs!

Also…

Private Updates to Secret-Shared Data

FSS for Point Functions ⇒ Private Histograms

To increment item i :

Define point function $f_{i,1} : [n] \to \mathbb{Z}_2$

$f_{i,1}$ → f_A, f_B

Anyone can increment!

$\xrightarrow{f_A}$

	#	Value
$f_A(1)$ +	1	10000101
$f_A(2)$ +	2	10010100
	3	01100111
⋮	4	10101010
	5	00101101
$f_A(6)$ +	6	11010101

$\xrightarrow{f_B}$

	#	Value
$f_B(1)$ +	1	10000101
$f_B(2)$ +	2	10010100
	3	01100111
⋮	4	10101010
	5	00101101
$f_B(6)$ +	6	11010101

Servers store **Secret-shared** DB

FSS for Point Functions ⇒ Private Histograms

To increment items satisfying P :

Define secret function

$$f^P(x) = \begin{cases} 1 & if\ P(x) = 1 \\ 0 & else \end{cases}$$

$$f^P \to f_A,\ f_B$$

$\xrightarrow{f_A}$

	#	Value
$f_A(1)$ +	1	10000101
$f_A(2)$ +	2	10010100
	3	01100111
⋮	4	10101010
	5	00101101
$f_A(6)$ +	6	11010101

$\xrightarrow{f_B}$

	#	Value
$f_B(1)$ +	1	10000101
$f_B(2)$ +	2	10010100
	3	01100111
⋮	4	10101010
	5	00101101
$f_B(6)$ +	6	11010101

Servers store **Secret-shared** DB

Leakage: Query class supported by FSS scheme (& columns applied to)

Remarks

- Why **additive** reconstruction?

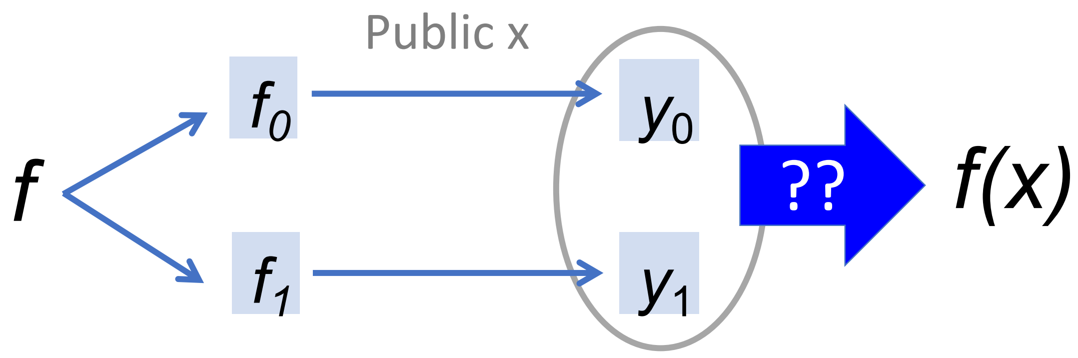
  Linear compressibility enables to compress server's reply

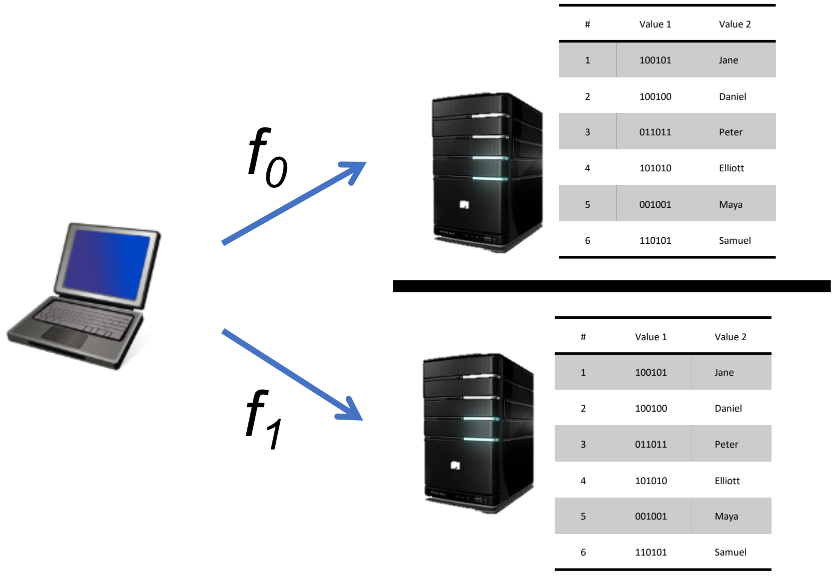
- **Linear** in DB: Are you crazy??

  Sometimes this is not so bad…

- What's the **leakage**?

  Reveals FSS query **class** $\mathcal{F}$
  Hides **query** from within class





FSS for $\mathcal{F} \Rightarrow$ Private Database Manipulation

FSS for more general function classes

$\Longrightarrow$ more expressive database manipulation

- **Private Updates to Secret-Shared Data**

  Voting, Secret histograms, Anonymous broadcast, …

- **"Attribute-Based" Information Retrieval**

  Multi-keyword search, Range queries, DB statistics, …

  - **Counting** Queries: f outputs $\{0,1\}$ in $\mathbb{Z}_N$
  - **Recovery** Queries: for m items, using sketching techniques (eg, [OS07])

Application: 2-Server **Private Database Queries**

Counting Query Example:

- Salary between \$100-200k,
- AND Birthday in October,
- AND Female

FSS for more expressive class $\mathcal{F}$

$$f\colon \mathbb{Z}_M \times \mathbb{Z}_{12} \times \mathbb{Z}_2 \to \mathbb{Z}_N$$

$$f(x,y,z) := \begin{cases} 1 & \text{if } x \in \$[100k, 200k] \\ & \wedge\, y = \text{Oct} \\ & \wedge\, z = \text{Female} \\ 0 & \end{cases}$$

Count = $y_A + y_B \in \mathbb{Z}_N$

f_A

y_A

f_B

y_B

$y_A = \Sigma\, f_A(x_j, y_j, z_j)$

Name	Salary	DOB	G
Alexandra Baker	\$289,000	3/14/80	F
Patricia Callman	\$215,000	7/11/76	F
Preston Greenly	\$98,000	1/11/81	M
Graeme Roberts	\$223,000	9/28/77	M
Martin Wolferson	\$109,000	10/9/79	M
Charles Zanzabar	\$72,000	6/24/86	M

Name	Salary	DOB	G
Alexandra Baker	\$289,000	3/14/80	F
Patricia Callman	\$215,000	7/11/76	F
Preston Greenly	\$98,000	1/11/81	M
Graeme Roberts	\$223,000	9/28/77	M
Martin Wolferson	\$109,000	10/9/79	M
Charles Zanzabar	\$72,000	6/24/86	M

Overview of FSS: What is Known

Side Note: Function vs Homomorphic SS (HSS)

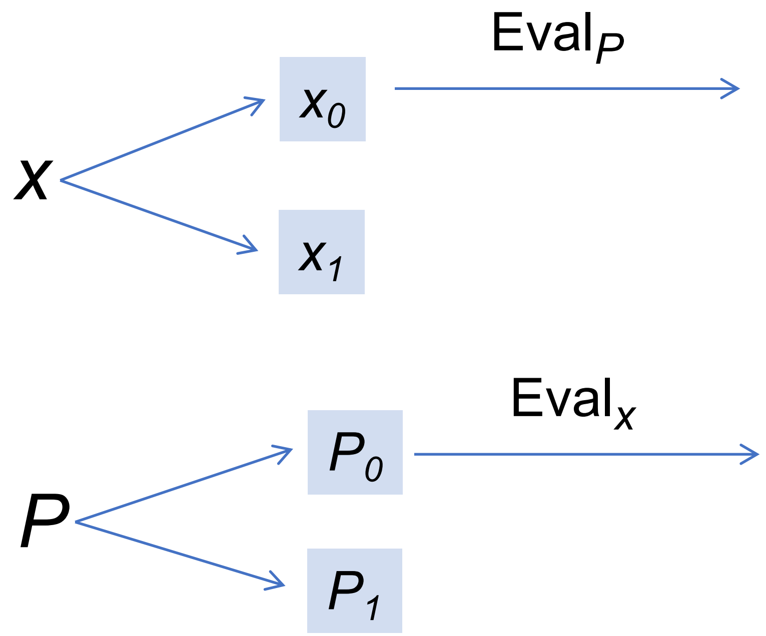
For $P \in \mathcal{P}$ and input x

- **H**SS for program class $\mathcal{P}$

  share size ~ $|x|$

- **F**SS for program class $\mathcal{P}$

  share size ~ $|P|$



FSS/HSS more natural in different applications

Function Secret Sharing: Current Landscape

Not efficient (Builds atop ***specific FHE***)

"High-level"		
LWE+	Circuits	[DHRW16, BGI15, BGILT18]
"Mid-level"		
DDH	Branching Programs	[BGI16, BCGIO17, DKK18]
Paillier		[FGJS17, OSY21, RS21]
LWE		[BKS19]
"Lapland"	Low-deg polynomials	[BCGIKS19]
LPN	Weird PRGs…	
"Low-level"		
OWF	Simple functions	[GI14, BGI15, BGI16b]
"Algorithmica"		
None	Linear Combinations w/ Secret Coeffs	[Ben86]

Structured assumptions yielding ***PKE***

"Weird PRGs": Wait for Peter/Yuval…

Requires one-way functions [GI14,BGI15]

In Particular… Lightweight Constructions

From any Pseudorandom Generator (PRG)

- Point Functions [GI14,BGI15,BGI16b] ⇒ PIR, keyword search
- Interval Functions [BGI15,BGI16b] ⇒ Range queries

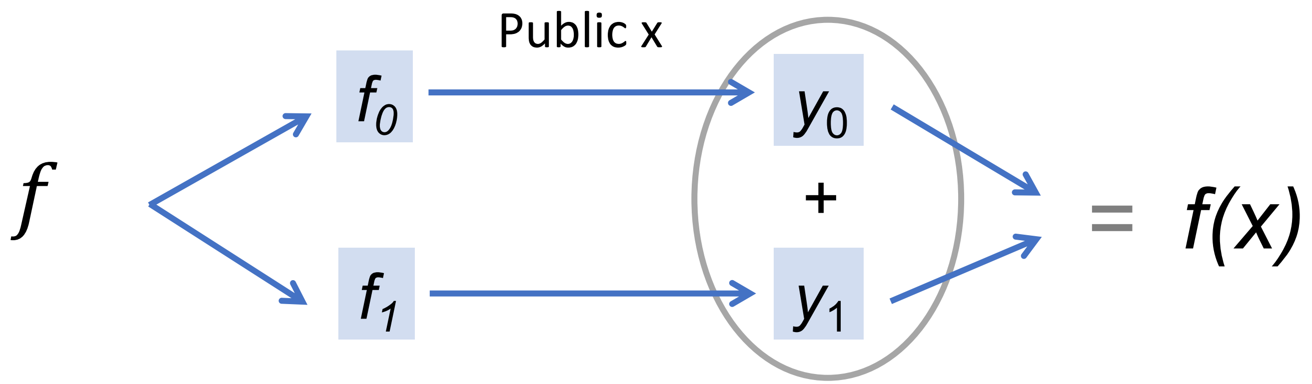
- (Small) Constant-Dimension Intervals ⇒ Small conjunctions
- Simple Decision Trees [BGI16b]

Stretch Break!

Cliffhanger… how can we **build** this great **FSS** thing?

Recap:

FSS:



- Useful applications in private data manipulation (& more!)
- "Distributed Point Function" (DPF) = FSS for point functions

Part II: Constructions of FSS

FSS for Point Functions

FSS for Comparison Functions

Construction:

FSS for **Point Functions**

= Distributed Point Functions (DPF)

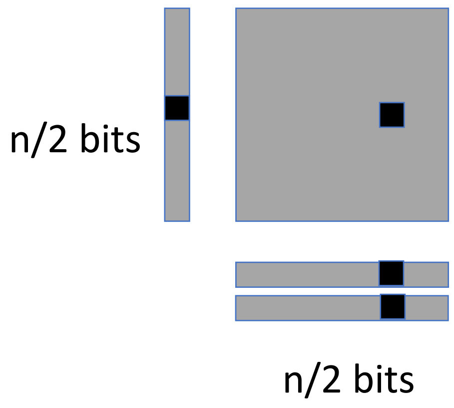
$$f_\alpha(x) := \begin{cases} 1 \text{ if } x = \alpha \\ 0 \text{ else} \end{cases}$$

key size in bits: n = input bit len
λ = sec param

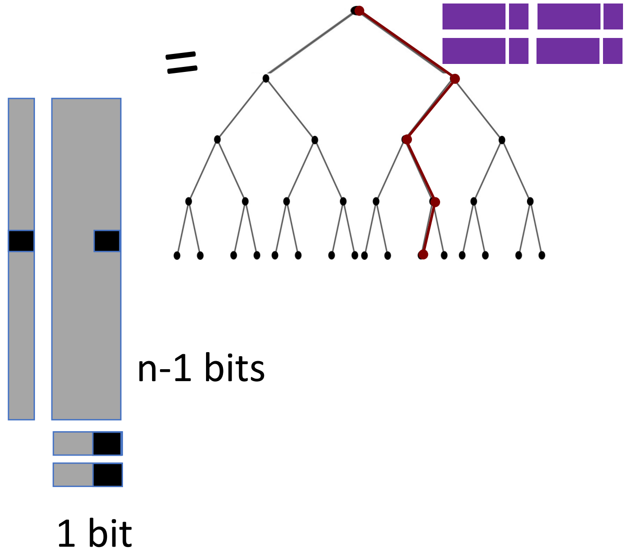
History of DPF from OWF

Implicit in [CG99]
$O(2^{\sqrt{n}}\lambda)$

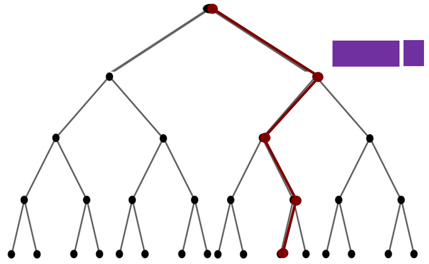
[GI14]
Recursive
$\sim O(n^{1.58}\lambda)$



[BGI15]
Tree-Based
$\sim 4\,n\lambda$



[BGI16b]
Optimized Tree
$\sim n\lambda$

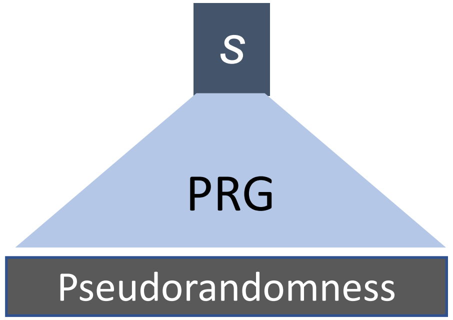


???

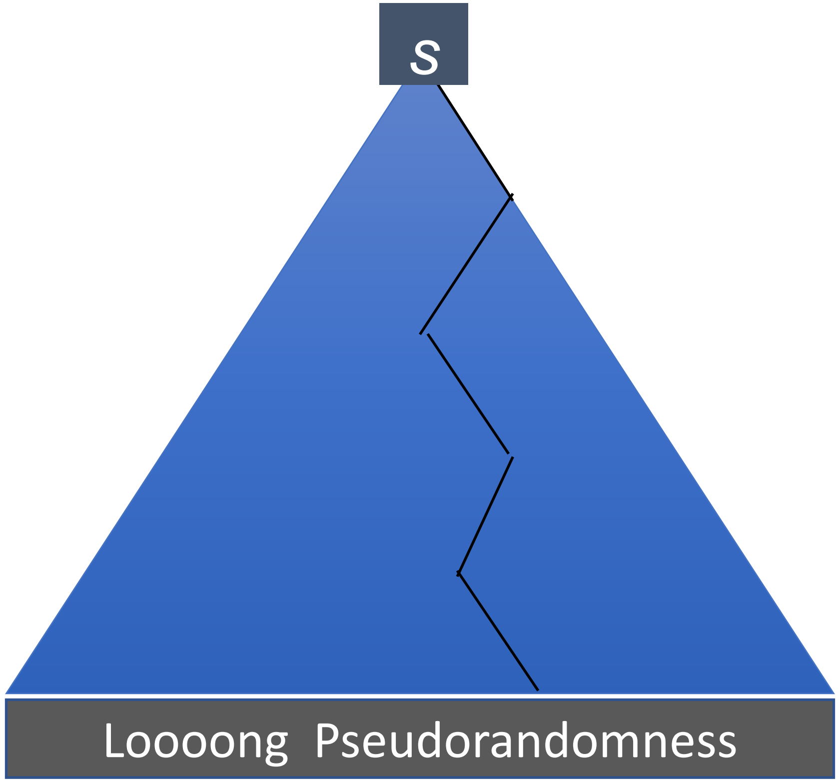
DPF Construction: Starting Tools

- Uses (any) length-doubling Pseudo-Random Generator (PRG)
- Useful Tool: GGM Pseudorandom function (PRF)
  [Goldreich-Goldwasser-Micali 84]

Length-doubling PRG



(Eg: 2 calls to AES)

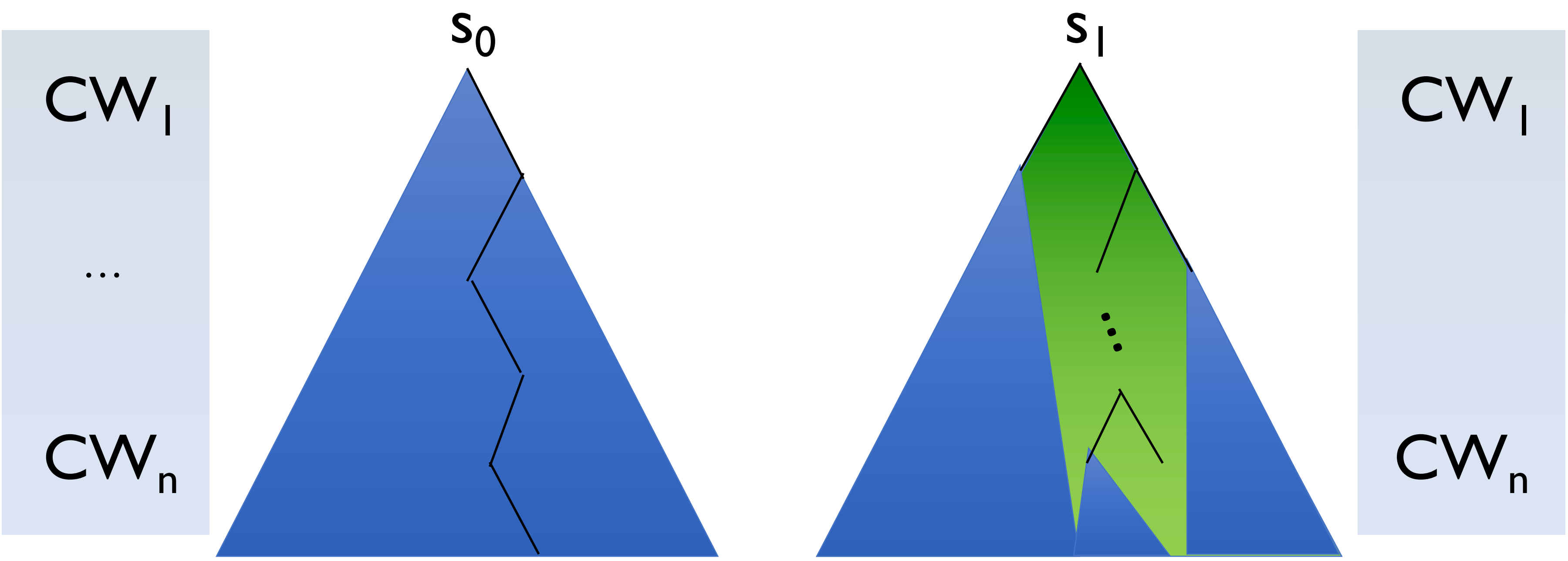


DPF Construction Overview

[Boyle-Gilboa-Ishai 16b]

Suppose domain $[N] = [2^n]$

Random PRG seeds



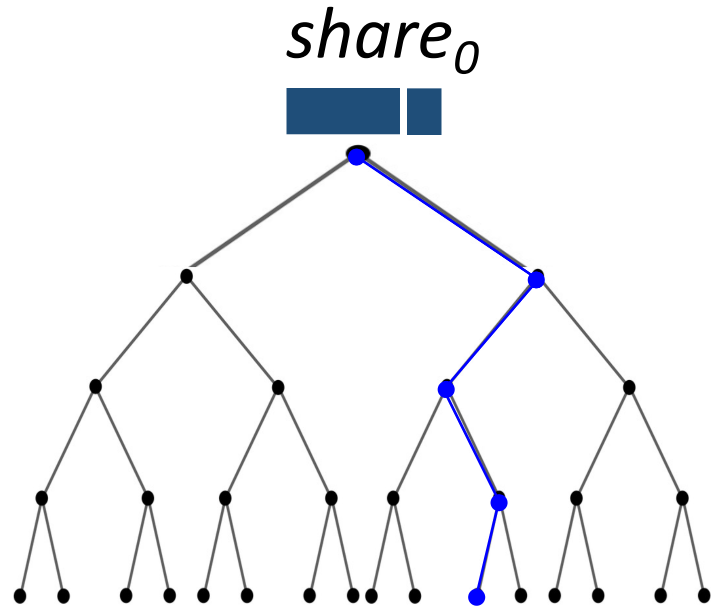
"Correction Words" at each level

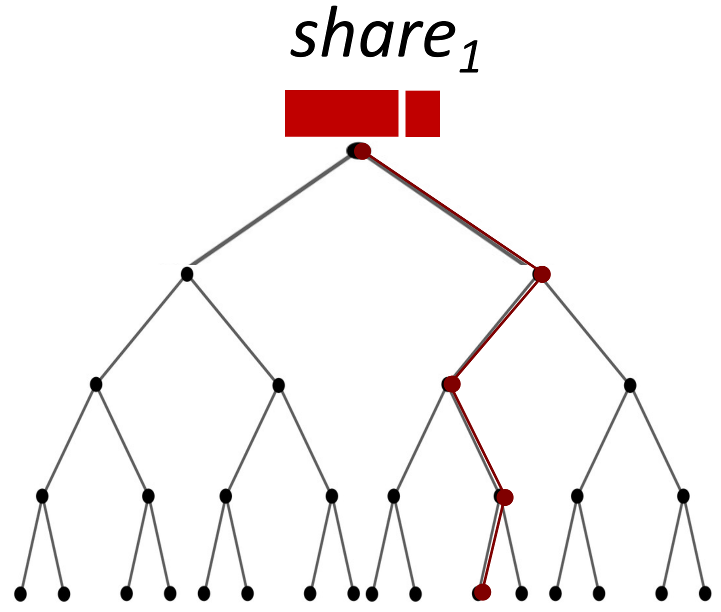
(to force equality once input disagrees with special value)

DPF Construction from PRGs

[BGI16b]

$f_\alpha: \{0,1\}^n \to \{0,1\}$





Invariant for Eval:

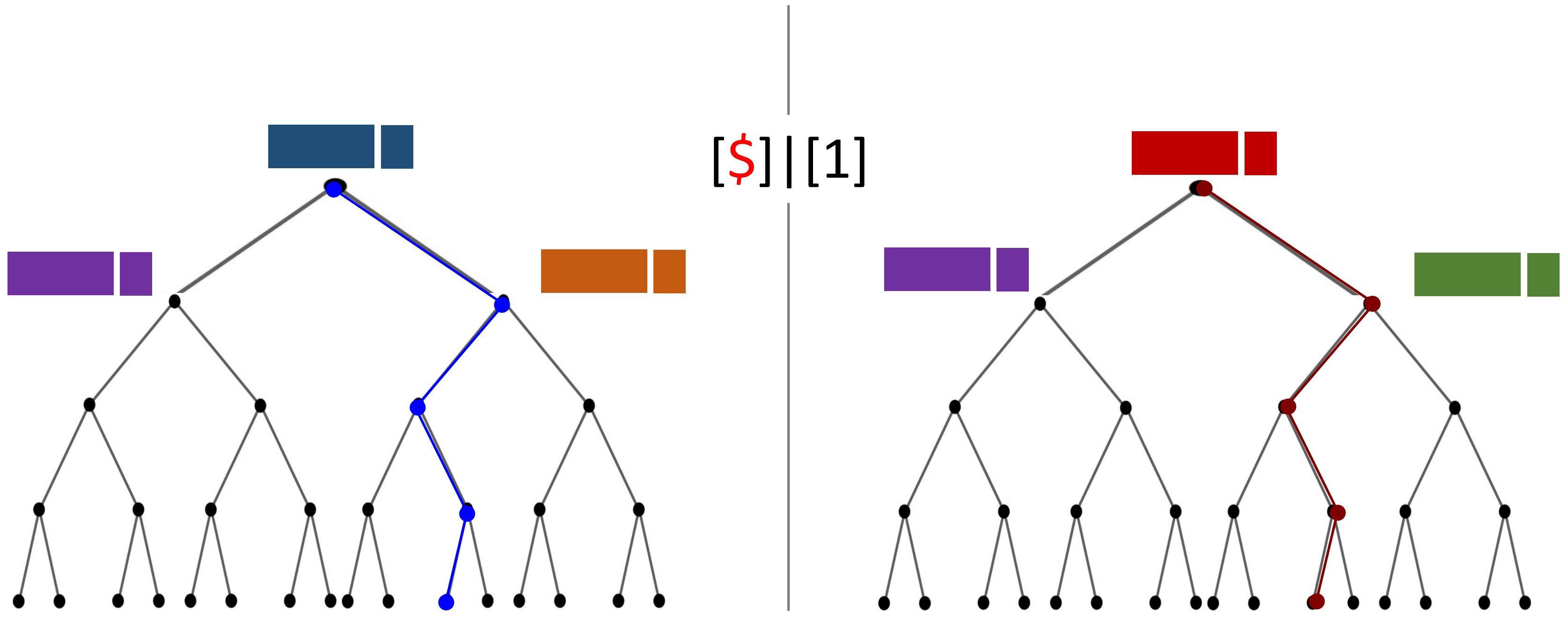
λ-bit

1-bit

For each node v on evaluation path we have [S] | | [b]

Additive secret shares

DPF Construction from PRGs

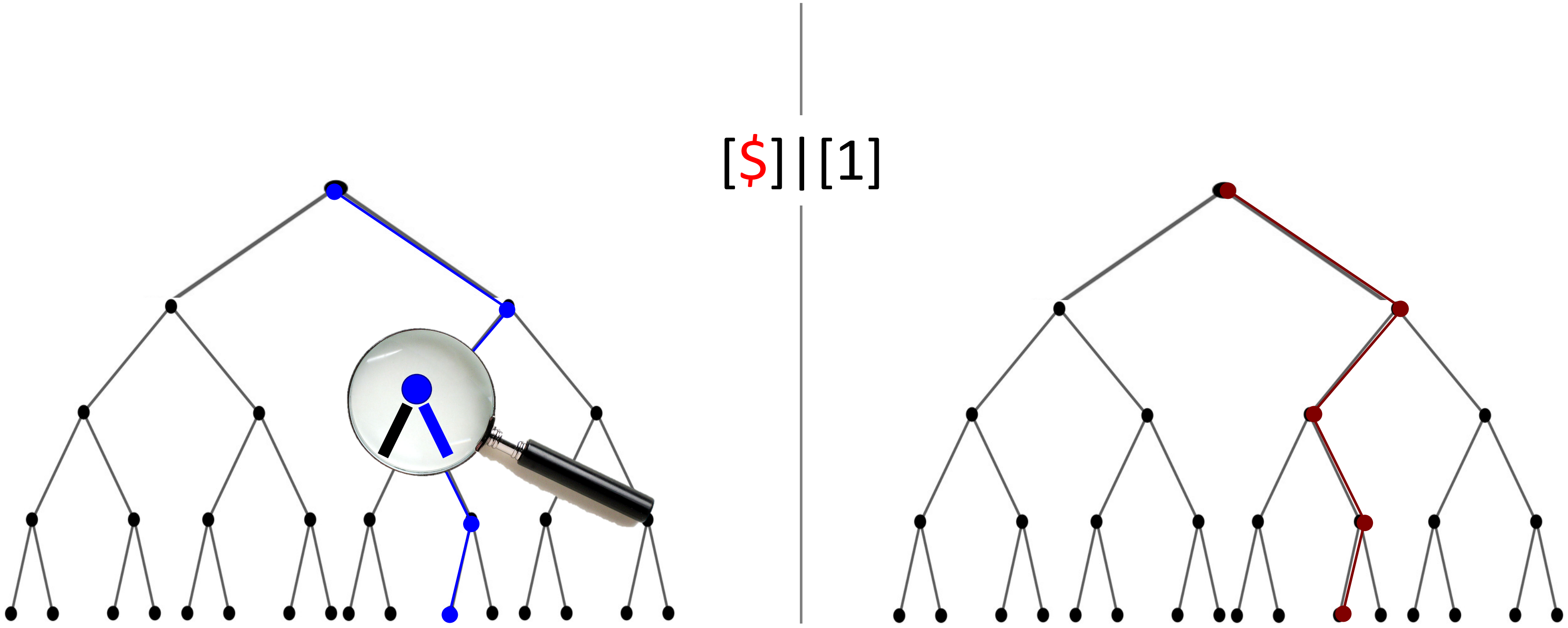


Invariant for Eval:

For each node v on evaluation path we have [S]||[b]

- v on special path: S is pseudorandom, b=1
- v off special path: S=0, b=0

DPF Construction from PRGs

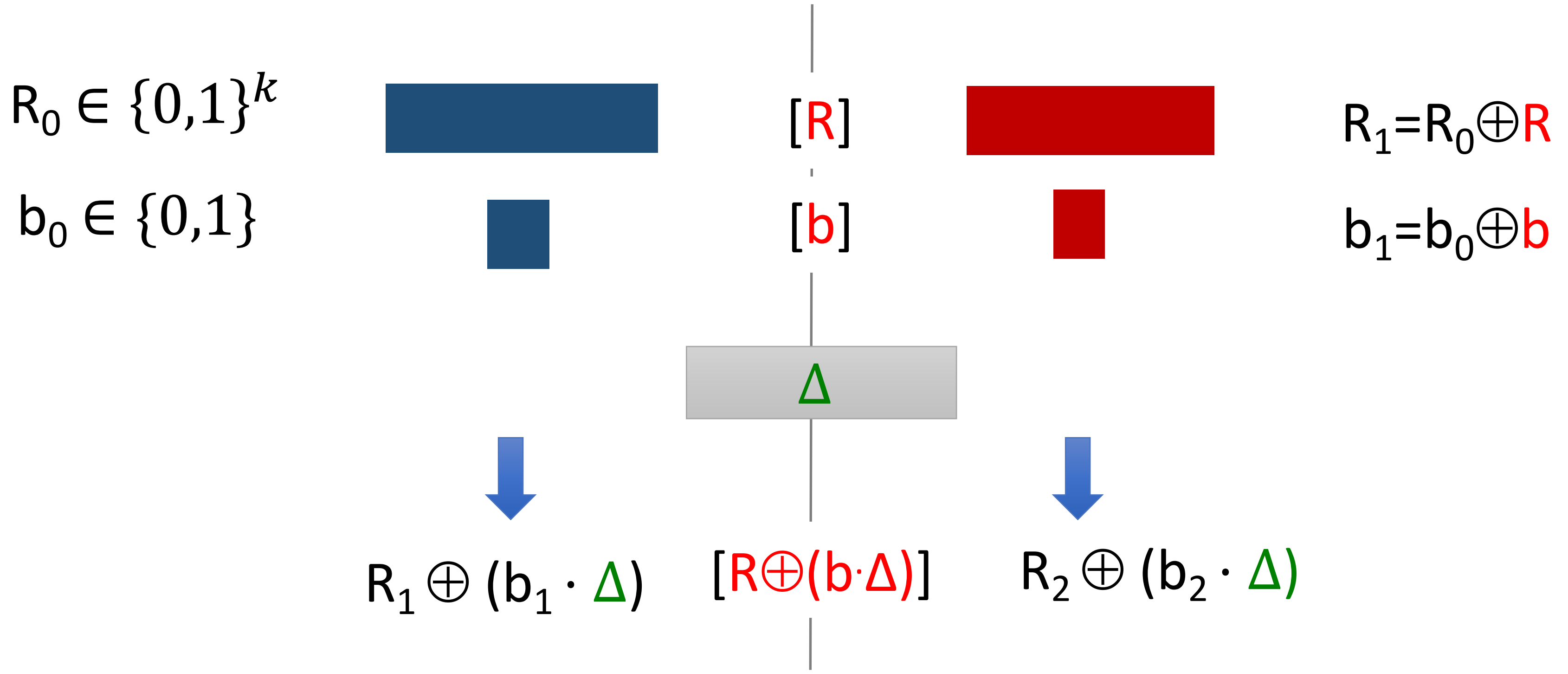


Invariant for Eval:

For each node v on evaluation path we have [S]||[b]

- v on special path: S is pseudorandom, b=1
- v off special path: S=0, b=0

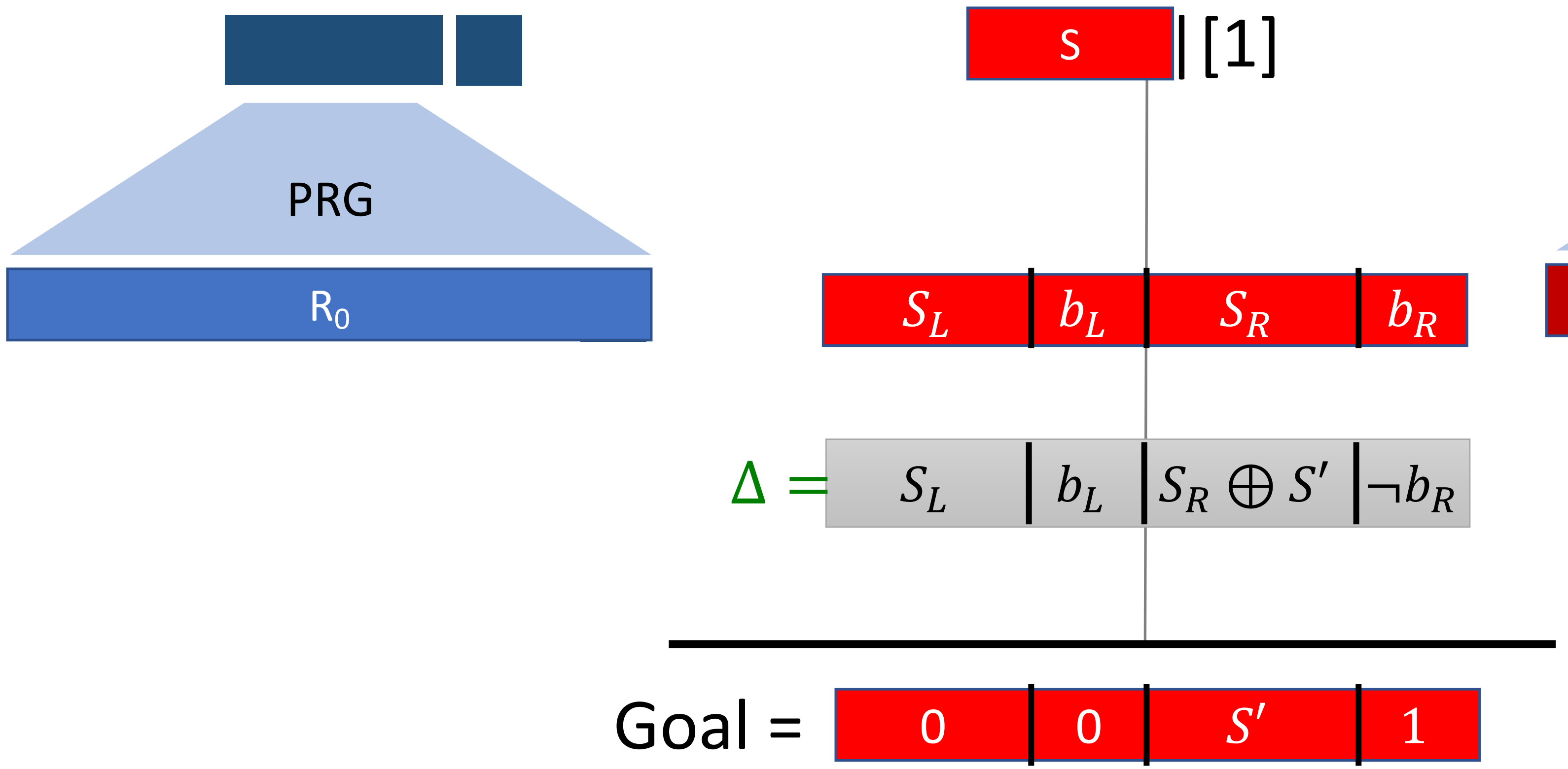
Gadget: Conditional Correction



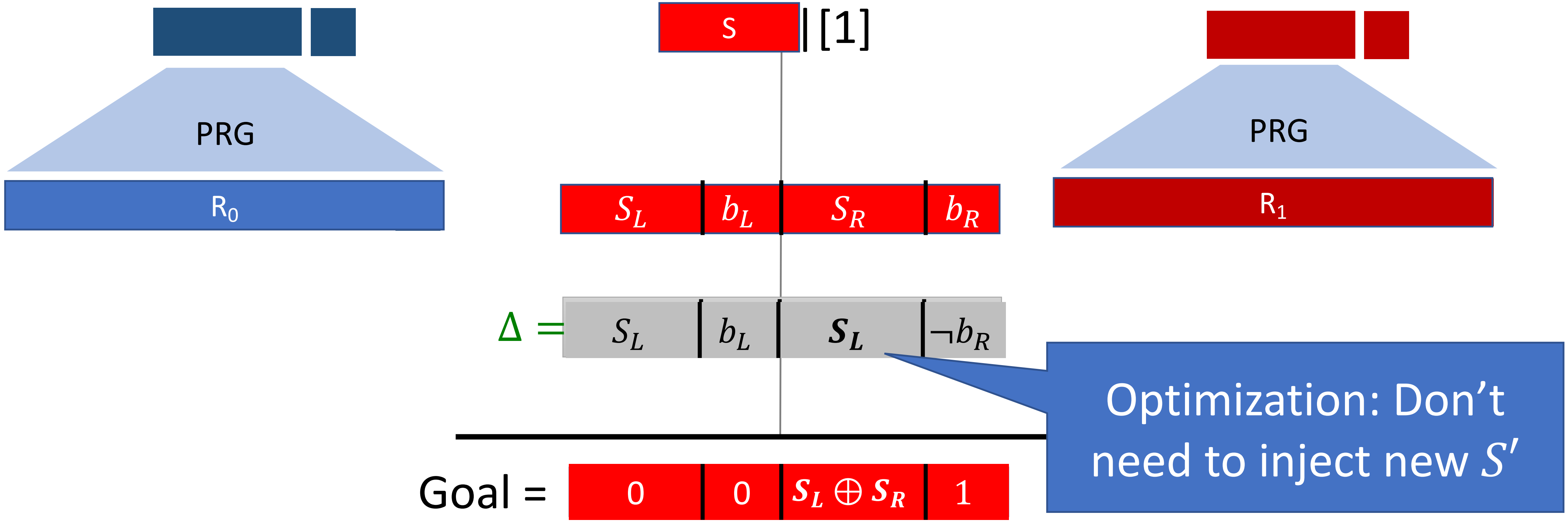
Test yourself:

- R=0, b=0 $\Rightarrow$ generate shares of… 0!
- Δ=R, b=1 $\Rightarrow$ generate shares of… 0!

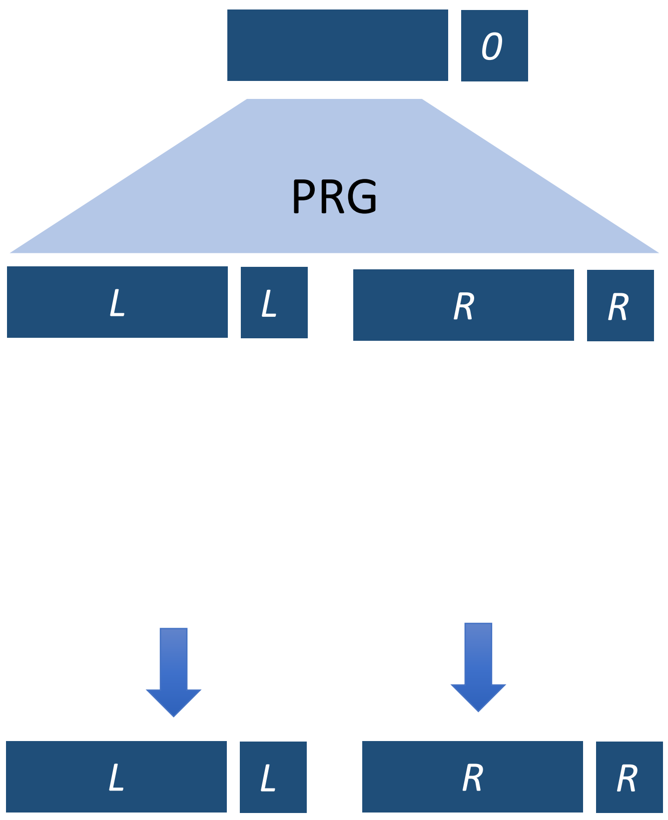
Building the Correction Word Δ

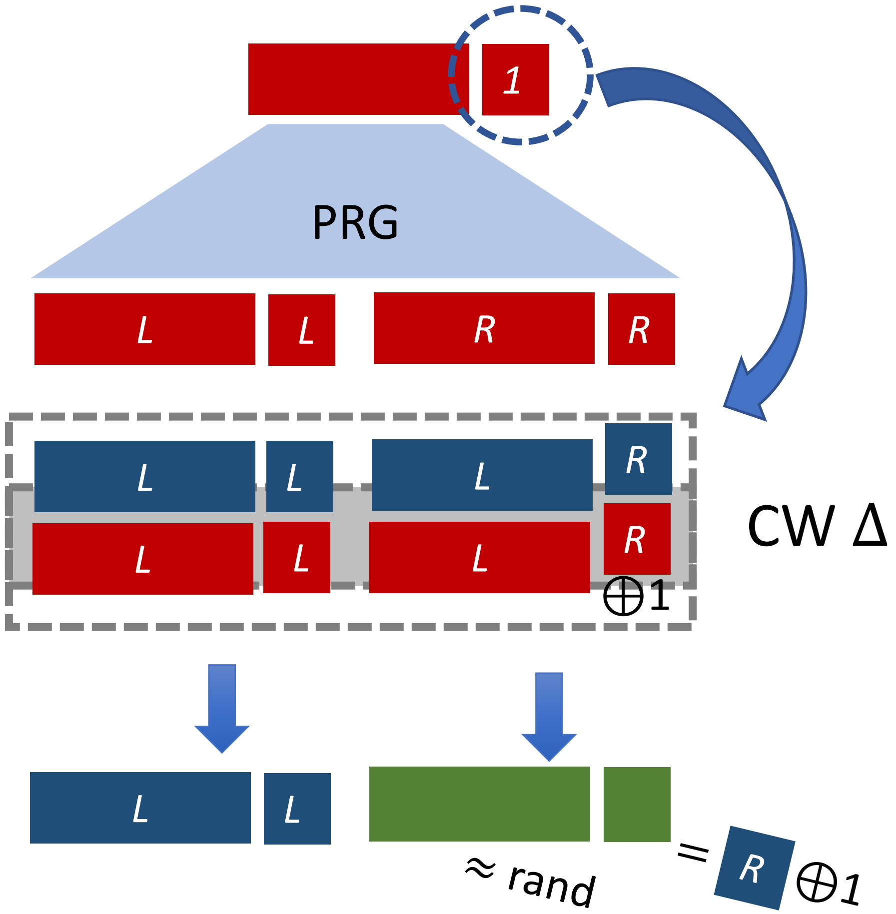


Building the Correction Word Δ

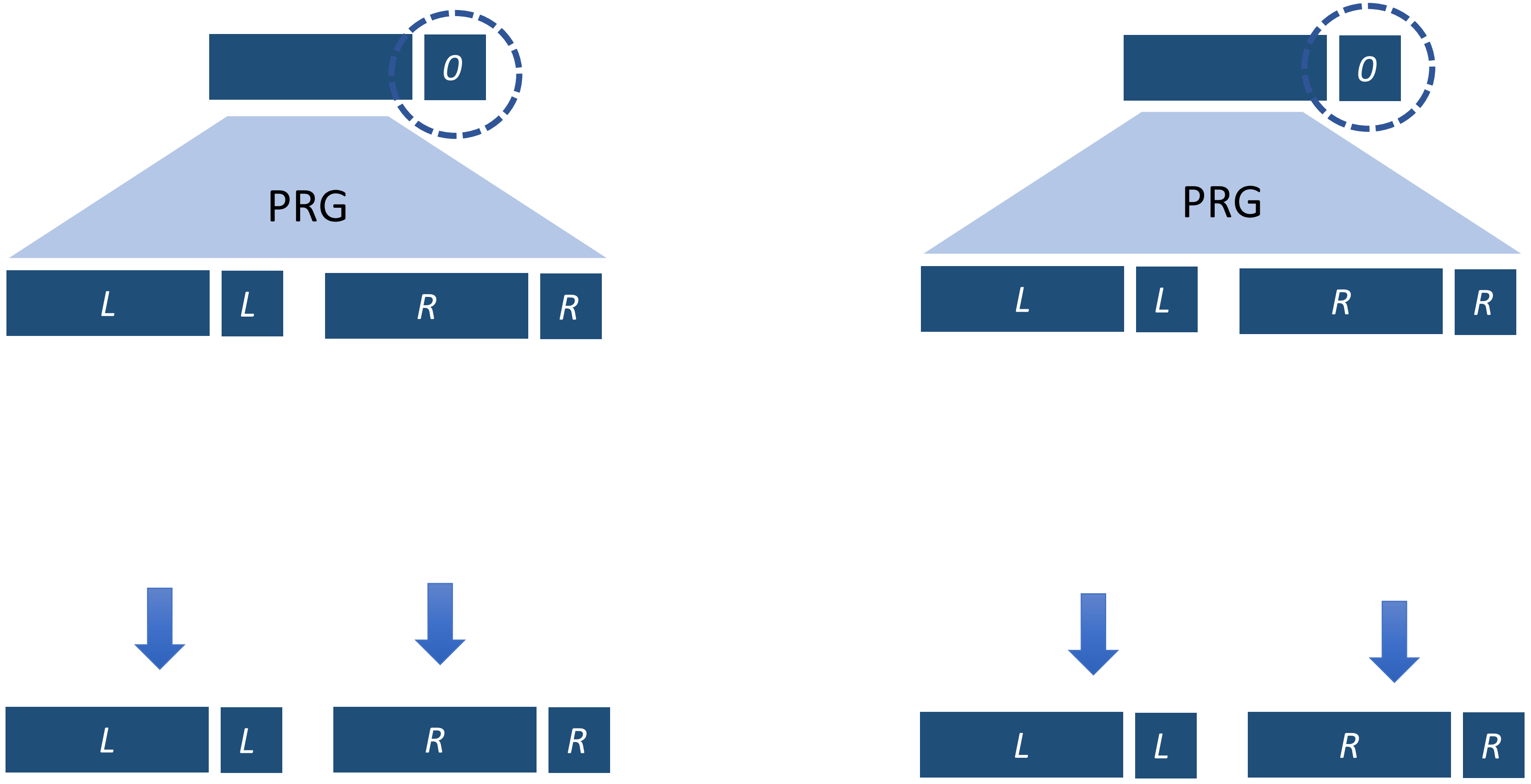


Using the CW Δ : On-Path

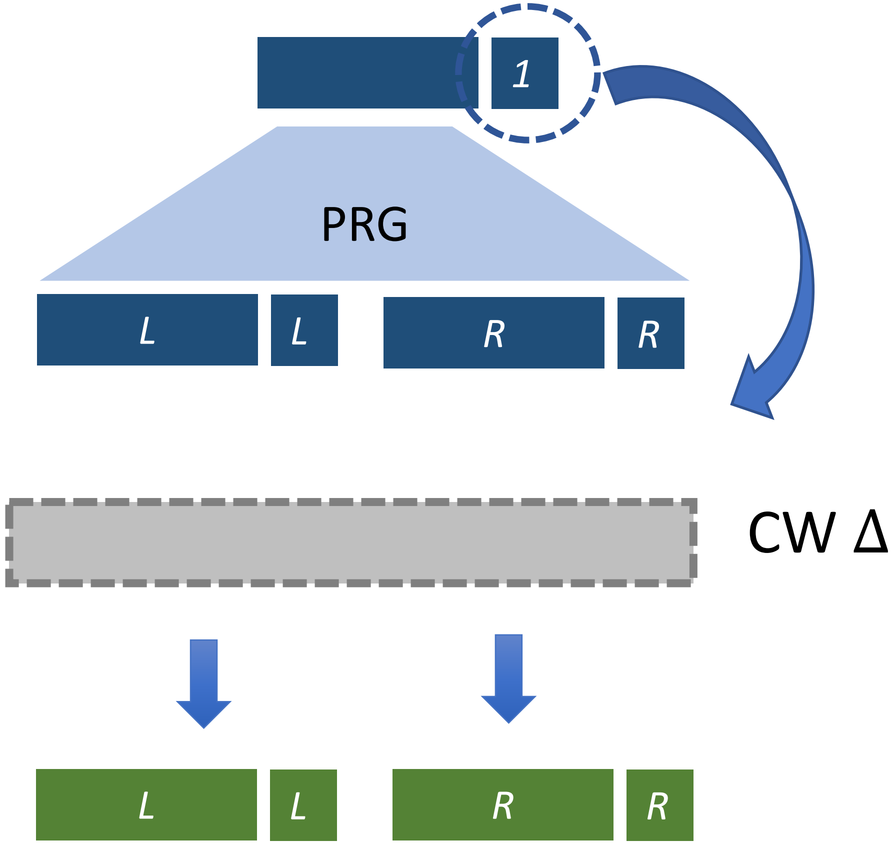


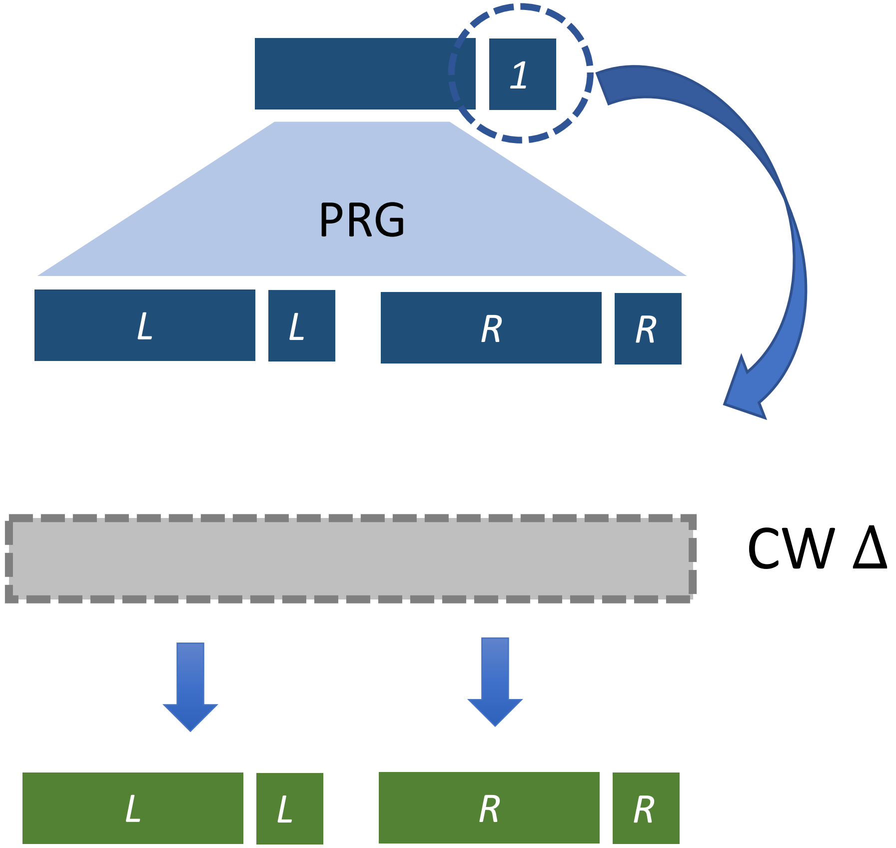


Using the CW Δ : Off-Path

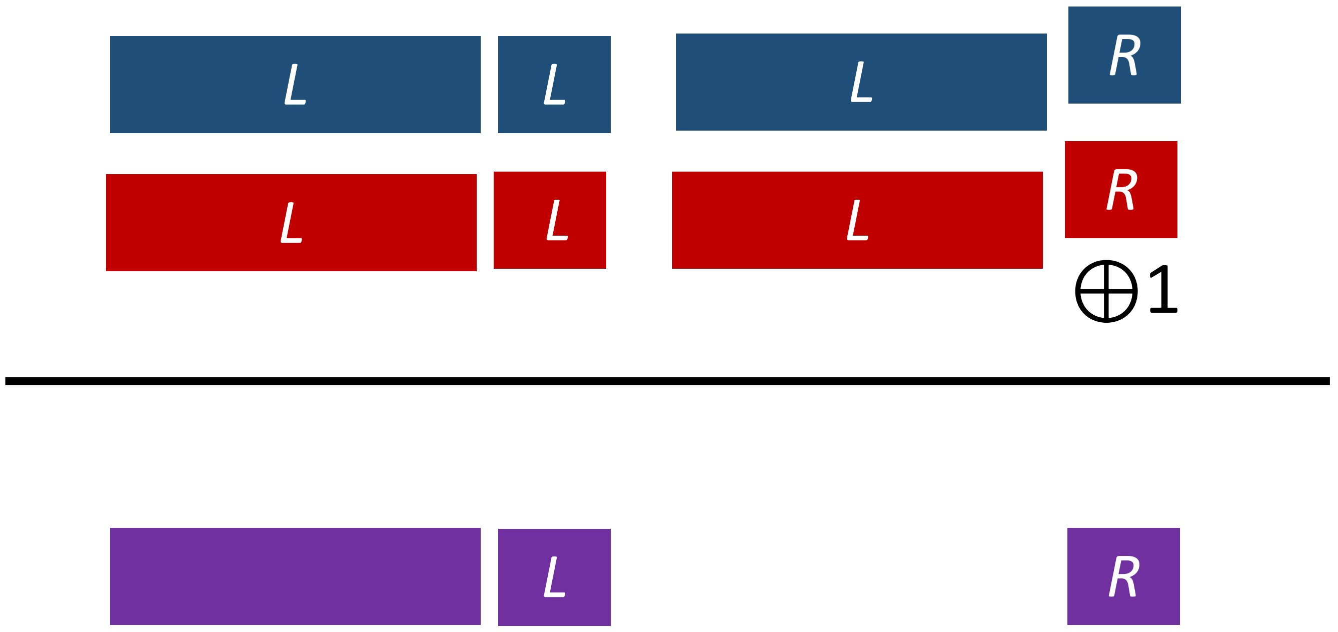


Using the CW Δ : Off-Path



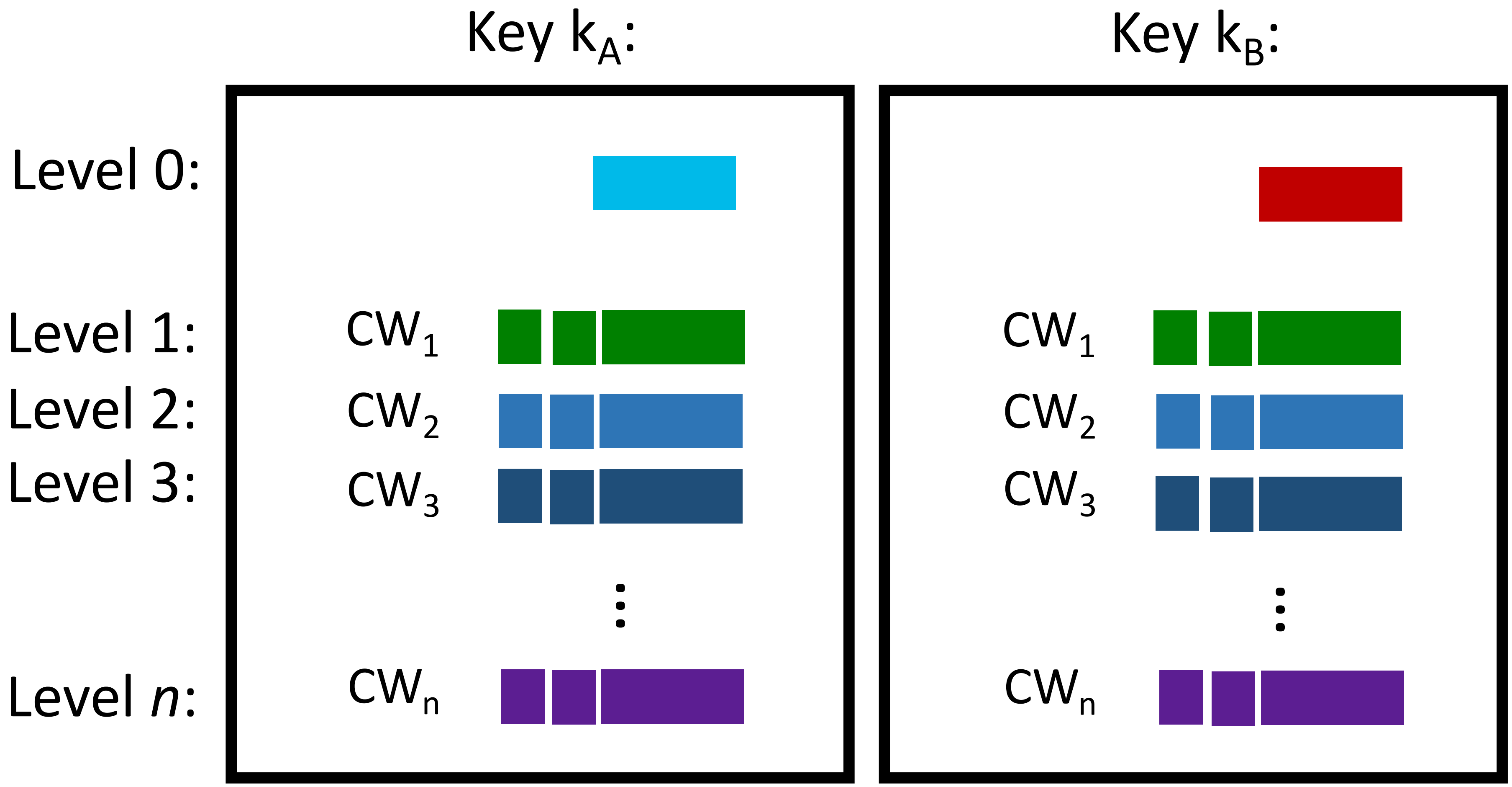


The DPF Keys: Correction Word per Level



Correction Word
For this level

DPF: Final Key Construction



$\lambda + (\lambda+2)n$ bits

DPF Construction: Complexity

[Boyle-Gilboa-Ishai 16b]

Domain
$[N] = [2^n]$

- Function share ("key") size:
  - PRG seed @ top λ bits
  - CW for n levels $(\lambda + 2)n$ total bits
- Generation / 1 evaluation cost:
  - n PRG evaluations (plus some xors)

s_0

CW_1

…

CW_n

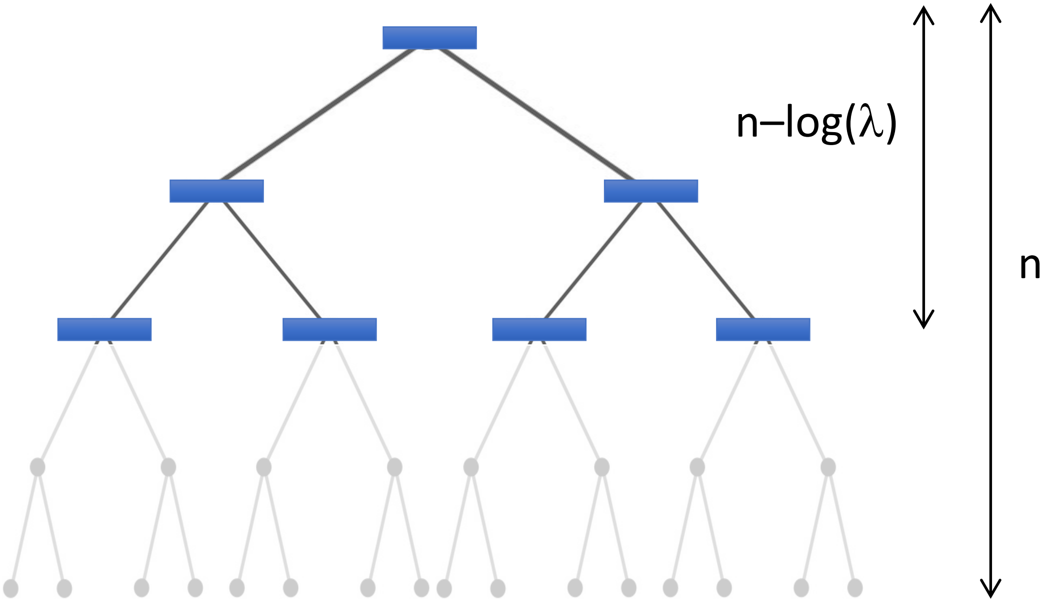
Example: PIR on 2^{25} records of length d

- Comm: 2578 bits → each server, d bits in return
- Comp: Dominated by reading + XORing all records

Optimizing PIR Applications

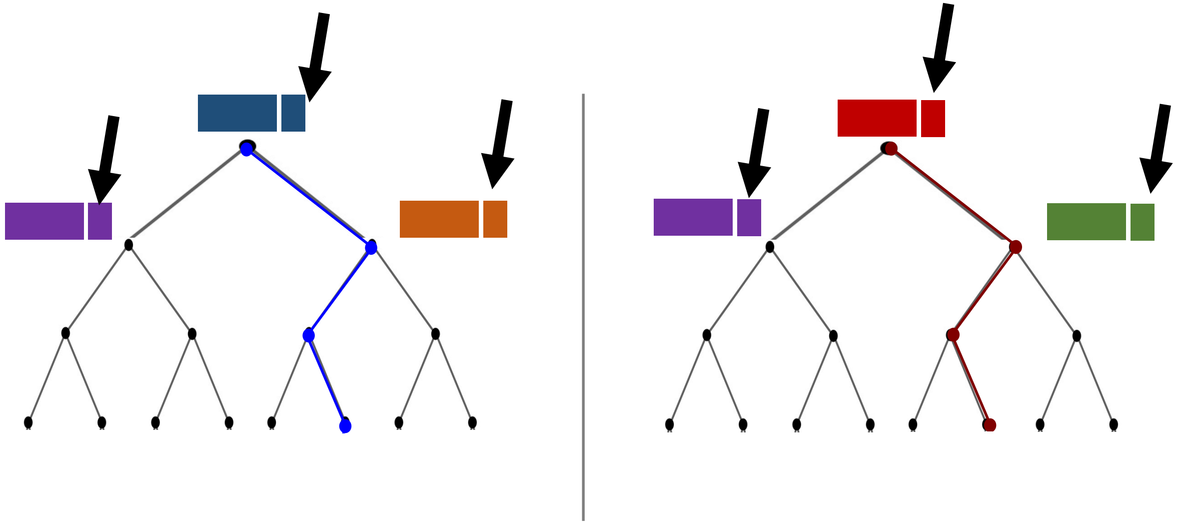
- Early termination: pack outputs into λ bits
- EvalAll: compute each *node* once

FSS computation costs dominated by lookup/xors

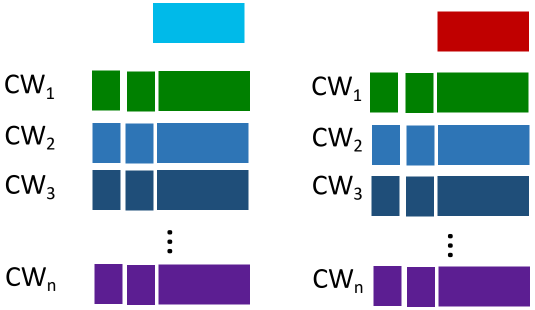


Observations on the Construction

- Incremental evaluation
  - Hidden **all-prefix FSS** inside!



- Almost everything is public
  - Ties hands of malicious key generator given public CW’s



- These properties are useful for applications! [BBCGI21]

Construction:

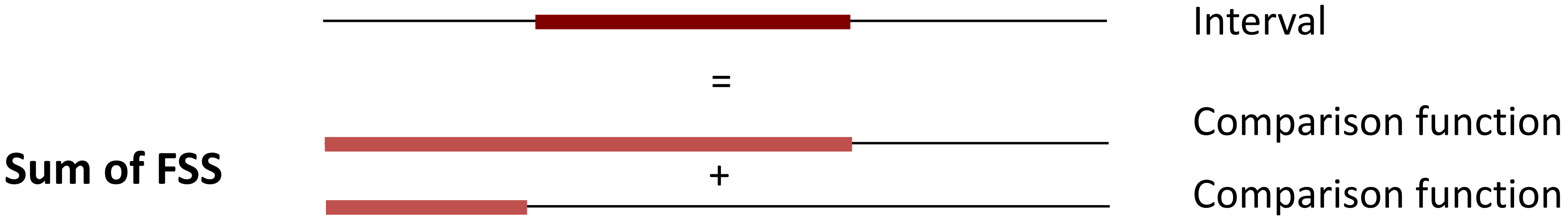
FSS for **Comparison Functions**

= Distributed **Comparison** Functions (D**C**F)

$$f_\alpha^{<}(x) = \begin{cases} 1 & \text{if } x < \alpha \\ 0 & \text{else} \end{cases}$$

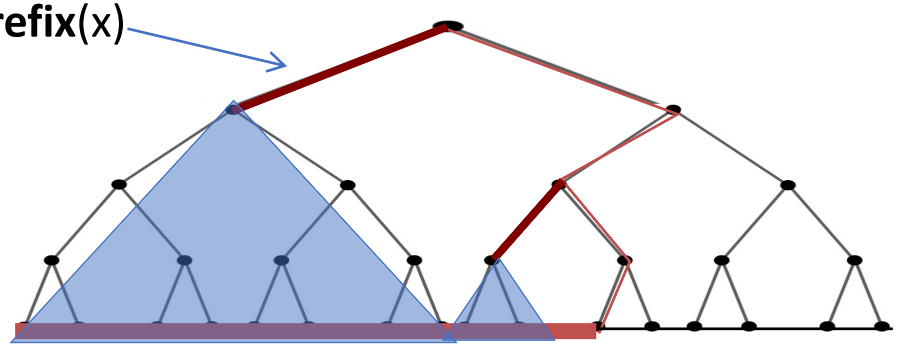
Warm-Up Observations

- 2 x DCF over $\{0,1\}$ ⇒ Intervals over $\{0,1\}$



- n x DPF ⇒ DCF (black box)

Point function applied to **Prefix**(x)



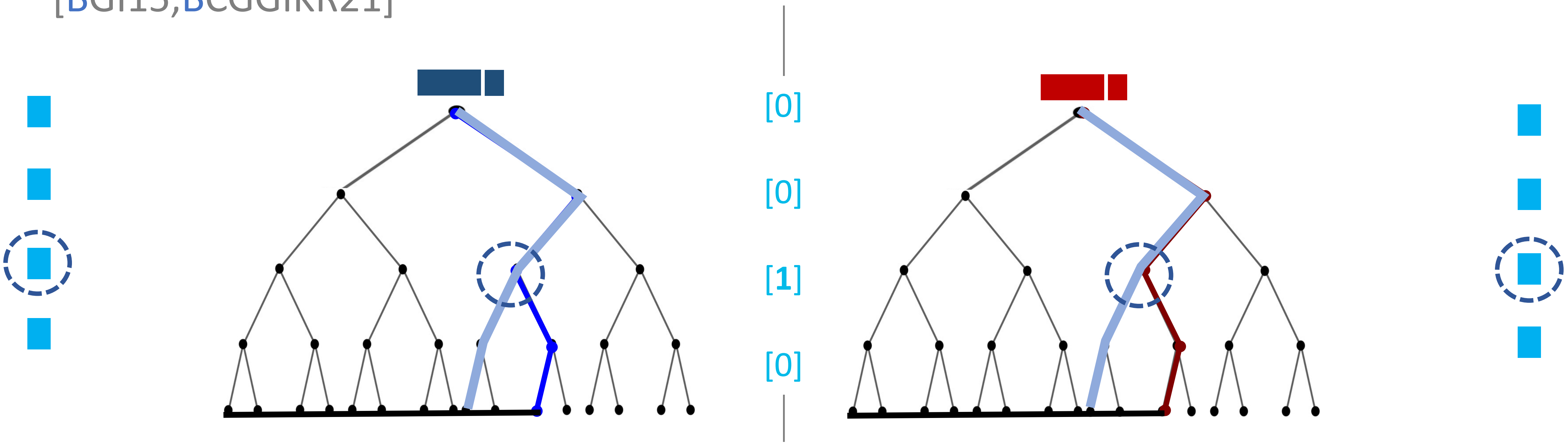
But: We can build non-black-box for much cheaper!

Note: almost like all-prefix DPF, but not quite… (co-paths)

$f_\alpha^<:\{0,1\}^n \rightarrow \{0,1\}$

D**C**F Construction from PRGs

[BGI15,BCGGIKR21]

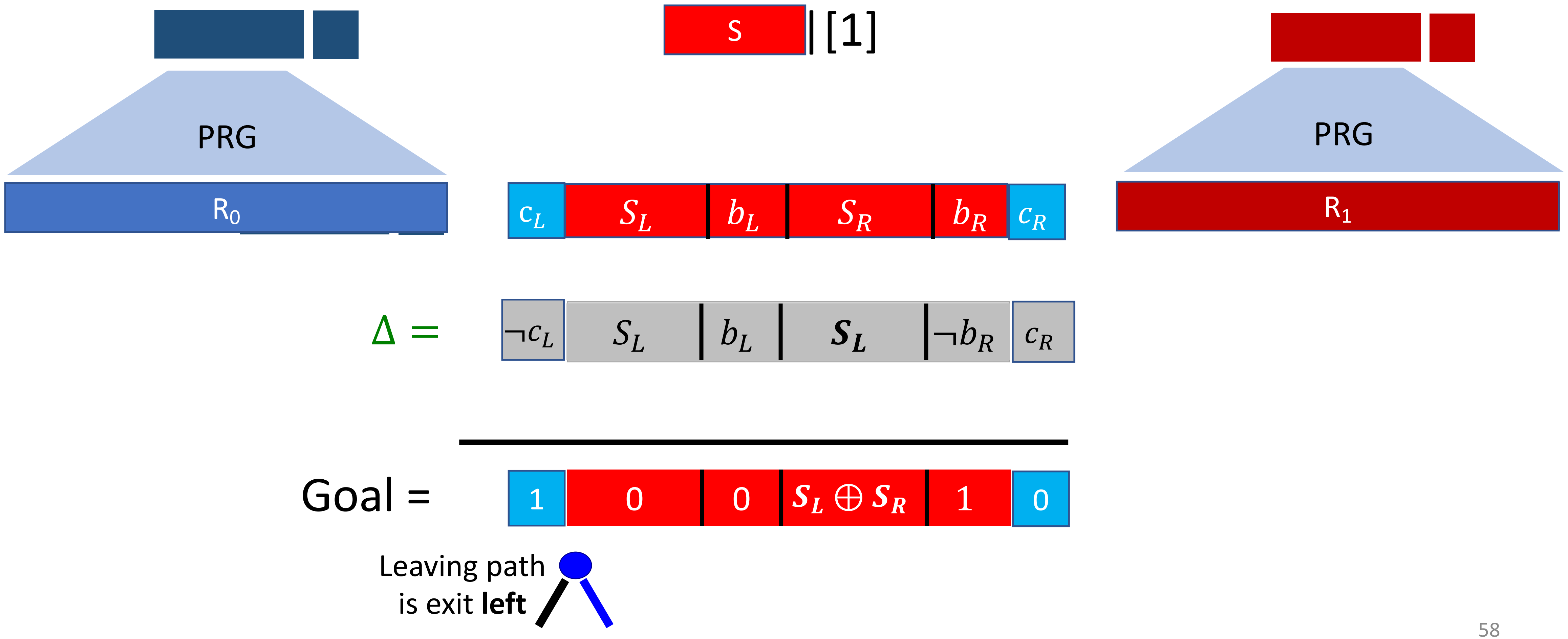


Same Per-Node Invariant for Eval (as DPF)

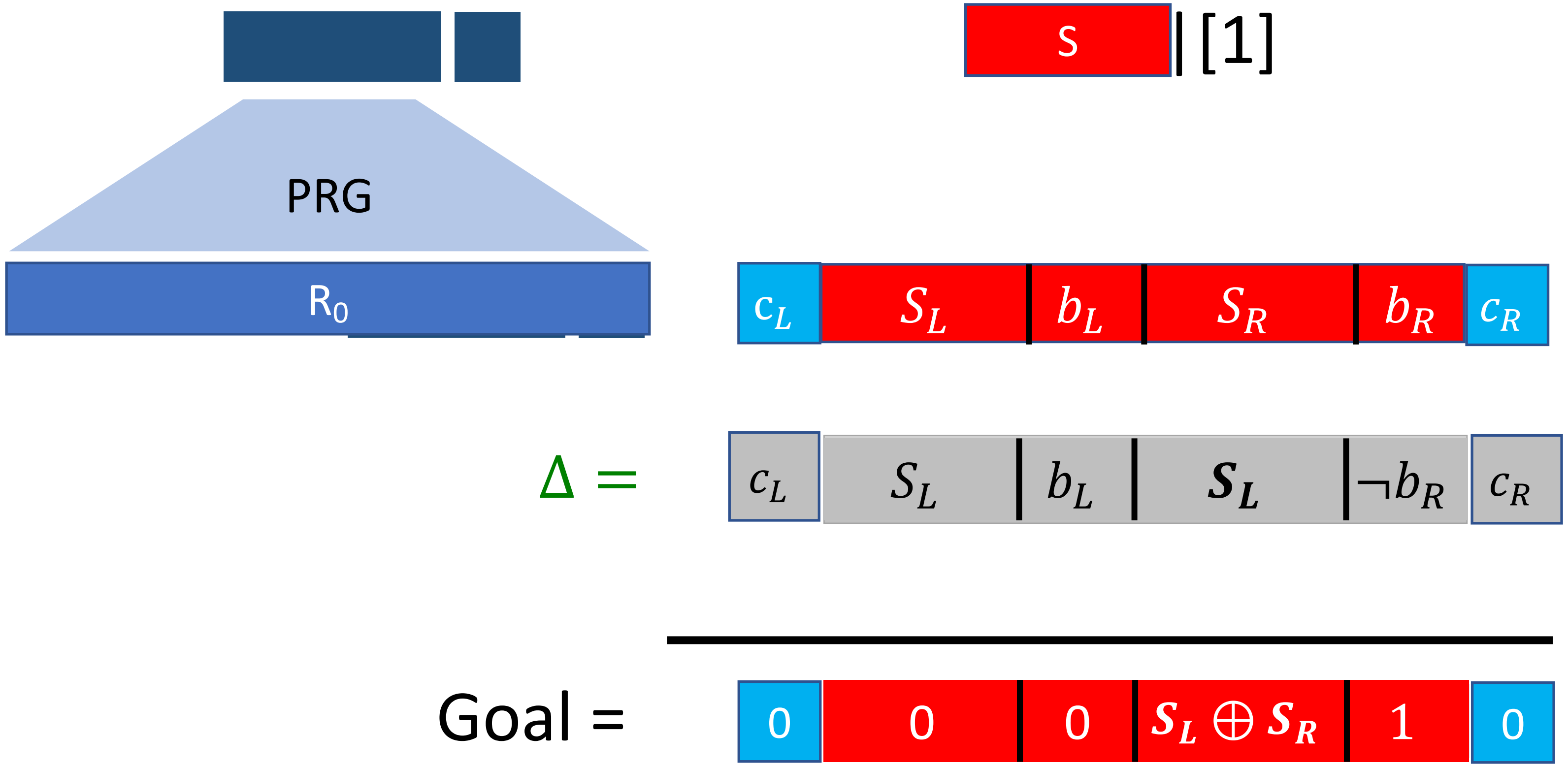
New: @ each level of Eval, compute **extra secret shared bit**

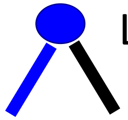
- Eval input x exits α-path **to the left** at this level $\Leftrightarrow$ bit shares 1
- Final output = DPF output + **sum of all levels' bits**

Building the Correction Word Δ



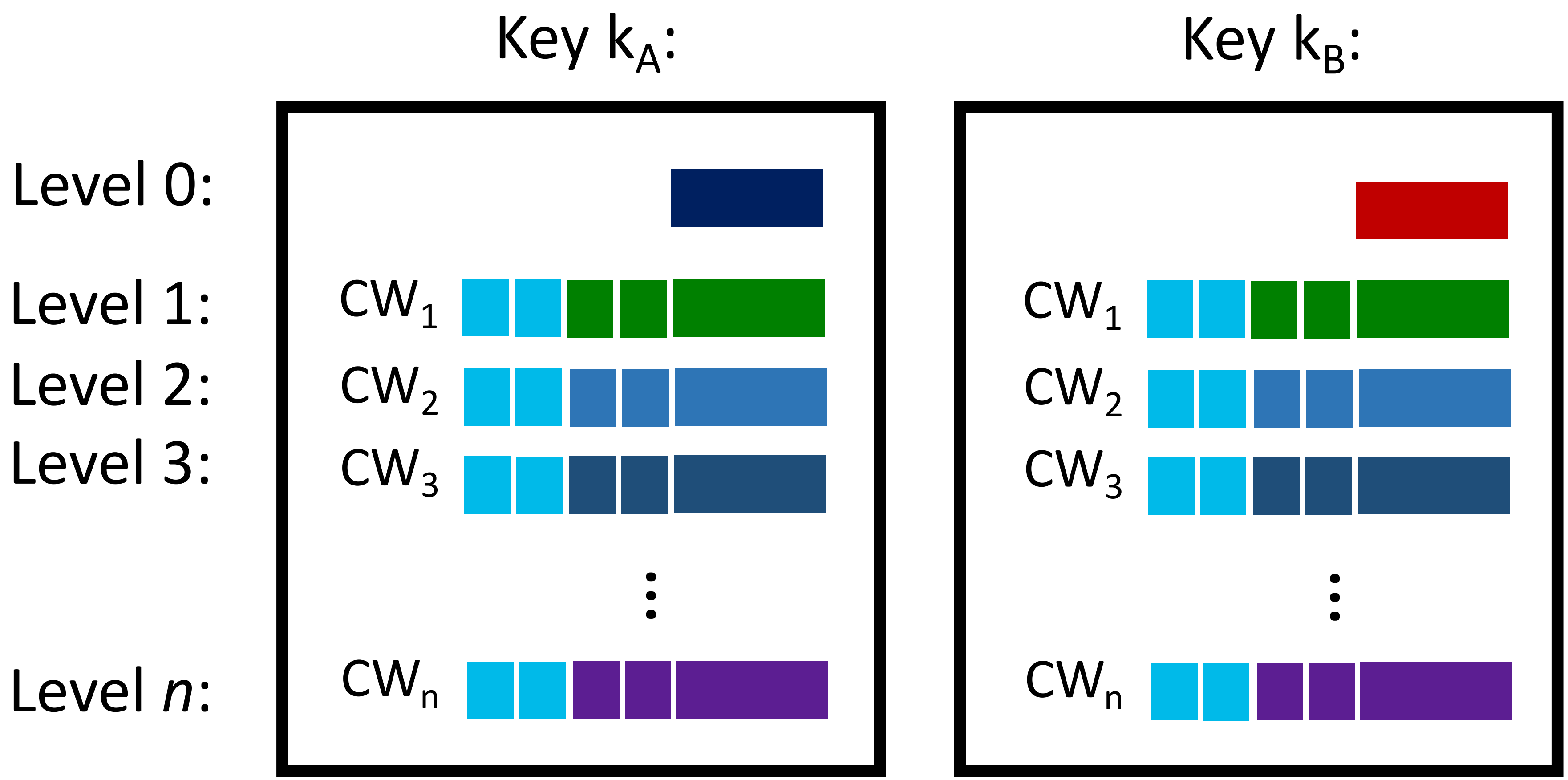
Building the Correction Word Δ





Leaving path
is exit **right**

DCF: Final Key Construction

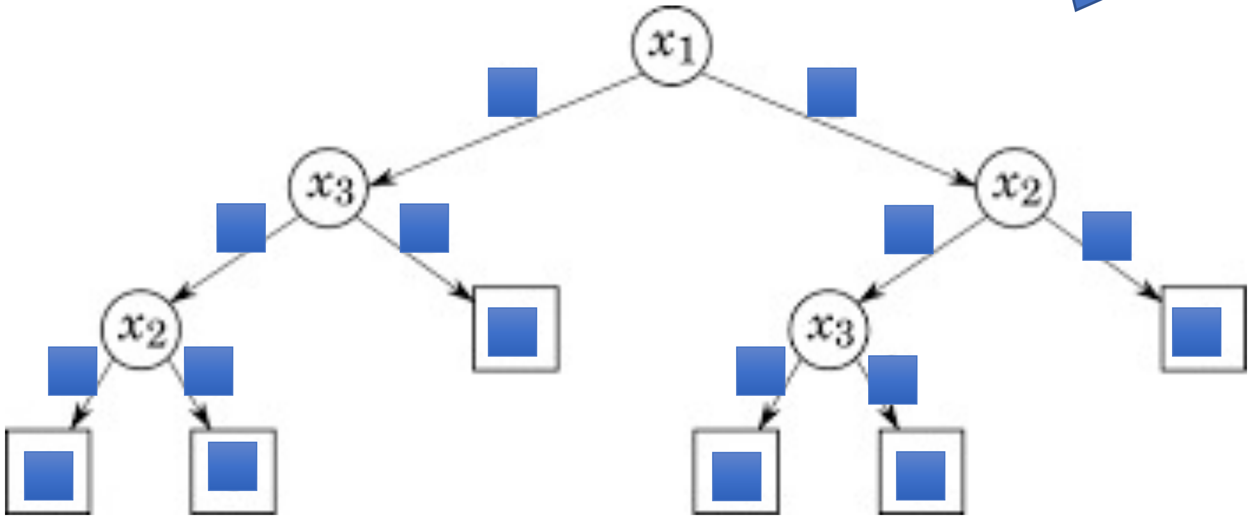


$\lambda + (\lambda\mathbf{+4})n$ bits

(Note: For general output group $\mathbb{G}$, each $\in \mathbb{G}$)

FSS for Decision Trees [BGI16b]

Note: DPF & DCF are special cases - Decision Lists



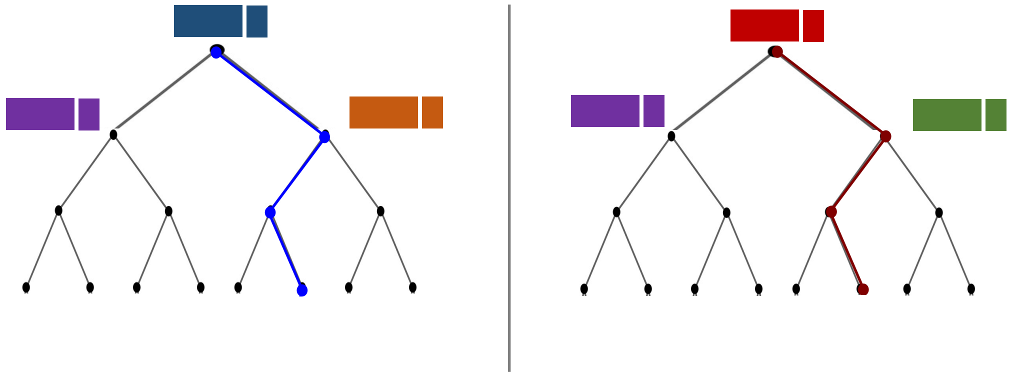
- Hides:
  - Edge labels
  - Leaf values
- Reveals:
  - Topology
  - Node labels

- Key size $\sim 4\lambda \cdot (\text{tree size})$
  Extends DPF/DCF but without optimizations

- Example application: k-dim intervals, $k \in O(1)$

Summary of Part II

- **Construction of DPF**
  - + Useful Properties



- Construction of DCF
  Distributed Comparison Function

- Briefly: FSS for Decision Trees

Part III:
Applications & Extensions

Application:

Secure Computation with Preprocessing

Secure (2-Party) Computation

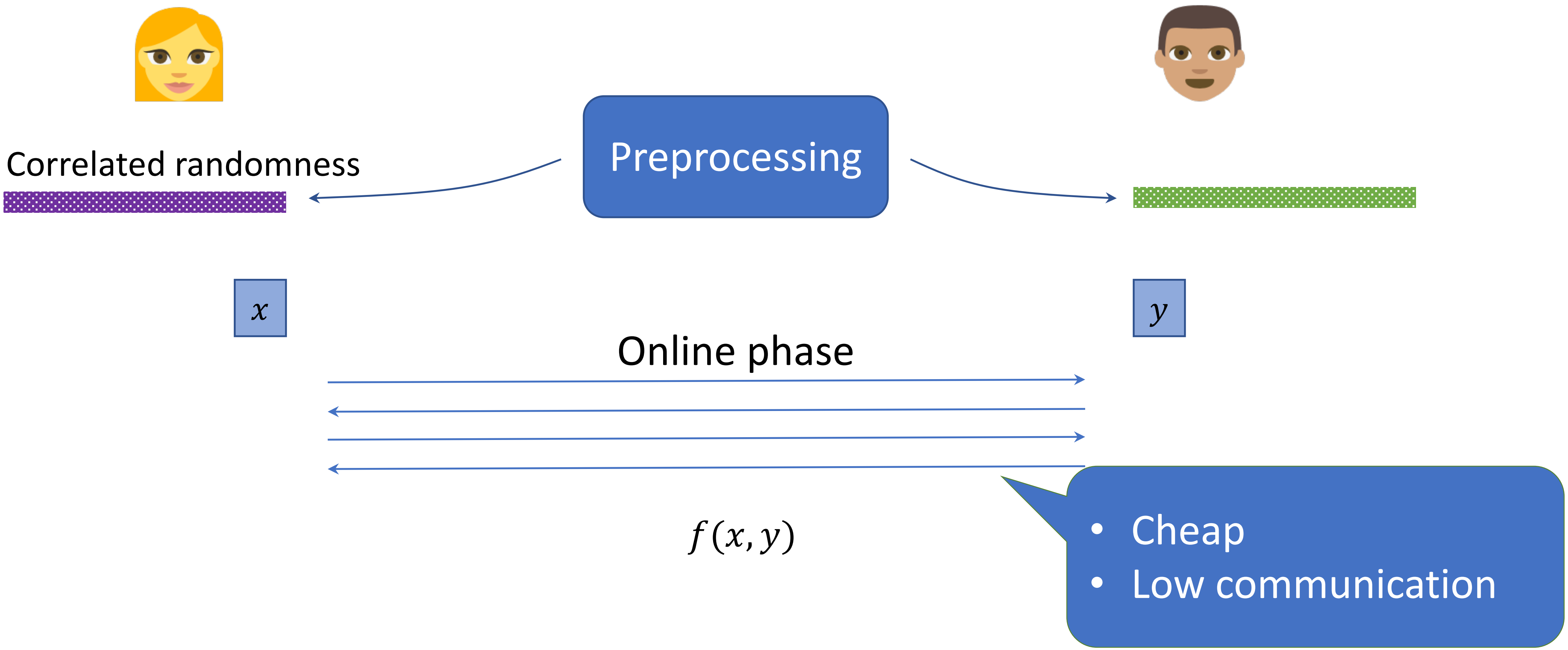
[Yao86,GMW87]

$f(x, y)$

Learn $f(x, y)$ and **nothing else** about x, y

Secure Computation with Preprocessing

[Beaver ’91]



Semi-Orthogonal Questions

- How to **<u>use</u>** correlations (& which are useful)?
  - Beaver triples, circuit-dependent Beaver [Bea91]
  - One-time truth tables (TinyTables) [IKMOP13, DNNR17]
  - Sublinear IT online comm for layered circuits [Cou19]
  - …

Now

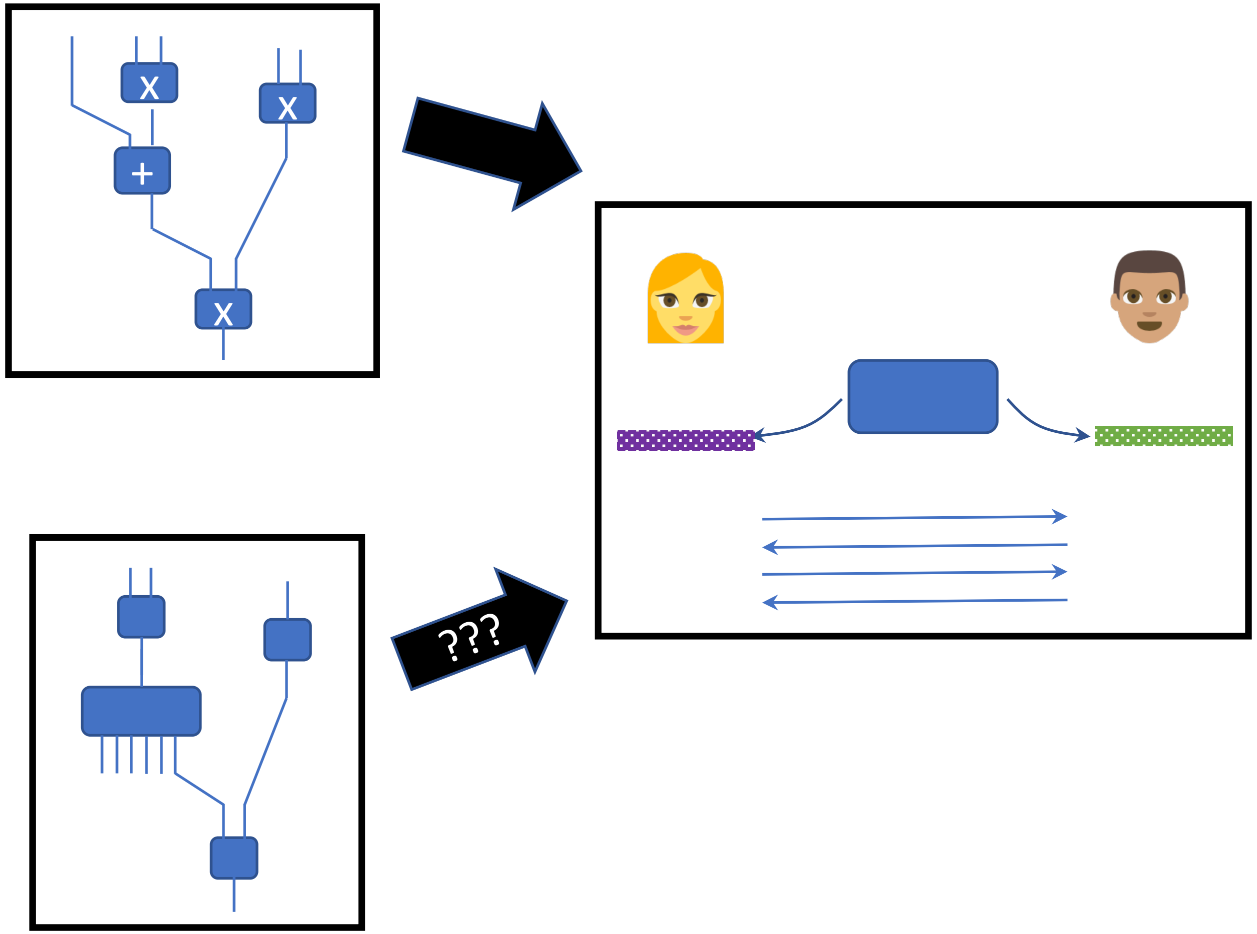
- How to **<u>generate</u>** correlations?

"Pseudorandom Correlation Generators"

Wed & Thurs! [BCGIKS19, BCGIKRS19,…]

Secure Computation with Preprocessing

- Arithmetic Circuit (+,x) over some ring R [Beaver'91]



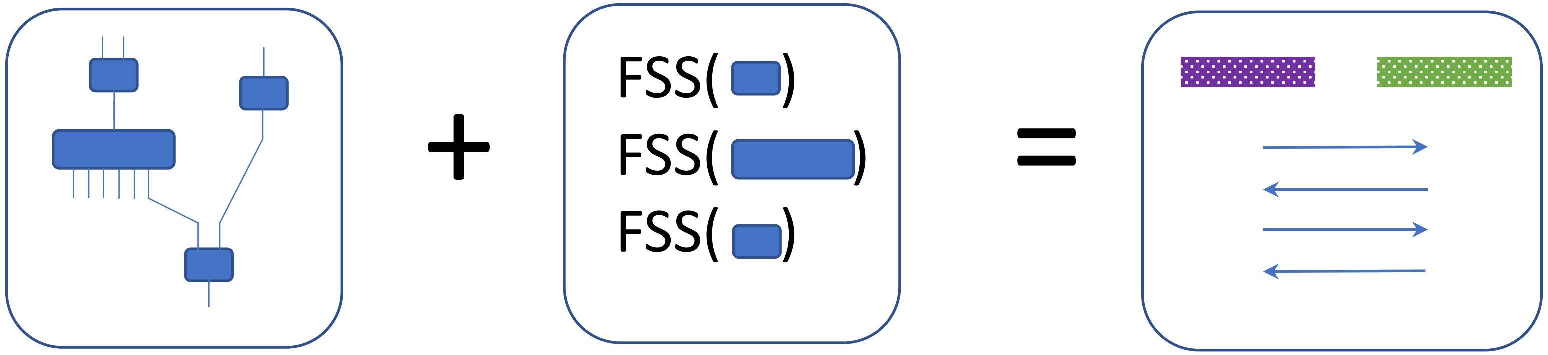
Goal:

- Possibly mixed domains (big)
- Useful nonlinear gates
  - Equality, Comparison, ReLU, Bit Decomposition, …

2PC with Preprocessing from FSS (High Level)

[BGI 19]

- General Framework: MPC with Preprocessing via FSS



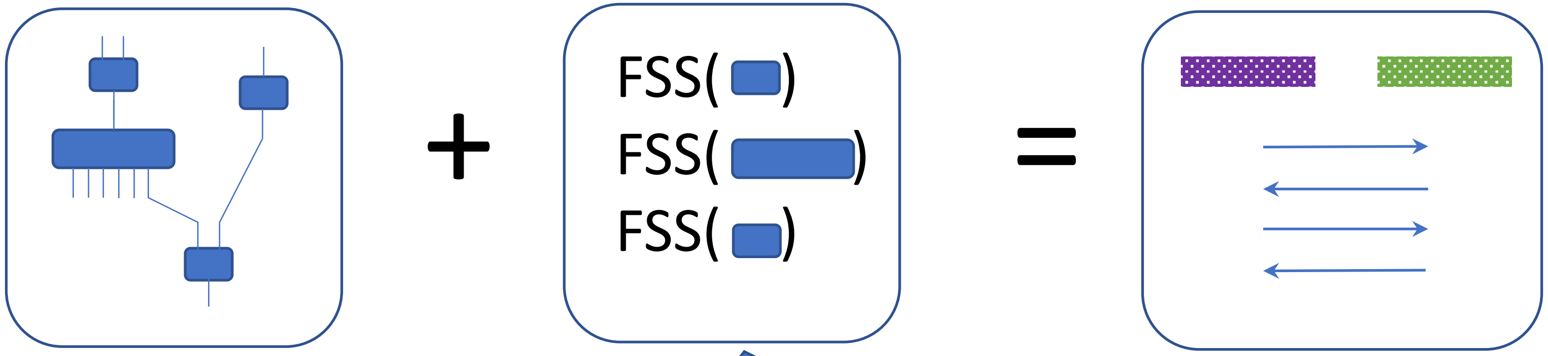
- Theoretical: Unifying approach
- Practical: Promising low-online-comm
  (equality, comparison, bit decomp,…)

- Necessity of FSS? “Shared equality” with <u>optimal</u> online communication ⇒ OWF

2PC with Preprocessing from FSS (High Level)

[BGI 19]

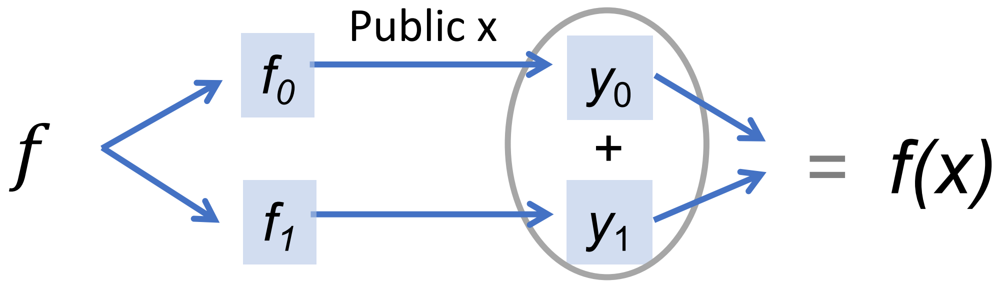
- General Framework: MPC with Preprocessing via FSS



“Secret Offset Functions”
$G(x-r)$ for gate G

Recall: Information-Theoretic FSS

- **Any** function class $\{ f : \{0,1\}^n \rightarrow \mathbb{G} \}$
  - Secret share the truth table



- Low-degree **polynomials** $\{ \sum_i \alpha_i x^i \}$
  - Secret share the coefficients α_i

- Function class $\{ \sum_i \alpha_i \; f_i(x) \}$ for **public** f_i
  - Secret share the coefficients α_i

Corollaries

- **Any** function class $\{ f: \{0,1\}^n \to \mathbb{G} \}$
  - Secret share the truth table

One-time truth tables [IKMOP13]
TinyTables [DNNR17]
(TT for local functions) [Cou19]

- Low-degree **polynomials** $\{ \sum_i \alpha_i x^i \}$
  - Secret share the coefficients α_i

Beaver triples [Bea91]
Circuit-dependent Beaver [DNNR17]

$$(x_1 - r_1)(x_2 - r_2) = x_1 x_2 - r_1 x_2 - x_1 r_2 + r_1 r_2$$

- Function class $\{ \sum_i \alpha_i \ f_i(x) \}$ for **public** f_i
  - Secret share the coefficients α_i

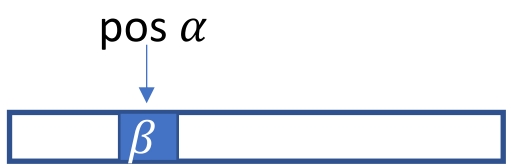
Degree-d gates
Bilinear maps, …

Lightweight FSS Constructions from OWF

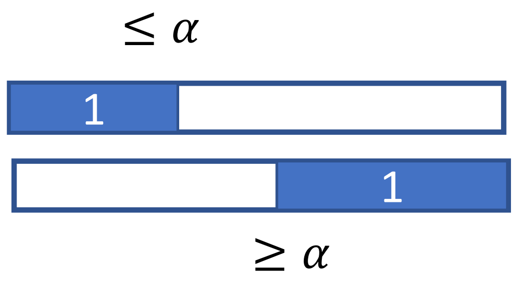
[BGI15, BGI16b]

General input groups too

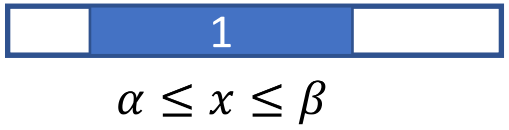
- Point Functions $f_{\alpha,\beta} : \{0,1\}^n \to \mathbb{G}$
  - Key size ~ $\lambda n + \log|\mathbb{G}|$ bits
  - Gen/Eval ~ n PRG evals



- "Special" Intervals
  - Cost $\leq$ Point Function x 2



- General Intervals
  - Cost $\leq$ Point Function x 4

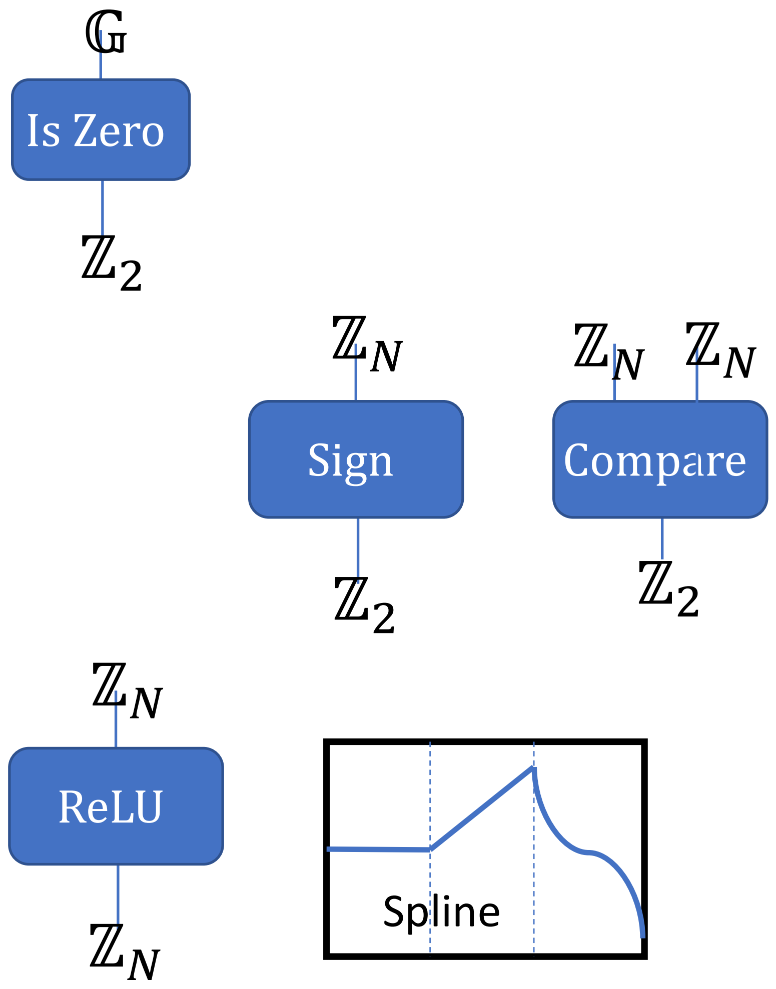


Corollaries from OWF

[BGI15, BGI16b, BGI19]

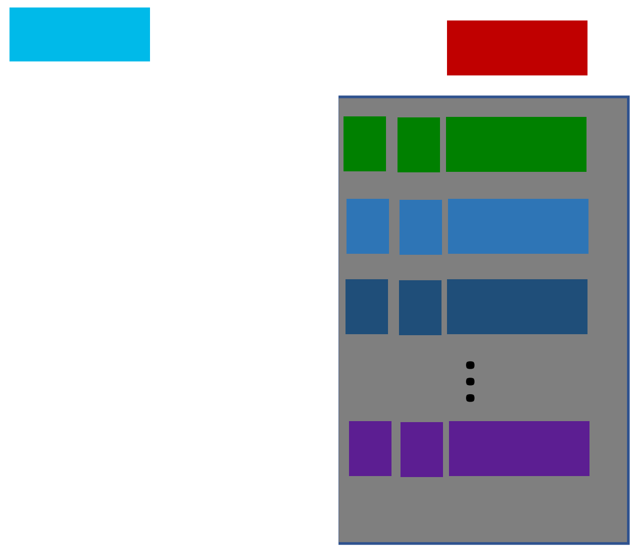
- Point Functions $f_{\alpha,\beta} : \{0,1\}^n \to \mathbb{G}$
  - Key size ~ $\lambda n + \log|\mathbb{G}|$ bits
  - Gen/Eval ~ n PRG evals
- "Special" Intervals
  - Cost $\leq$ Point Function x 2
- General Intervals
  - Cost $\leq$ Point Function x 4

<u>2PC with Preprocessing for:</u>



Other Cool FSS Things

“Programmable” DPF [BGIK??]



- One key is λ bits
- Builds on “Puncturable Pseudorandom Sets” of [CK20] (from online/offline PIR)
- Very different DPF structure!
  - Punctured histogram
  - Amplify 1/poly error → negligible

Multi-Party DPF (Security Against t>1)

- Bottom Line: Sort of sucks. [Boyle, personal communication '22]
  - Eg [BGI15]: 2 parties, t =1 | 3 parties, t = 2 | m parties, t = m-1
    Key size: $\sim \boldsymbol{n}\lambda$ | $O(\mathbf{2}^{\boldsymbol{n}/\mathbf{2}}\lambda)$ | $O(\mathbf{2}^{\boldsymbol{m}} \cdot 2^{n/2}\lambda)$

- The reason: 2 parties ⇒ Shares of 0 are **identical** values (leveraged!)

- Improvements given gap between # parties & # corrruptions [BKO21]
  - Eg: 5 parties, 2 corruptions, $O(2^{\boldsymbol{n}/\mathbf{4}})$ instead of $O(2^{\boldsymbol{n}/\mathbf{2}})$

Relation to Other Crypto Objects

- **“Nontrivial” FSS** ⇒ **OWF** [GI14,BGI15]
  Functions f_0, f_1 must be PRFs [BGI15]

- FSS for **Class containing SKE Dec circuit**
  ⇒ (amortized) **succinct secure computation** [BGI15]

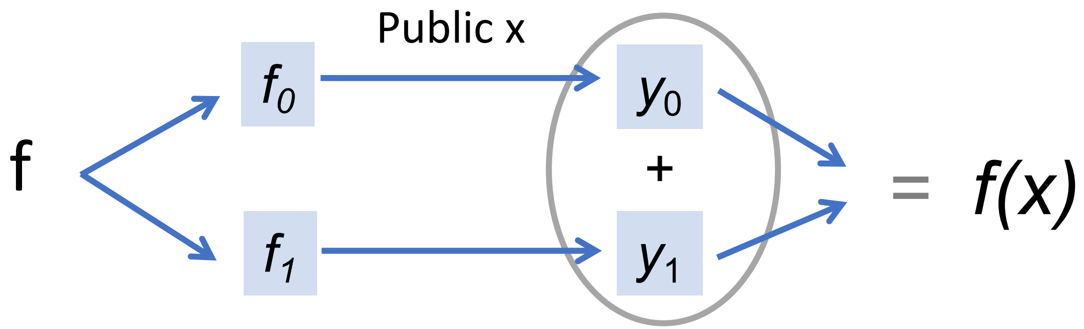
- **Privately Puncturable PRF** [BLW17] ⇒ “adaptive” DPF
  Can set 1 key before knowing the secret α

- **Targeted Lossy Functions** [QWW21]
  DPF **equivalent** to “Targeted All-Lossy-But-One” functions

FSS: Summary

Lecture Conclusion – Part I

- Function Secret Sharing (FSS)

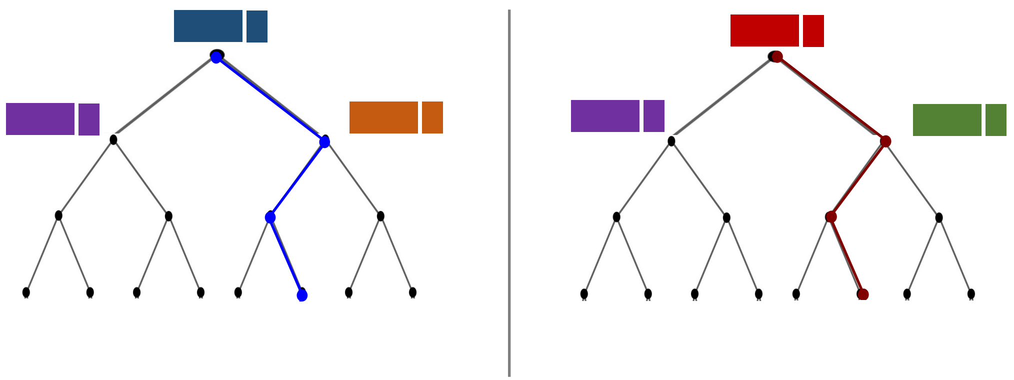


- Approach to 2-server private DB queries / updates (+ more!)

- Current FSS: Richness vs complexity tradeoff
  - Simple functions: Lightweight from any PRG
  - NC^1: Uses public-key crypto, but getting reasonable
  - Above: Heavy crypto…

Lecture Conclusion – Part II

- **Construction of DPF**
  - + Useful Properties

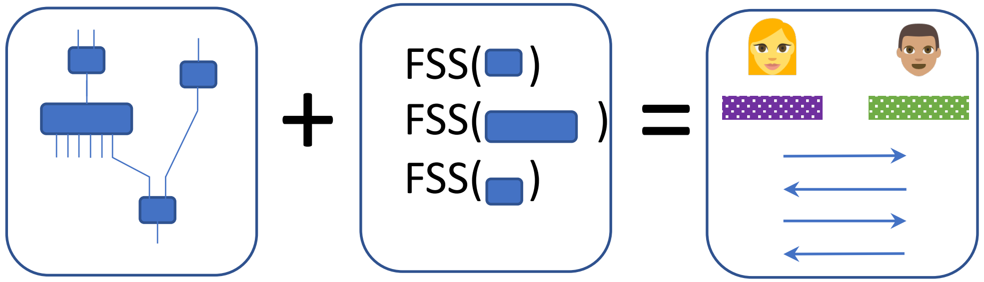


- Construction of DCF
  Distributed Comparison Function

- Briefly: FSS for Decision Trees

Lecture Conclusion – Part III

- Application: 2PC with Preprocessing



- Other Highlights
  - “Programmable” DPF
  - Multi-Party DPF
  - Relation to other primitives

Some Things We **Don't** Know

FSS: Sample Open Problems

- **Richer FSS** from OWF
  - Broader function classes **(CNF/DNF?)**
    Barriers known for > AC^0
  - 3-server FSS with **security against 2 servers**
    To beat: key size ($\lambda\ 2^{n/2}$)   vs (λn) for security against 1
- More **efficient FSS**
  - 2-server FSS for Point Functions from OWF: **Beat λn** key size?
  - Amortizing cost of **multi-point function**?
  - Better efficiency from **"mid-level"** constructions
- New & improved **applications**

Coming up next…

What About **Malicious** Parties?