Pseudorandom Correlation Functions for Garbled Circuits and Applications

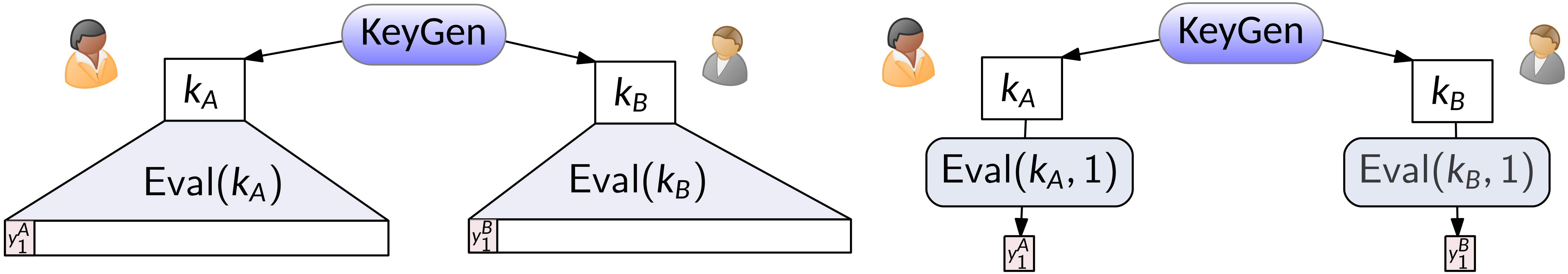
Eurocrypt 2024 · Submission #145

No Author Given *

January 22, 2024· Presented by Hongrui Cui

* For the Purpose of Reviewing and Internal Discussion **DO NOT DISTRIBUTE**

Introduction



Correlation Examples

- $y_1^A = y_1^B$
- $y_1^A = (w_1, \Delta), y_2^B = (u_1, v_1),$ s.t. $w_1 = v_1 + u_1 \cdot \Delta$
- $y_1^A + y_1^B = (a, b, a \cdot b)$

- Technical contributions of this paper

$\mathsf{PCG.KeyGen}(\mathcal{C}) \mapsto (k_A, k_B)$

$\mathsf{PCG.Eval}(k_{A/B}) \mapsto (\mathsf{GC}^1_{A/B}, ..., \mathsf{GC}^n_{A/B})$

$\mathsf{GC}^i_A \oplus \mathsf{GC}^i_B = \mathsf{Garble}(\mathcal{C})$

$\mathsf{PCF.KeyGen}() \mapsto (k_A, k_B)$

$\mathsf{PCF.Eval}(k_{A/B}, \mathcal{C}, i) \mapsto \mathsf{GC}^i_{A/B}$

$\mathsf{GC}^i_A \oplus \mathsf{GC}^i_B = \mathsf{Garble}(\mathcal{C})$

Sharing Friendly Garbled Circuit

$\rho_{i,\alpha}$	$\rho_{j,\beta}$	$(2\rho_{i,\alpha}+\rho_{j,\beta})$	Truth table	Garbling of G
0	0	0	$\rho_{00} = (\mathsf{r}_i \wedge \mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L_{i,0} \| L_{j,0} \| k \| 0) \oplus (\rho_{00} \| L_{k,\rho_{00}})$
0	1	1	$\rho_{01} = (\mathsf{r}_i \wedge \neg\mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L_{i,0} \| L_{j,1} \| k \| 1) \oplus (\rho_{01} \| L_{k,\rho_{01}})$
1	0	2	$\rho_{10} = (\neg\mathsf{r}_i \wedge \mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L_{i,1} \| L_{j,0} \| k \| 2) \oplus (\rho_{10} \| L_{k,\rho_{10}})$
1	1	3	$\rho_{11} = (\neg\mathsf{r}_i \wedge \neg\mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L_{i,1} \| L_{j,1} \| k \| 3) \oplus (\rho_{11} \| L_{k,\rho_{11}})$

Each party has its own label

$\rho_{i,\alpha}$	$\rho_{j,\beta}$	Truth table	Secret-shared garbling of G
0	0	$\rho_{00} = (\mathsf{r}_i \wedge \mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L^A_{i,0} \| L^A_{j,0} \| k \| 0) \oplus \mathsf{H}(L^B_{i,0} \| L^B_{j,0} \| k \| 0) \oplus (\rho_{00} \| L^A_{k,\rho_{00}} \| L^B_{k,\rho_{00}})$
0	1	$\rho_{01} = (\mathsf{r}_i \wedge \neg\mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L^A_{i,0} \| L^A_{j,1} \| k \| 1) \oplus \mathsf{H}(L^B_{i,0} \| L^B_{j,1} \| k \| 1) \oplus (\rho_{01} \| L^A_{k,\rho_{01}} \| L^B_{k,\rho_{01}})$
1	0	$\rho_{10} = (\neg\mathsf{r}_i \wedge \mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L^A_{i,1} \| L^A_{j,0} \| k \| 2) \oplus \mathsf{H}(L^B_{i,1} \| L^B_{j,0} \| k \| 2) \oplus (\rho_{10} \| L^A_{k,\rho_{10}} \| L^B_{k,\rho_{10}})$
1	1	$\rho_{11} = (\neg\mathsf{r}_i \wedge \neg\mathsf{r}_j) \oplus \mathsf{r}_k$	$\mathsf{H}(L^A_{i,1} \| L^A_{j,1} \| k \| 3) \oplus \mathsf{H}(L^B_{i,1} \| L^B_{j,1} \| k \| 3) \oplus (\rho_{11} \| L^A_{k,\rho_{11}} \| L^B_{k,\rho_{11}})$

- $L^{A/B}_{i,0}$ can be generated and hashed locally
- We only need to generate shares of $(r_i, r_j, r_i \cdot r_j) \otimes (1, \Delta_A, \Delta_B)$
- The idea is to use Ring-LPN or EA-LPN

- Remaining problems: How to get input labels?
- Recall that the evaluator can only get one label for each input wire

Garbling Pseudorandom Correlation Generator

- Recall the Ring-LPN based PCG for OLE and it's **programable** property

$R_q = F_q[X]/f(X)$

$e \leftarrow \mathsf{HW}_t$ s.t.

$e = e_{i_1} X^{i_1} + \ldots + e_{i_t} X^{i_t}$

Ring-LPN: $a \leftarrow R_q;\, s, e \leftarrow \mathsf{HW}_t$

$(a, as + e) \approx (a, U)$

Module-LPN: $a_1, \ldots, a_c \leftarrow R_q;\, s_1, \ldots, s_c, e \leftarrow \mathsf{HW}_t$

$(a_1, \ldots, a_c, a_1 s_1 + \ldots + a_c s_c + e) \approx (a_1, \ldots, a_c, U)$

- Using Ring-LPN, we can express the quadratic relation using only $(2t)^2$ terms (FSS keys)

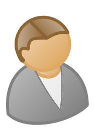
$a \in R_q$ is public $\quad \mathbf{x} = a \cdot s_1 + e_1 \quad \mathbf{y} = a \cdot s_2 + e_2 \quad \mathbf{x} * \mathbf{y} = (a, 1) \otimes (a, 1) \cdot (s_1, e_1) \otimes (s_2, e_2)$

$\mathsf{FSS.KGen}(s_1, e_1)$ $\quad$ $\mathsf{FSS.KGen}(s_2, e_2)$ $\quad$ $\mathsf{FSS.KGen}((s_1, e_1) \otimes (s_2, e_2))$

$\boxed{k_A} = (\ k_A^1 \quad k_A^2 \quad k_A^3\ )$

$\langle x/y \rangle^A = (a, 1) \cdot \mathsf{Eval}(k_A^1 / k_A^2)$

$\langle z \rangle^A = (a, 1) \otimes (a, 1) \cdot \mathsf{Eval}(k_A^3)$



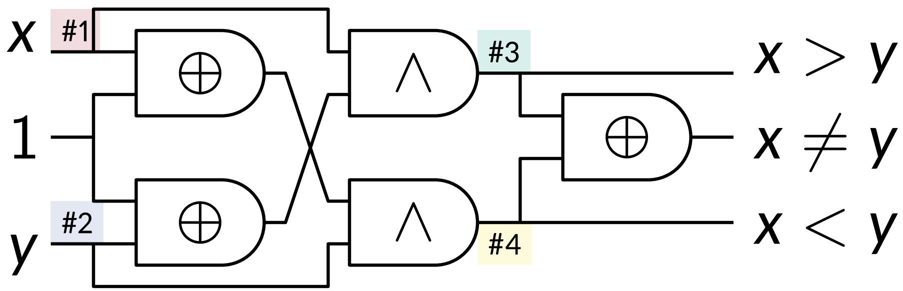
$\boxed{k_B} = (\ k_B^1 \quad k_B^2 \quad k_B^3\ )$

$\langle x/y \rangle^B = (a, 1) \cdot \mathsf{Eval}(k_B^1 / k_B^2)$

$\langle z \rangle^B = (a, 1) \otimes (a, 1) \cdot \mathsf{Eval}(k_B^3)$

Garbling Pseudorandom Correlation Generator

- Idea: One Ring-LPN instance for each AND-output wire



Prepare

s_1, e_1 s_2, e_2 s_3, e_3 s_4, e_4

$(s_1, e_1) \otimes (s_2, e_2)$

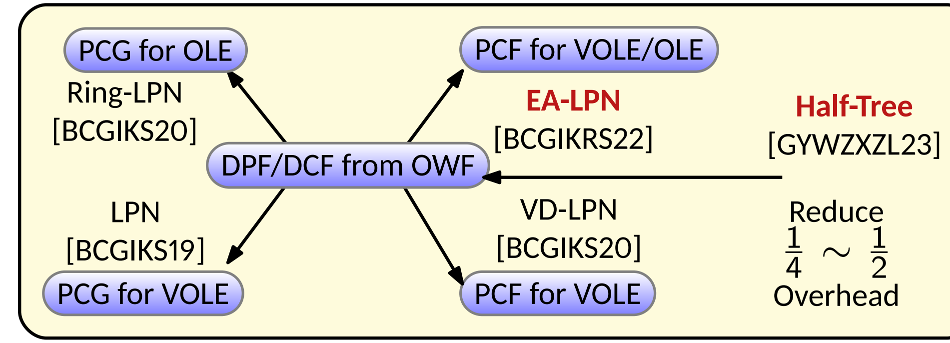
FSS.KeyGen

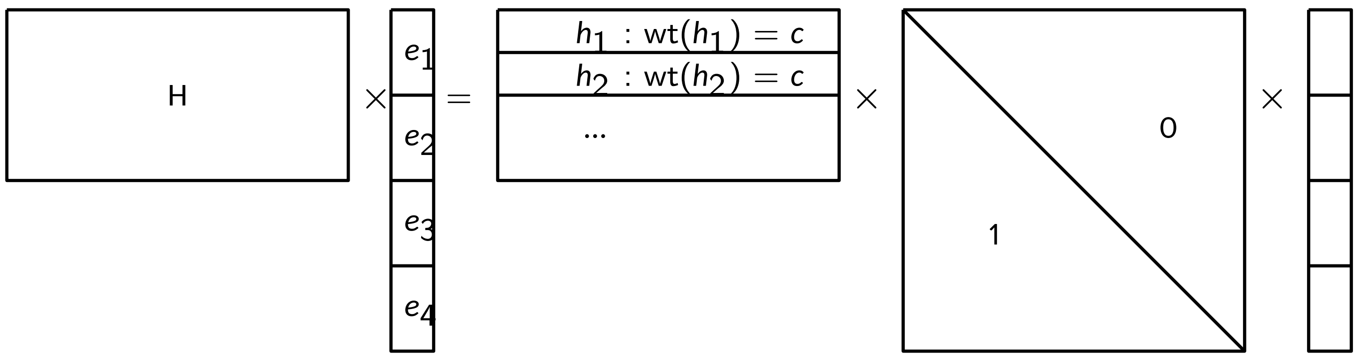
- PCG output length = # Fresh Garbled Circuit

- Discussion
- PCG key size proportional to circuit size
- Not very good

Garbling Pseudorandom Correlation Function

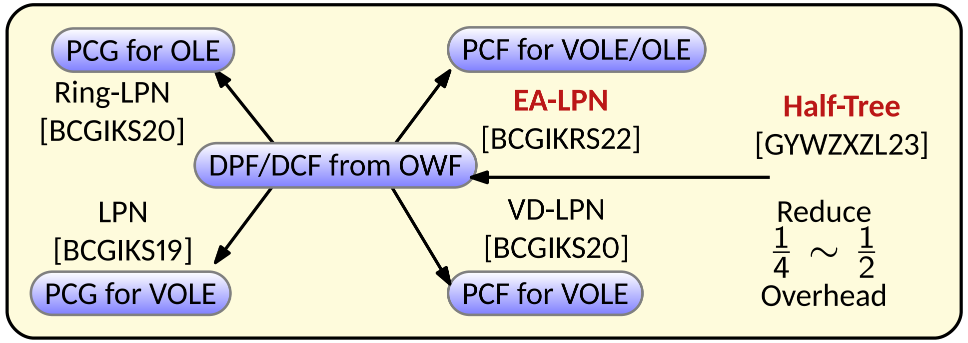
- PCF relies on special LPN flavors (i.e. VD-LPN, EA-LPN)
- This paper uses EA-LPN (Crypto'22)
- (i-th row) $h_i = \mathrm{HW}_c \cdot$

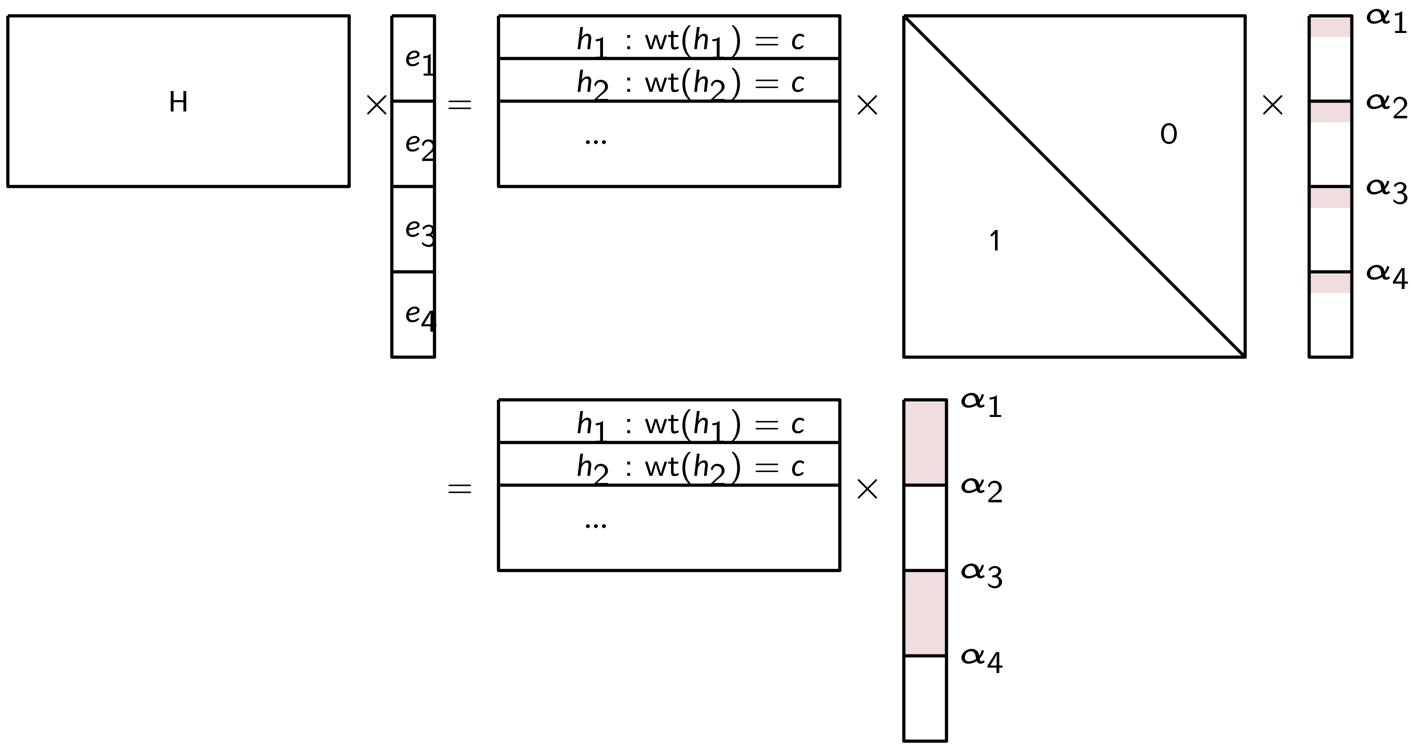




Garbling Pseudorandom Correlation Function

- PCF relies on special LPN flavors (i.e. VD-LPN, EA-LPN)
- This paper uses EA-LPN (Crypto'22)
- (i-th row) $h_i = \mathrm{HW}_c \cdot$ ◺

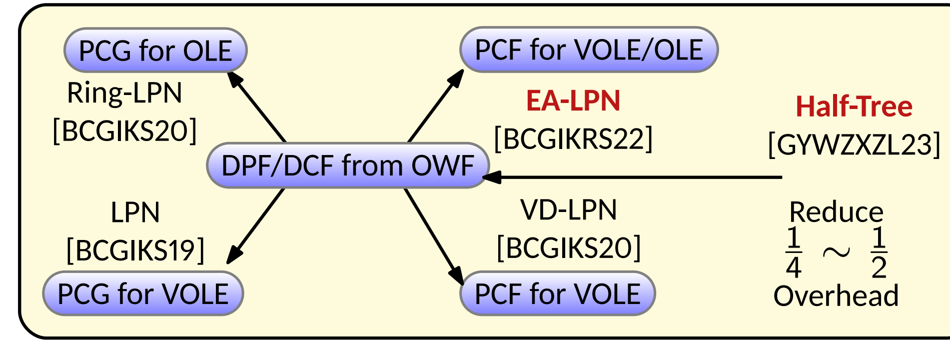




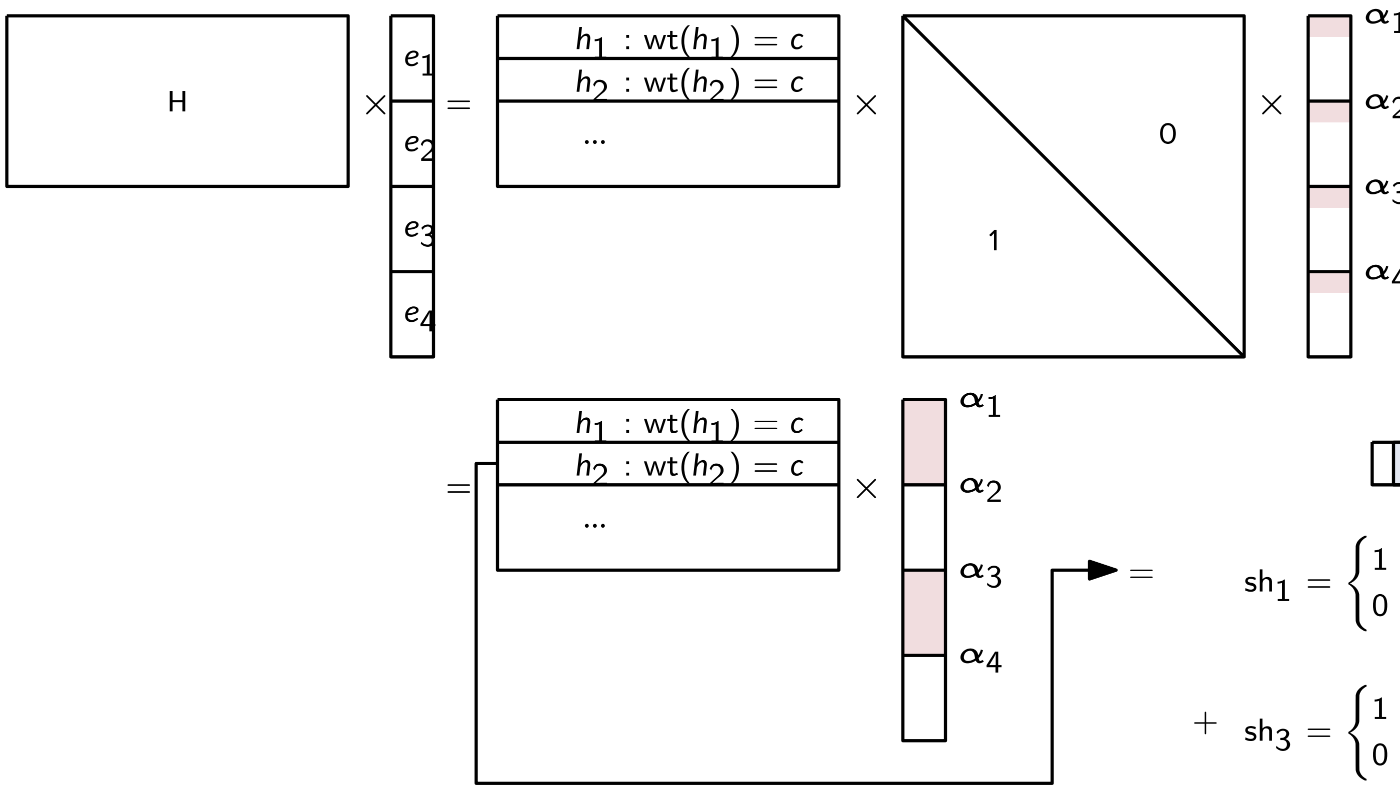
For the easy of demonstration
we assume the first entry in each block is one

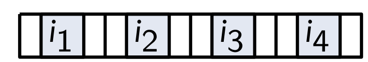
Garbling Pseudorandom Correlation Function

- PCF relies on special LPN flavors (i.e. VD-LPN, EA-LPN)
- This paper uses EA-LPN (Crypto'22)
- (i-th row) $h_i = \mathrm{HW}_c \cdot \triangle$



For the easy of demonstration
we assume the first entry in each block is one



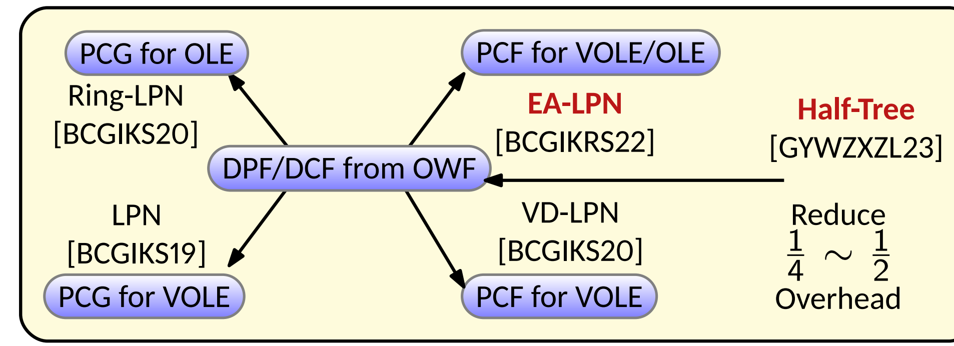


$$\mathrm{sh}_1 = \begin{cases} 1 & i_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_2 = \begin{cases} 1 & i_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases}$$

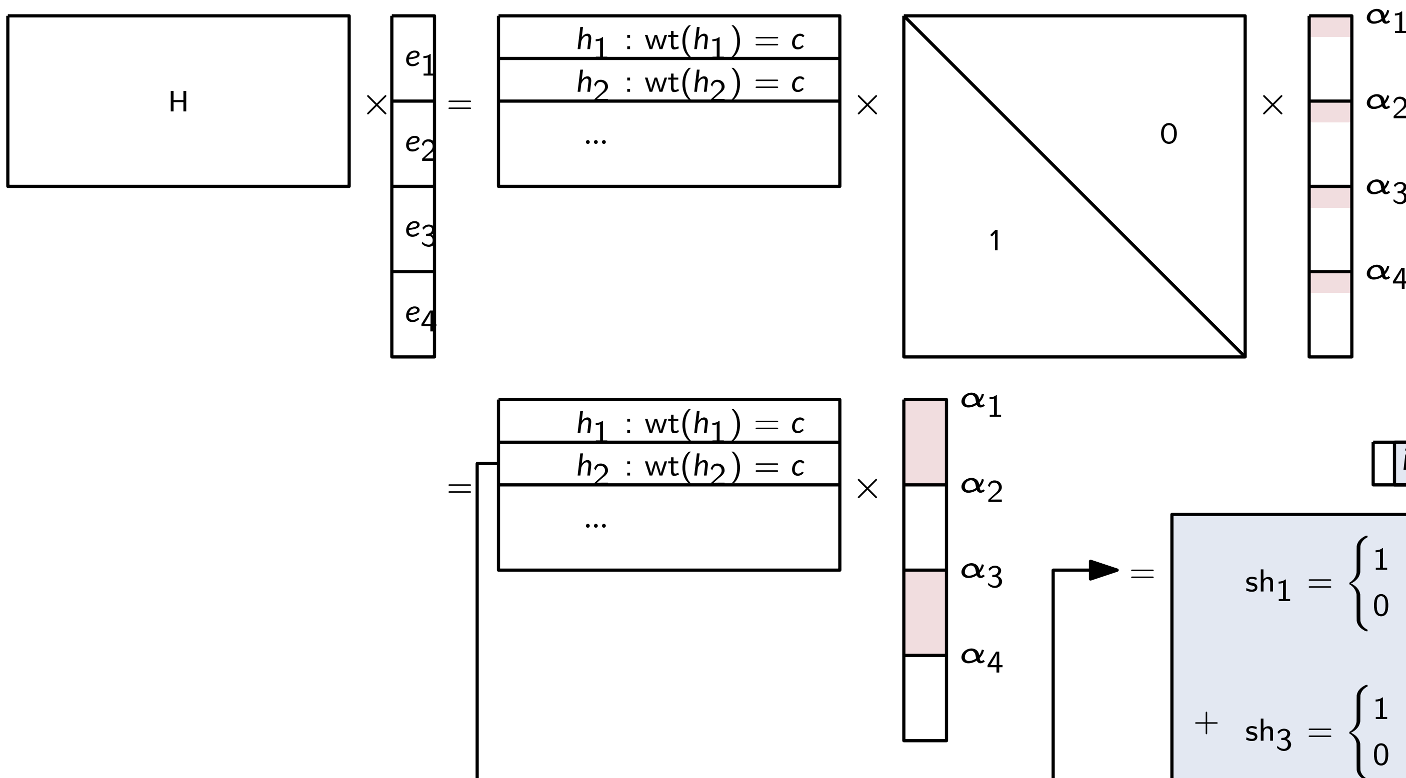
$$+ \ \mathrm{sh}_3 = \begin{cases} 1 & i_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_4 = \begin{cases} 1 & i_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases}$$

Garbling Pseudorandom Correlation Function

- PCF relies on special LPN flavors (i.e. VD-LPN, EA-LPN)
- This paper uses EA-LPN (Crypto'22)
- (i-th row) $h_i = \mathrm{HW}_c \cdot \triangle$



For the easy of demonstration
we assume the first entry in each block is one



$$\mathrm{sh}_1 = \begin{cases} 1 & i_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_2 = \begin{cases} 1 & i_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases}$$

$$+ \mathrm{sh}_3 = \begin{cases} 1 & i_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_4 = \begin{cases} 1 & i_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases}$$

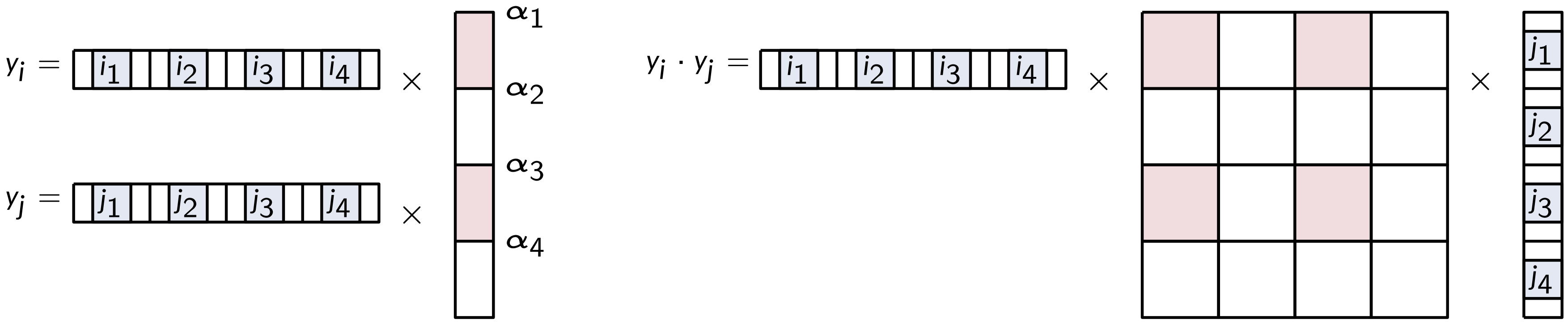
DCT for indices $\alpha_1, \alpha_2, \alpha_3, \alpha_4$

Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, ... polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = h_j^T \cdot e$
- Therefore, $y_i \cdot y_j = (\mathbf{h_i} \otimes \mathbf{h_j})^T \cdot (\mathbf{e} \otimes \mathbf{e})$

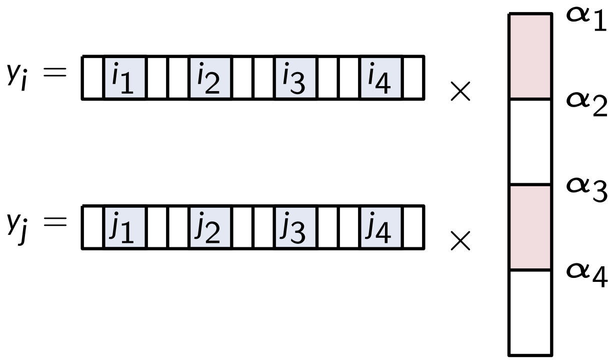
Garbling Pseudorandom Correlation Function

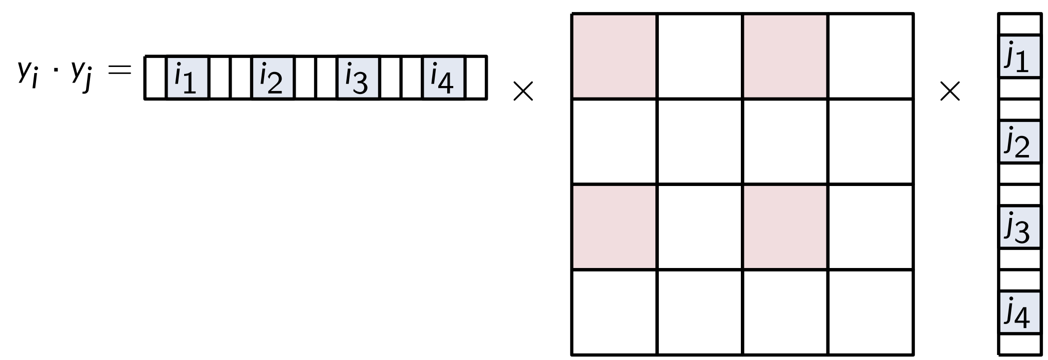
- Using FSS for decision tree, we can support degree-2, 3, ... polynomials
- Recall that with EA-LPN, we have $y_i = \underbrace{h_i^T}_{c\text{-sparse}} \cdot \underbrace{e}_{t\text{-sparse}}, y_j = h_j^T \cdot e$
- Therefore, $y_i \cdot y_j = (\underbrace{\mathbf{h_i} \otimes \mathbf{h_j}}_{c^2\text{-sparse}})^T \cdot (\underbrace{\mathbf{e} \otimes \mathbf{e}}_{t^2\text{-sparse}})$
- With FSS for decision trees, we can support constant degree polynomial evaluation over the $\mathbf{y}$ coordinates.



Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, ... polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = \underbrace{h_j^T}_{c\text{-sparse}} \cdot \underbrace{e}_{t\text{-sparse}}$
- Therefore, $y_i \cdot y_j = (\underbrace{\mathbf{h_i} \otimes \mathbf{h_j}}_{c^2\text{-sparse}})^T \cdot (\underbrace{\mathbf{e} \otimes \mathbf{e}}_{t^2\text{-sparse}})$
- With FSS for decision trees, we can support constant degree polynomial evaluation over the $\mathbf{y}$ coordinates.





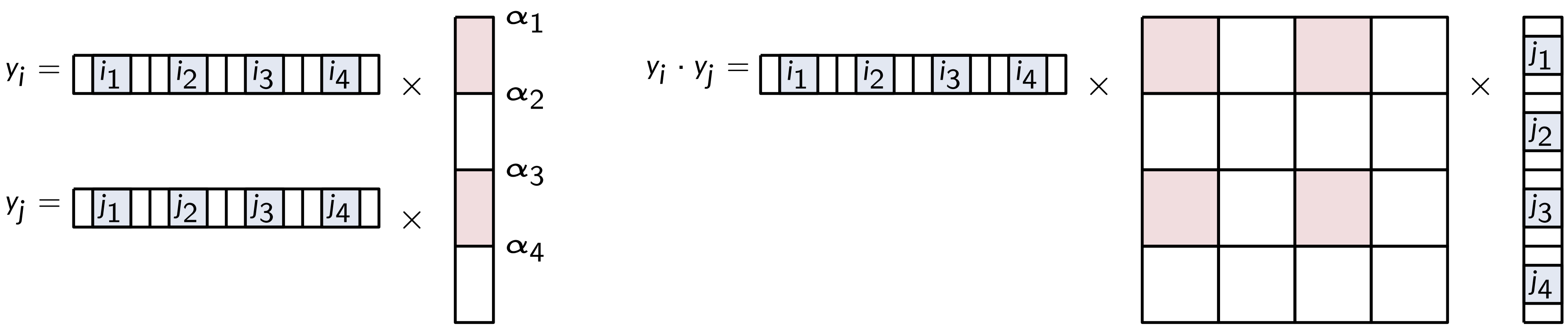
$$y_i \cdot y_j = \quad \mathsf{sh}_{1,1} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathsf{sh}_{1,2} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases}$$

$$+ \ \mathsf{sh}_{1,3} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathsf{sh}_{1,4} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases}$$

$$+ \ldots$$

Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, … polynomials
- Recall that with EA-LPN, we have $y_i = \underbrace{h_i^T}_{c\text{-sparse}} \cdot \underbrace{e}_{t\text{-sparse}}$, $y_j = h_j^T \cdot e$
- Therefore, $y_i \cdot y_j = (\underbrace{\mathbf{h_i} \otimes \mathbf{h_j}}_{c^2\text{-sparse}})^T \cdot (\underbrace{\mathbf{e} \otimes \mathbf{e}}_{t^2\text{-sparse}})$
- With FSS for decision trees, we can support constant degree polynomial evaluation over the $\mathbf{y}$ coordinates.



$$
\begin{aligned}
y_i \cdot y_j = \; & \mathrm{sh}_{1,1} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,2} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases} \\
+ \; & \mathrm{sh}_{1,3} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,4} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases} \\
+ \; & \ldots
\end{aligned}
$$

FSS for (2,3,…)-dim rectangles

FSS for Constant Dimension Rectangles

- Originates from FSS for decision trees in [BGI16]

Theorem (informal): There exists FSS for decision trees
1). with key size bounded by $3|V|(\lambda + 1)$ bits
2). assuming PRG

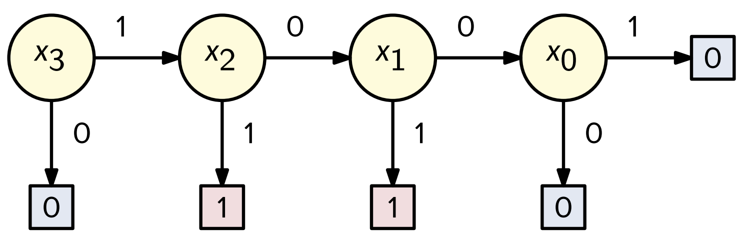
FSS for Constant Dimension Rectangles

- Originates from FSS for decision trees in [BGI16]

> Theorem (informal): There exists FSS for decision trees
> 1). with key size bounded by $3|V|(\lambda + 1)$ bits
> 2). assuming PRG

- For an n-bit string, we can make a decision tree of size $O(n)$

E.g., $\boldsymbol{\alpha} = 1001, f(x) = \begin{cases} 1 & x > \boldsymbol{\alpha} \\ 0 & x \leq \boldsymbol{\alpha} \end{cases}$



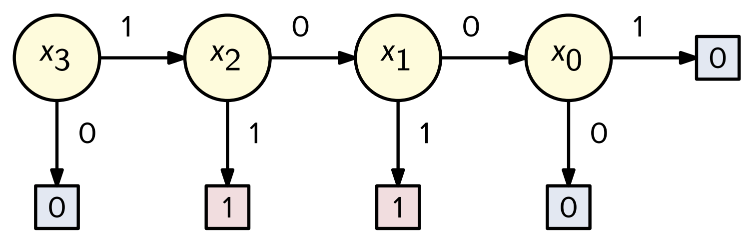
FSS for Constant Dimension Rectangles

- Originates from FSS for decision trees in [BGI16]

> Theorem (informal): There exists FSS for decision trees
> 1). with key size bounded by $3|V|(\lambda+1)$ bits
> 2). assuming PRG

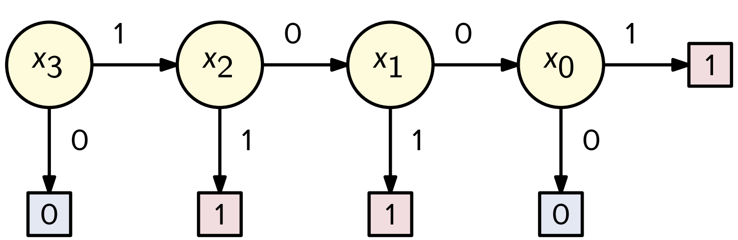
- For an n-bit string, we can make a decision tree of size $O(n)$

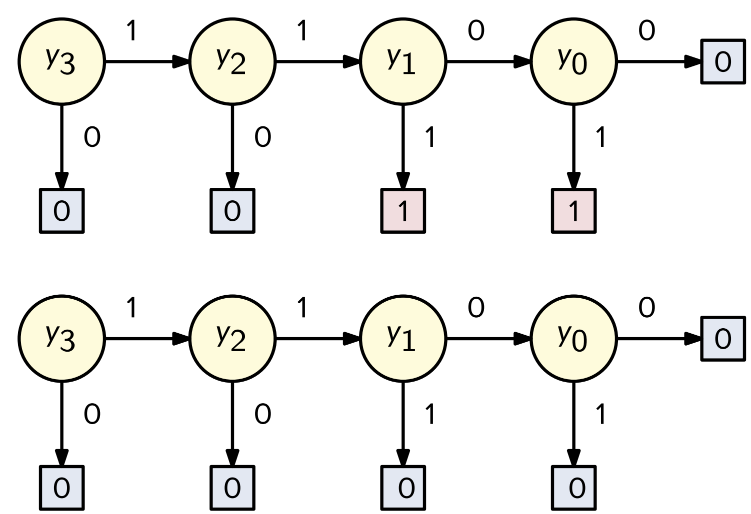
E.g., $\alpha = 1001, f(x) = \begin{cases} 1 & x > \alpha \\ 0 & x \leq \alpha \end{cases}$



- For a pair of n-bit strings, we can use "tensor" operation to cascade trees

E.g., $\alpha = 1000, \beta = 1100, f(x,y) = \begin{cases} 1 & x > \alpha \wedge y > \beta \\ 0 & \text{otherwise} \end{cases}$





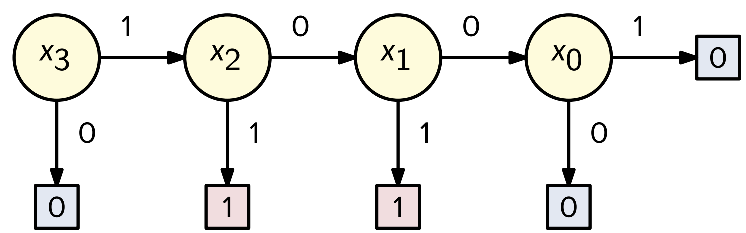
FSS for Constant Dimension Rectangles

- Originates from FSS for decision trees in [BGI16]

> Theorem (informal): There exists FSS for decision trees
> 1). with key size bounded by $3|V|(\lambda+1)$ bits
> 2). assuming PRG

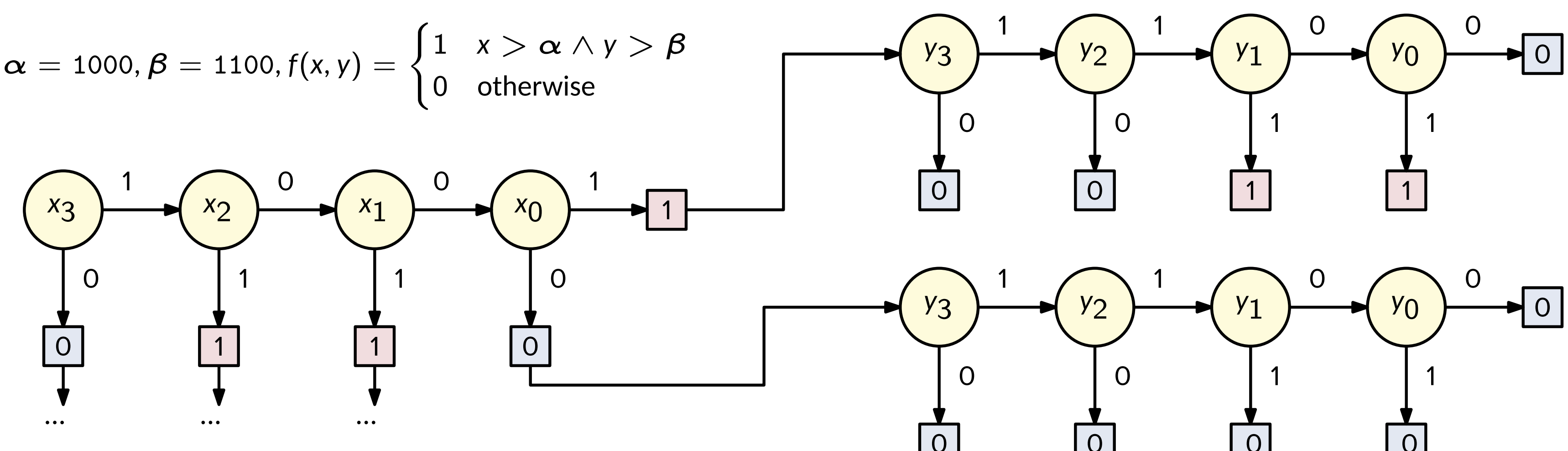
- For an n-bit string, we can make a decision tree of size $O(n)$

E.g., $\alpha = 1001, f(x) = \begin{cases} 1 & x > \alpha \\ 0 & x \leq \alpha \end{cases}$



- For a pair of n-bit strings, we can use “tensor” operation to cascade trees

E.g., $\alpha = 1000, \beta = 1100, f(x,y) = \begin{cases} 1 & x > \alpha \wedge y > \beta \\ 0 & \text{otherwise} \end{cases}$

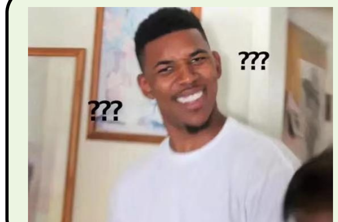


Input Encoding

- $\mathsf{GI.Setup}(\Delta_A, \Delta_B) \mapsto (\mathsf{pk}, \mathsf{gik}_A, \mathsf{gik}_B)$
- $\mathsf{GI.PKEncode}(\mathsf{pk}, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SKEncode}(\mathsf{gik}_\sigma, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SelectLabel}(\sigma, \mathsf{gik}_\sigma, \langle x \rangle_\sigma, \mathsf{L}_0^\sigma, \langle r \rangle_\sigma) \mapsto (\langle \mathsf{L}_x^A \rangle_\sigma, \langle \mathsf{L}_x^B \rangle_\sigma)$

Input Encoding

- $\mathsf{GI.Setup}(\Delta_A, \Delta_B) \mapsto (\mathsf{pk}, \mathsf{gik}_A, \mathsf{gik}_B)$
- $\mathsf{GI.PKEncode}(\mathsf{pk}, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SKEncode}(\mathsf{gik}_\sigma, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SelectLabel}(\sigma, \mathsf{gik}_\sigma, \langle x \rangle_\sigma, \mathsf{L}_0^\sigma, \langle r \rangle_\sigma) \mapsto (\langle \mathsf{L}_x^A \rangle_\sigma, \langle \mathsf{L}_x^B \rangle_\sigma)$



Why L_x
Rather than $\mathsf{L}_{x \oplus r}$

Input Encoding

- $\mathsf{GI.Setup}(\Delta_A, \Delta_B) \mapsto (\mathsf{pk}, \mathsf{gik}_A, \mathsf{gik}_B)$
- $\mathsf{GI.PKEncode}(\mathsf{pk}, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SKEncode}(\mathsf{gik}_\sigma, x) \mapsto (\langle x \rangle_A, \langle x \rangle_B)$
- $\mathsf{GI.SelectLabel}(\sigma, \mathsf{gik}_\sigma, \langle x \rangle_\sigma, \mathsf{L}_0^\sigma, \langle r \rangle_\sigma) \mapsto (\langle \mathsf{L}_x^A \rangle_\sigma, \langle \mathsf{L}_x^B \rangle_\sigma)$

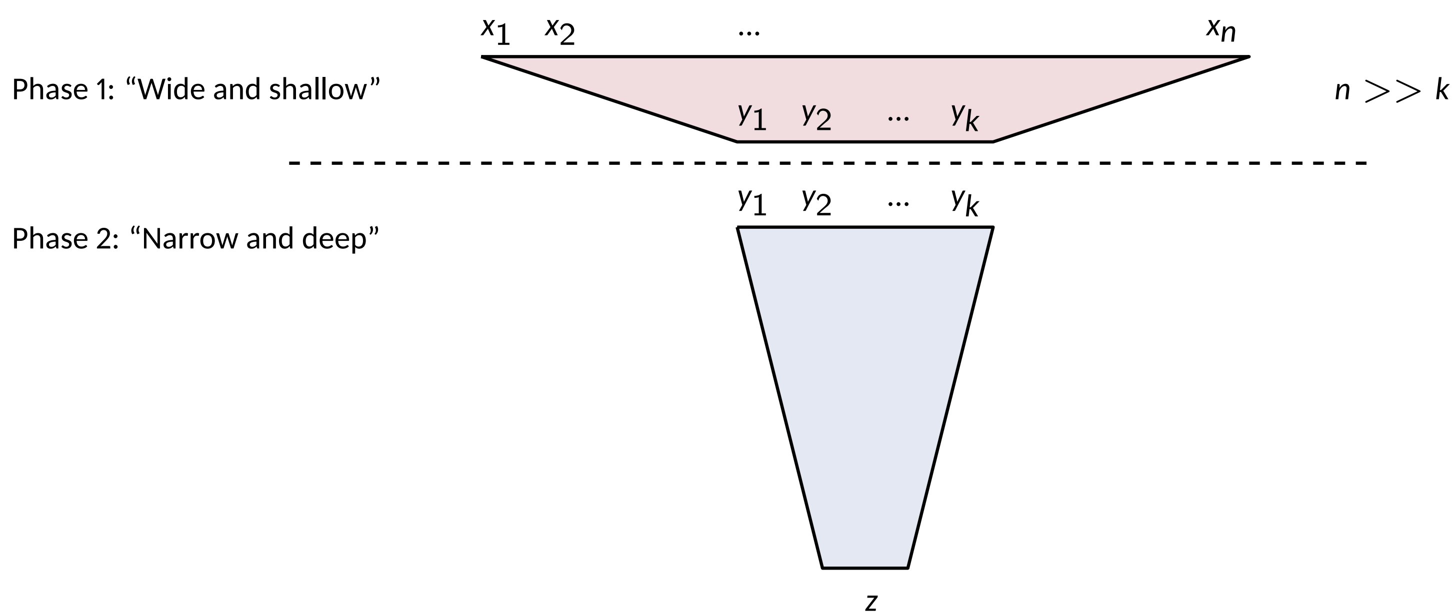
- Construction 1: During Setup, run $\langle \Delta_A, \Delta_B \rangle \leftarrow \mathsf{HSS.Share}(\Delta_A, \Delta_B)$.
- During PKEncode, run $\langle x \rangle \leftarrow \mathsf{HSS.Share}(x)$
- During SelectLabel, use $\mathsf{HSS.Mult}(\langle x \rangle, \langle \Delta_\sigma \rangle)$ to get shares of $x \cdot \Delta_\sigma$

- Construction 2: During Setup, run $(k_A, k_B) \leftarrow \mathsf{PCF.KeyGen}(\Delta_A, \Delta_B)$.
- $\mathsf{PCF.Eval}(k_\sigma, \mathsf{idx}) \mapsto (r_\sigma, \langle r_A \Delta_A \rangle_\sigma, \langle r_A \Delta_B \rangle_\sigma, \langle r_B \Delta_A \rangle_\sigma, \langle r_B \Delta_B \rangle_\sigma)$
- During SKEncode, Party-σ runs $\mathsf{PCF.Eval}(k_\sigma, \mathsf{idx})$ to get r_σ and sets $\langle x \rangle_A = \langle x \rangle_B = r_\sigma \oplus x$
- During SelectLabel, set

$$\langle \mathsf{L}_x^{A/B} \rangle = \langle \mathsf{L}_0^{A/B} \rangle \oplus (x \oplus r_\sigma) \cdot \langle \Delta_{A/B} \rangle \oplus \langle r \Delta_{A/B} \rangle$$

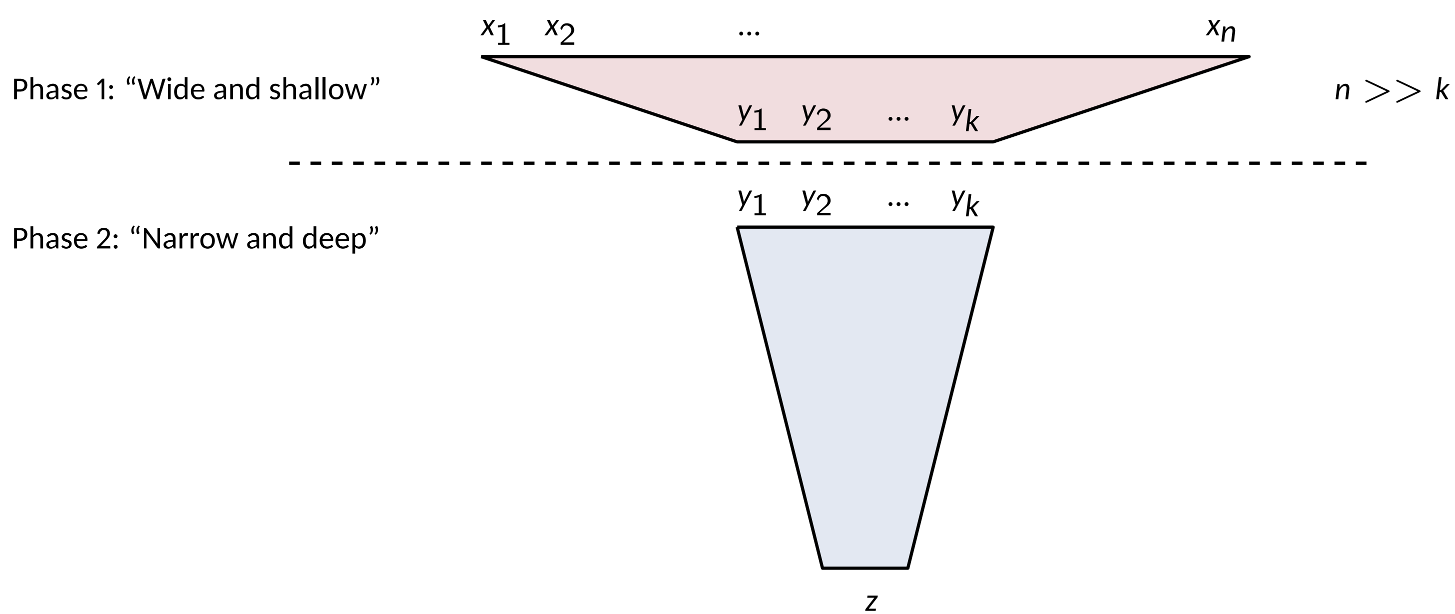
Application #1: HSS for T-shaped Circuit

- Idea: use HSS to specify input
- Move the large-depth evaluation work to reconstruction



Application #1: HSS for T-shaped Circuit

- Idea: use HSS to specify input
- Move the large-depth evaluation work to reconstruction



Evaluation: given GC-PCG/PCF keys and $\langle x_i \rangle$

Step 1: Use an HSS scheme to convert $\langle x_i \rangle$ into $\langle y_j \rangle$

Step 2: Use HSS-based Input Encoding scheme to convert $\langle y_j \rangle$ into $\langle \mathsf{L}^{A/B}_{j,y_j} \rangle$

Step 3: Use GC-PCG/PCF to generate shares of $\hat{C}$
With $\langle \hat{C} \rangle$ and $\mathsf{L}^{A/B}_{j,y_j}$, one can recover z

- HSS Keys = Phase 1 HSS Keys + CG-PCG/PCF Keys
- We assume GI.Encode can be done with input shares under Phase 1 HSS

4.1 Definitions

We base our definitions of homomorphic secret sharing (HSS) on those given by Boyle *et al.* [BKS19].

Definition 4.1 (Homomorphic Secret Sharing). *A (2-party, public-key)* Homomorphic Secret Sharing (HSS) *scheme for a class of programs $\mathcal{P}$ over a ring R with input space $\mathcal{I} \subseteq R$ consists of PPT algorithms* $(\mathsf{HSS.Setup}, \mathsf{HSS.Input}, \mathsf{HSS.Eval})$ *with the following syntax:*

- $\mathsf{HSS.Setup}(1^\lambda) \rightarrow (\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1))$: *Given a security parameter 1^λ, the setup algorithm outputs a public key* pk *and a pair of evaluation keys* $(\mathsf{ek}_0, \mathsf{ek}_1)$.
- $\mathsf{HSS.Input}(\mathsf{pk}, x) \rightarrow (\mathsf{I}_0, \mathsf{I}_1)$: *Given public key* pk *and private input value $x \in \mathcal{I}$, the input algorithm outputs input information* $(\mathsf{I}_0, \mathsf{I}_1)$.
- $\mathsf{HSS.Eval}(\sigma, \mathsf{ek}_\sigma, (\mathsf{I}_\sigma^{(1)}, \ldots, \mathsf{I}_\sigma^{(\rho)}), P) \rightarrow y_\sigma$: *On input a party index $\sigma \in \{0,1\}$, evaluation key* ek_σ, *vector of ρ input values and a program $P \in \mathcal{P}$ with ρ input values, the homomorphic evaluation algorithm outputs $y_\sigma \in R$, which is party σ's share of an output $y \in R$.*

Note that, in the constructions we consider, we have $\mathsf{I}_0 = \mathsf{I}_1$. *We say that* $(\mathsf{HSS.Setup}, \mathsf{HSS.Input}, \mathsf{HSS.Eval})$ *is a* homomorphic secret sharing scheme *for the class of programs $\mathcal{P}$ if the following conditions hold:*

Orlandi, C., Scholl, P., Yakoubov, S. *The Rise of Paillier: Homomorphic Secret Sharing and Public-Key Silent OT.* EUROCRYPT 2021

Application #2: ZK on Secret-Shared Data

- Prover encodes the input using $\mathsf{GI.PKEncode}$
- Verifiers generate proof using distributed garbling
- Verification process reconstructs and gets the verdict

Example Construction in Appendix E.3

$\mathsf{zkPSD.Audit}(\mathsf{crs}, \mathsf{ak}_\sigma, x_\sigma, \pi_\sigma)$:

$\textbf{parse}\ \mathsf{crs} = \mathsf{pk},\ \mathsf{ak}_\sigma = (\sigma, \mathsf{k}^\sigma, \mathsf{gik}^\sigma),\ x_\sigma = \langle\!\langle x \rangle\!\rangle^\sigma,\ \pi_\sigma = \langle\!\langle w \rangle\!\rangle^\sigma$

$\left(\langle \widehat{\mathcal{C}} \rangle^\sigma, \vec{L}_0^\sigma, \langle \vec{\mathsf{r}} \rangle^\sigma\right) \leftarrow \mathsf{GPCF.Eval}(\sigma, \mathsf{k}^\sigma, \mathcal{C})$

$\langle \widehat{X} \rangle^\sigma \leftarrow \mathsf{GI.SelectLabels}(\sigma, \mathsf{gik}^\sigma, (\langle\!\langle x \rangle\!\rangle^\sigma, \langle\!\langle w \rangle\!\rangle^\sigma), \vec{L}_0^\sigma, \langle \vec{\mathsf{r}} \rangle^\sigma)$ ▷ See Construction 3

$\textbf{return}\ \tau_\sigma := (\langle \widehat{\mathcal{C}} \rangle^\sigma, \langle \widehat{X} \rangle^\sigma)$

$\mathsf{GI.SelectLabels}(\sigma, \mathsf{gik}^\sigma, (\langle\!\langle x_A \rangle\!\rangle^\sigma, \langle\!\langle x_B \rangle\!\rangle^\sigma), \vec{L}_0^\sigma, \langle \vec{\mathsf{r}} \rangle^\sigma)$:

$\textbf{parse}\ \mathsf{gik}^\sigma := (\mathsf{ek}^\sigma, \langle\!\langle \Delta^A \rangle\!\rangle^\sigma, \langle\!\langle \Delta^B \rangle\!\rangle^\sigma)$

$\textbf{parse}\ \vec{L}_0^\sigma := (\vec{L}_{1,0}^\sigma, \ldots, L_{s,0}^\sigma)\ \textbf{and}\ \langle \vec{\mathsf{r}} \rangle^\sigma = (\langle \mathsf{r}_1 \rangle^\sigma, \ldots, \langle \mathsf{r}_s \rangle^\sigma)$

$(\langle\!\langle x_1 \rangle\!\rangle^\sigma, \ldots, \langle\!\langle x_s \rangle\!\rangle^\sigma) := (\langle\!\langle x_{A,1} \rangle\!\rangle^\sigma, \ldots, \langle\!\langle x_{A,s_A} \rangle\!\rangle^\sigma, \langle\!\langle x_{B,1} \rangle\!\rangle^\sigma, \ldots, \langle\!\langle x_{B,s_B} \rangle\!\rangle^\sigma)$

$\textbf{foreach}\ i \in [s]$:

$\quad \langle x_i \rangle^\sigma \leftarrow \mathsf{Convert}^\sigma(\mathsf{ek}^\sigma, \langle\!\langle x_i \rangle\!\rangle^\sigma)$

$\quad \langle y_i^A \rangle^\sigma \leftarrow \mathsf{Mult}^\sigma(\mathsf{ek}^\sigma, \langle\!\langle \Delta^A \rangle\!\rangle^\sigma, \langle\!\langle x_i \rangle\!\rangle^\sigma)$

$\quad \langle y_i^B \rangle^\sigma \leftarrow \mathsf{Mult}^\sigma(\mathsf{ek}^\sigma, \langle\!\langle \Delta^B \rangle\!\rangle^\sigma, \langle\!\langle x_i \rangle\!\rangle^\sigma)$

Contradiction!

Definition 23 (Soundness: Malicious provers, honest verifiers). *A zkPSD proof system is sound against a malicious prover and semi-honest verifiers, if for all efficient adversaries $\mathcal{A}$, and all $x \in \{0,1\}^*$ such that $x \notin \mathcal{L}$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l|c} (\mathsf{crs}, (\mathsf{ak}_A, \mathsf{ak}_B)) \leftarrow \mathsf{Setup}(1^\lambda) & x_A + x_B = x \\ (x_A, x_B, \pi_A, \pi_B) \leftarrow \mathcal{A}(\mathsf{crs}, x) & \wedge \\ \tau_A \leftarrow \mathsf{Audit}(\mathsf{crs}, \mathsf{ak}_A, x_A, \pi_A) & \mathsf{Verify}(\tau_A, \tau_B) = 1 \\ \tau_B \leftarrow \mathsf{Audit}(\mathsf{crs}, \mathsf{ak}_B, x_B, \pi_B) & \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of $\mathcal{A}$ and Setup.

Application #3: Reusable Non-Interactive Secure Computation

- Message 1: Encode the input
- Message 2: Recover the GC and Input labels to derive the output

Setup

$k^A_{\mathrm{GI}}, k^A_{\mathrm{GPCF}}$

$k^B_{\mathrm{GI}}, k^B_{\mathrm{GPCF}}$

Phase 1: Input Encoding

$\mathrm{GI.Encode}(x_A, k^A_{\mathrm{GI}}) \mapsto (\langle x_A \rangle^A, \langle x_B \rangle^B)$

$\mathrm{GI.Encode}(x_B, k^B_{\mathrm{GI}}) \mapsto (\langle x_B \rangle^A, \langle x_B \rangle^B)$

$\langle x_A \rangle^B \longrightarrow$

$\longleftarrow \langle x_B \rangle^A$

Phase 2: Circuit Evaluation

$\mathrm{GI.SelectLabel}(k^A_{\mathrm{GI}}\ \langle x_A \rangle^A\ \langle x_B \rangle^A) \mapsto (\langle L \rangle^A)$

$\mathrm{GPCF.Eval}(k^A_{\mathrm{GPCF}}, \mathcal{C}) \mapsto (\langle G_{\mathcal{C}} \rangle^A)$

Bob gets output $\longrightarrow$

$\longleftarrow$ Alice gets output

EA-LPN is Possibly too Agreesive

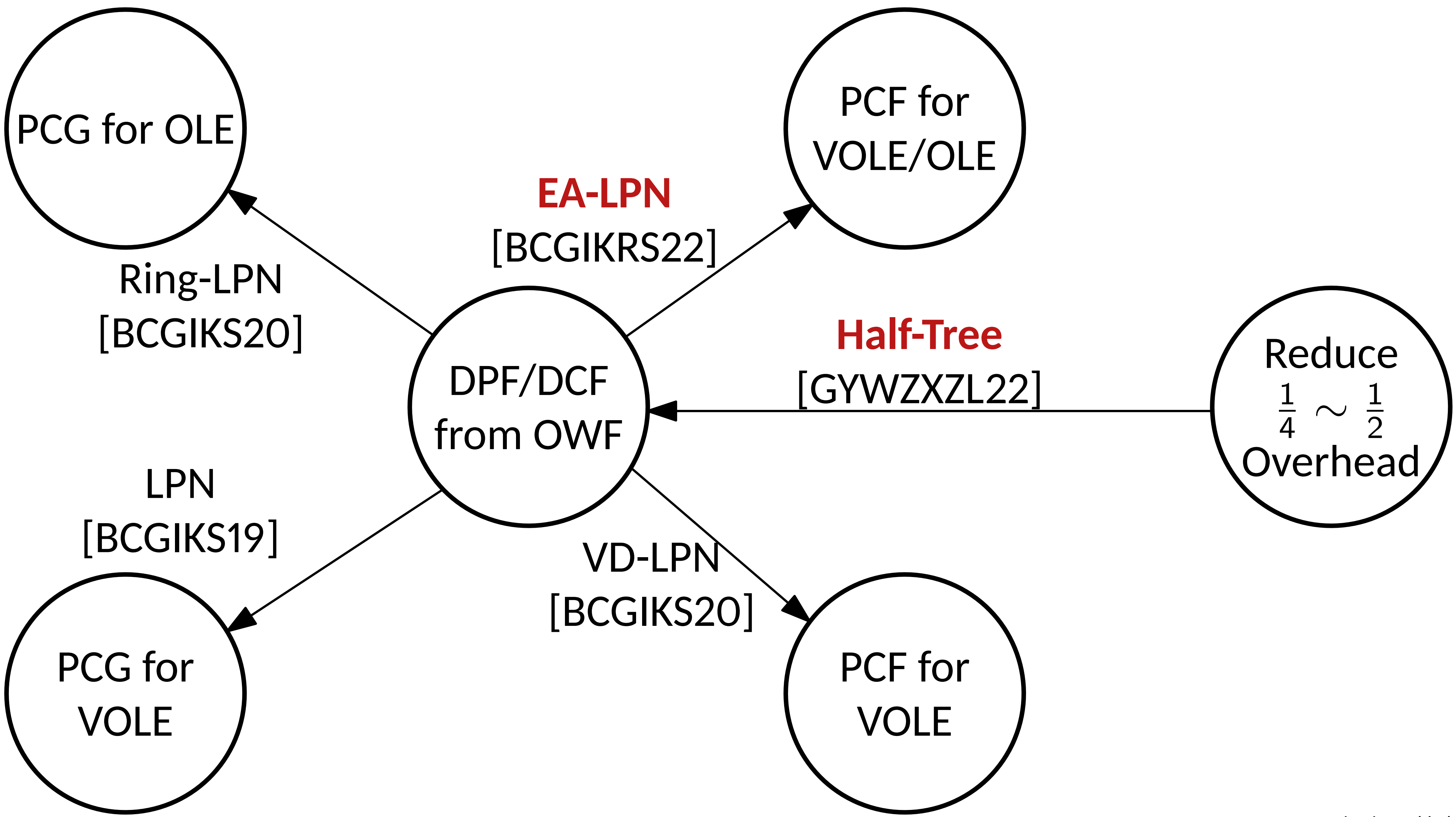
of the resulting construction (see [33, Section 5.4 of the full version]).

Communication costs. The communication cost (in bits) is defined in terms of n, t, and c, and is given by $4\lambda \cdot (\log(5n/t) \cdot t)^2$ when using the state-of-the-art FSS schemes for 2-dimensional interval functions [24], where we will fix $\lambda = 128$ for the purposes of our estimates. Since we fix $n = 2^{35}$, we must carefully choose the parameters c and t. Boyle et al. [33] give both provable and heuristic parameter regimes. In the provable security regime, we can set $t = 85$ and $c = 3\ln(5n)$. This results in a concrete key size of roughly 800 MB.

However, if we opt for heuristic parameters instead (which opti[illegible]e for concrete *computation* costs), we can we can set $t = 664$ and $c = 11$. These parameter choices are validated through simulation on the minimum distance of the code [33, 90]. However, we use a more conservative choice for c compared to Boyle et al. [33], in light of an improved minimum distance algorithm Raghuraman et al. [90], affecting the security of the original heuristic parameters proposed in Boyle et al. [33].[4] With these, *compute-optimized* heuristic parameter choices, the key size grows to 40 GB, but significantly improves the concrete computation time, as we will explain next. We stress that these parameters should be taken with care and might change in light of future cryptanalysis.

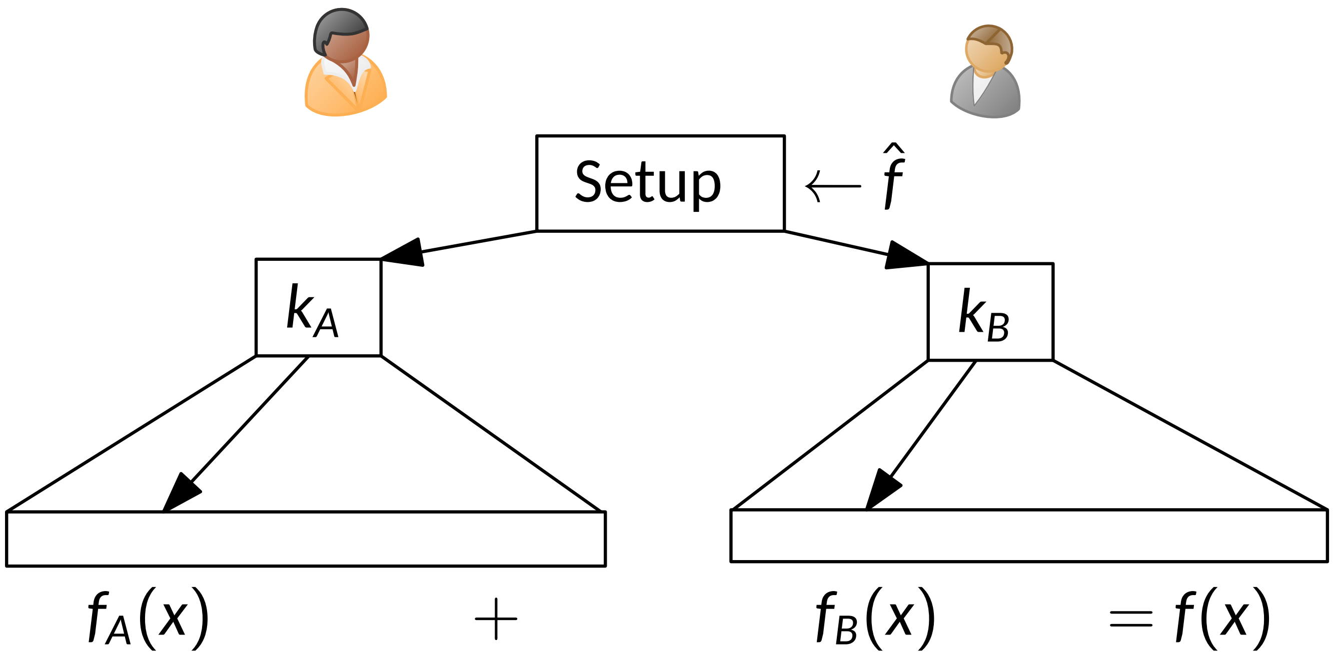
Introduction

- PCG/PCF paradigm = FSS + LPN
- The main contribution is a new LPN variant and FSS optimization



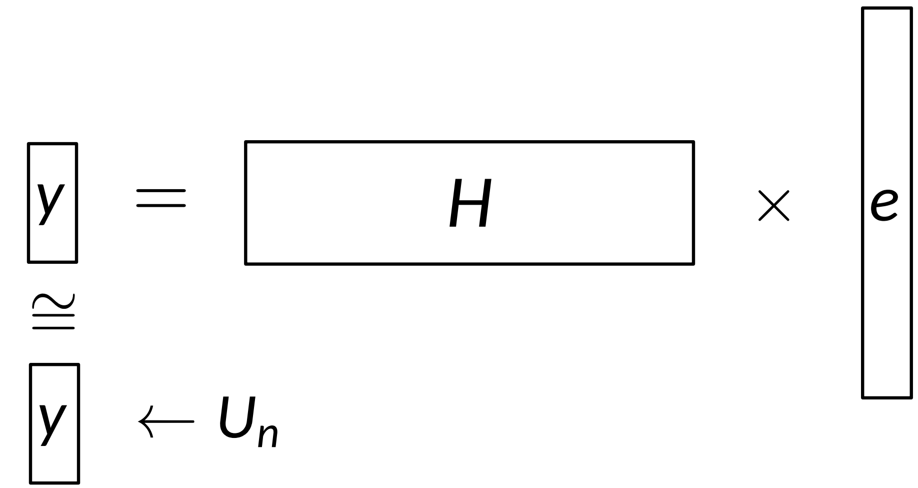
Preliminaries on PCG/PCF

Function Secret Sharing



- Succinctness: $|k_A|, |k_B| \ll 2^{|x|}$
- Efficient FSS exists for point/comparison functions

dual-LPN



- View e as seed, H is a linear PRG
- PCG idea: generate sparse correlations as seed and expand them using dual-LPN

Example: PCG for VOLE

- KeyGen:

Step 1: $e \leftarrow \mathrm{Ber}^N$ $\quad e = \boxed{\begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_\ell \end{array}}$

$$\begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_\ell \end{array}$$

Step 2: $(k_0^1, k_1^1) \leftarrow FSS.KeyGen(\alpha_1, \beta_1 \cdot \Delta)$

$\cdots$

$(k_0^\ell, k_1^\ell) \leftarrow FSS.KeyGen(\alpha_\ell, \beta_\ell \cdot \Delta)$

 $key_0 := \{k_0^1, ..., k_0^\ell\}, \Delta$  $key_1 := \{k_1^1, ..., k_1^\ell\}, e$

- Expand:

 $w := H \cdot (FullEval(k_0^1) + ... + FullEval(k_0^\ell))$

 $v := H \cdot (FullEval(k_1^1) + ... + FullEval(k_1^\ell)), u := H \cdot e$

$$w + v = \begin{array}{c} \boxed{\quad \beta_1 \cdot \Delta \qquad\qquad\qquad\qquad\qquad} \\ + ... + \\ \boxed{\qquad\qquad\qquad\qquad\qquad \beta_\ell \cdot \Delta} \end{array} = H \cdot e \cdot \Delta = u \cdot \Delta$$

From PCG to PCF

- Analogous to the extension from PRG to PRF
- Main problem: N is super-polynomial
- If H has no structure, then evaluating the inner-product is infeasible

$$N > n$$

$$n = \kappa^{\omega(1)} \quad y = H \times e$$

Expand-Accumulate LPN

- Structure:

$$\boxed{\quad H \quad} = \underset{c_1\, c_2 \quad \cdots \quad c_N}{\boxed{\quad Ber \quad}} \times \boxed{\begin{matrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 & 1 \end{matrix}}$$

  - $h_N = c_N$
  - $h_{N-1} = h_N + c_{N-1}$
  - $h_1 = h_2 + c_1$

- Proof strategy: Prove that one row in *H* is high-weight whp.

- Intuition1: Bernoulli noise accumulates to uniform due to pilling up lemma
- Intuition2: Columns of *H* corresponds to random walk

Tools: Random Walk and Markov Chain

Theorem 3.6 (Expander Hoeffding Bound) *Let $(\mathcal{V}, P)$ denote a finite, irreducible and reversible Markov chain with stationary distribution $\vec{\pi}$ and second largest eigenvalue λ. Let $f : \mathcal{V} \to [0,1]$ with $\mu = \mathbb{E}_{V\sim\vec{\pi}}[f(V)]$. For any integer $N \geq 1$, consider the random variable $S_N = \sum_{i=1}^{N} f(V_i)$, where V_0 is sampled uniformly at random from V and then $V_1, \ldots, V_N$ is a random walk starting at V_0.*

Then, for $\lambda_0 = \max(0, \lambda)$ and any $\varepsilon > 0$ with $\mu + \varepsilon < 1$, the following bound holds:

$$\Pr\left[S_N \geq N(\mu + \varepsilon) \right] \leq \exp\left(-2\frac{1-\lambda_0}{1+\lambda_0} N\varepsilon^2 \right) .$$

- Applying Markov bound

Corollary 3.7 *Let $(\mathcal{V}, P)$ denote a finite, irreducible and reversible Markov chain with $\mathcal{V} = \{v_0, v_1\}$, stationary distribution $\vec{\pi} = (1/2, 1/2)$ and second largest eigenvalue λ. Let $f : \mathcal{V} \to [0,1]$ with $1/2 = \mathbb{E}_{V\sim\vec{\pi}}[f(V)]$. For any integer $N \geq 1$, consider the random variable $\tilde{S}_N = \sum_{i=1}^{N} f(V_i)$, where $V_0 = v_0$ with probability 1 and then $V_1, \ldots, V_N$ is a random walk starting at v_0.*

Then, for $\lambda_0 = \max(0, \lambda)$ and any $\varepsilon > 0$ with $1/2 + \varepsilon < 1$, the following bound holds:

$$\Pr\left[\tilde{S}_N \geq N(1/2 + \varepsilon) \right] \leq 2\exp\left(-2\frac{1-\lambda_0}{1+\lambda_0} N\varepsilon^2 \right) .$$

Security of EA-LPN against Linear Attack

Theorem 3.10 *Let $n, N \in \mathbb{N}$ with $n \leq N$ and put $R = \frac{n}{N}$, which we assume to be a constant. Let $C > 0$ and set $p = \frac{C \ln N}{N} \in (0, 1/2)$. Fix $\delta \in (0, 1/2)$ and put $\beta = 1/2 - \delta$. Assume the following relation holds:*

$$R < \min\left\{\frac{2}{\ln 2} \cdot \frac{1 - e^{-1}}{1 + e^{-1}} \cdot \boxed{\beta^2}, \frac{2}{e}\right\} \tag{2}$$

Then, assuming N is sufficiently large we have

$$\begin{aligned}\Pr\left[\mathsf{d}(H) \geq \delta N \mid H \xleftarrow{\$} \mathsf{EAGen}(n, N, p)\right] &\geq 1 - 2\sum_{r=1}^{n}\binom{n}{r}\exp\left(-2\frac{1-\xi_r}{1+\xi_r}N\beta^2\right) \\ &\geq 1 - 2RN^{-2\beta^2 C + 2}.\end{aligned} \tag{3}$$

- (ϵ, η)-security: $\Pr[\mathsf{d}(H) \geq d] > \eta$ and $\max_{|v| \geq d} \mathsf{bias}_v(\chi^N) \leq \epsilon$
- $\mathsf{d}(H) \geq \delta N \rightarrow \mathsf{bias} \leq \frac{1}{2} \cdot (1 - 2 \cdot \frac{t}{N})^{\delta N} \approx \frac{1}{2} \cdot 2^{-2t\delta}$
- For $C = O(1)$, $\eta = 1 - \frac{1}{\mathsf{poly}}$; for $C = \log(N)$, $\eta = 1 - \mathsf{negl}$

Proving Theorem 3.10 Using Random Walk

- Differentiate between different hamming weight of x

$$\boxed{x^T} \times \boxed{H} = \boxed{x^T} \times \boxed{Ber(p)} \times \boxed{\begin{matrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 & 1 \end{matrix}}$$

$$= \boxed{Ber'(p')} \times A$$

- Pilling-up lemma: $2 \cdot Bias' = (2 \cdot Bias)^{|x|} \iff \xi_r = \xi^r$ s.t. $|x| = r$
- Applying the Hoeffding bound:

$$\Pr[\mathrm{wt}(Ber') \le (\tfrac{1}{2} - \beta) \cdot N] \le 2 \cdot \exp(-2 \cdot \tfrac{1-2\cdot Bias'}{1+2\cdot Bias'} \cdot N \cdot \beta^2)$$

- This gives the first inequality in Theorem 3.10

$$\Pr\left[\mathsf{d}(H) \ge \delta N \mid H \xleftarrow{\$} \mathsf{EAGen}(n, N, p)\right] \ge 1 - 2\sum_{r=1}^{n} \binom{n}{r} \exp\left(-2\frac{1-\xi_r}{1+\xi_r} N \beta^2\right)$$

Bounding the Failure Probability

- $r = 1$

$$\binom{n}{1}\exp(-2\cdot\frac{1-\xi}{1+\xi}\cdot N\cdot\beta^2) \le RN\cdot\exp(-2pN\beta^2)$$
$$= RN\cdot\exp(-2\frac{C\ln N}{N}N\beta^2)$$
$$\le N^{-2C\beta^2+1}$$

- $2 \le r \le \frac{N}{2C\ln N}$: Equivalent to prove

$$\ln\left(\binom{n}{r}\exp(-2\cdot\tfrac{1-\xi_r}{1+\xi_r}N\beta^2)\right) = -\Omega(\log N)$$
$$-1\cdot\ln\left(\binom{n}{r}\exp(-2\cdot\frac{1-\xi_r}{1+\xi_r}N\beta^2)\right) = 2\cdot\frac{1-\xi_r}{1+\xi_r}N\beta^2 - \ln\binom{n}{r}$$
$$\ge (1-\xi_r)N\beta^2 - r\ln(\frac{eRN}{r}) \ge \ln(N^{2C\beta^2-1})$$
$$R \le \frac{e}{2}$$

Bounding Failure Probability (Continued)

- $r \geq \frac{N}{2C \ln N}$

$$\xi_r = (1 - \tfrac{2C \ln N}{N})^r \leq e^{-1}$$
$$\ln \tbinom{n}{r} \leq \ln(2^{RN}) = RN \ln(2)$$

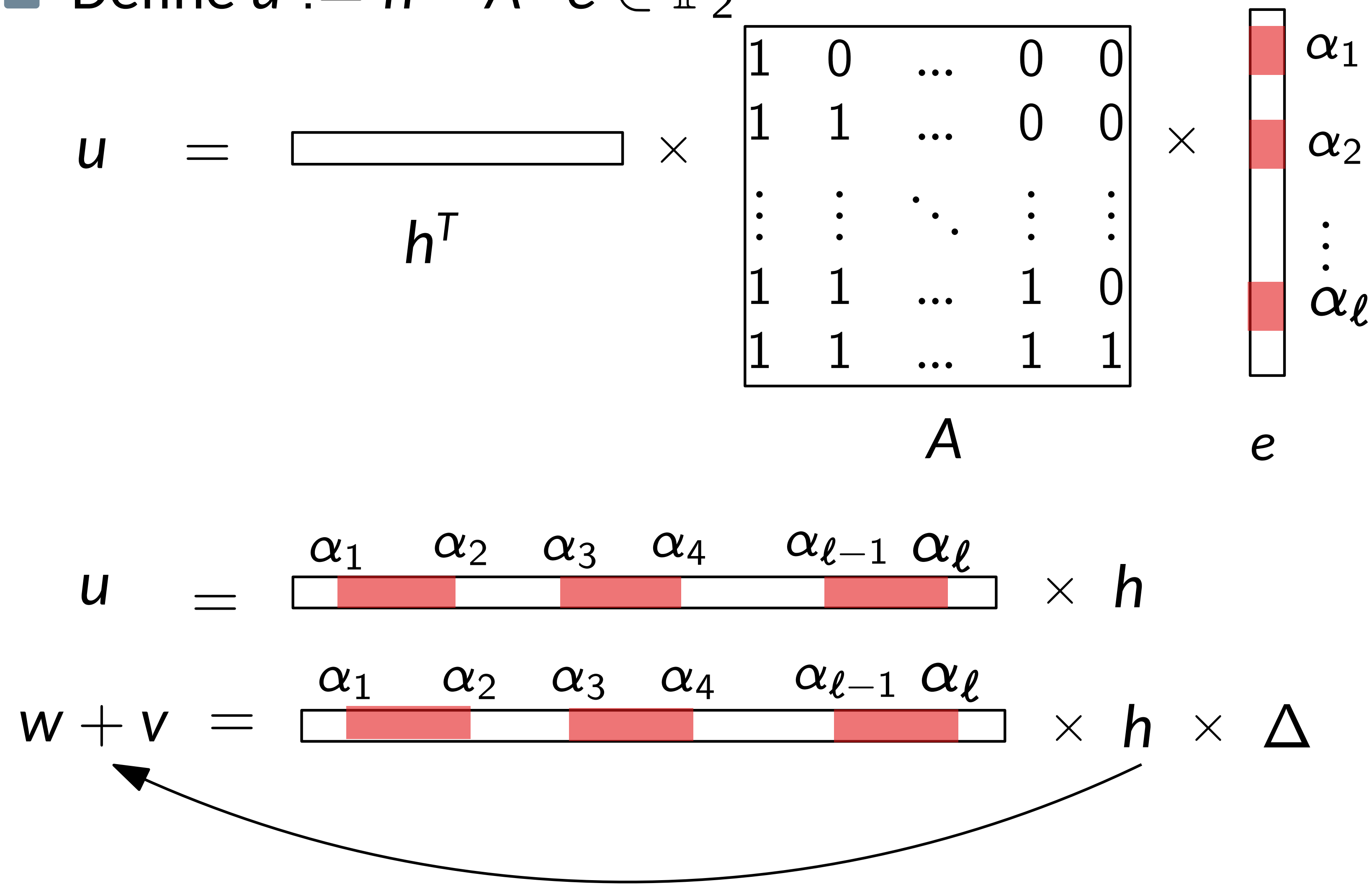
$$-1 \cdot \ln \left(\binom{n}{r} \exp(-2 \cdot \frac{1-\xi_r}{1+\xi_r} N\beta^2) \right) = 2 \cdot \frac{1-\xi_r}{1+\xi_r} N\beta^2 - \ln \binom{n}{r}$$
$$\geq 2 \cdot \frac{1-e^{-1}}{1+e^{-1}} N\beta^2 - RN \ln(2) > 0 \qquad R < 2 \cdot \tfrac{1-e^{-1}}{1+e^{-1}} \cdot \tfrac{\beta^2}{\ln(2)}$$

- Summing over $1 \leq r \leq n$:

$$\Pr[\text{Fail}] \leq 2 \cdot n \cdot N^{-2C\beta^2+1} = 2 \cdot R \cdot N^{-2C\beta^2+2}$$

Constructing PCF from EA-LPN

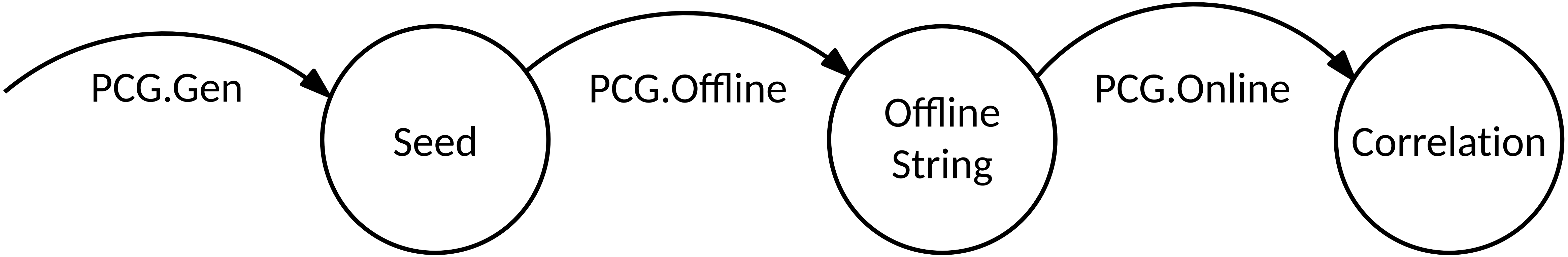
- Sample one row of H: $\mathsf{Samp}(x) \mapsto h^T$
- Define $u := h^T \cdot A \cdot e \in \mathbb{F}_2$



Run DCF on every (public) non-zero coordinate of h

Generalizations

- Offline/Online PCG



- Motivation: Utilize online idle time to mitigate offline burden

- Expand:

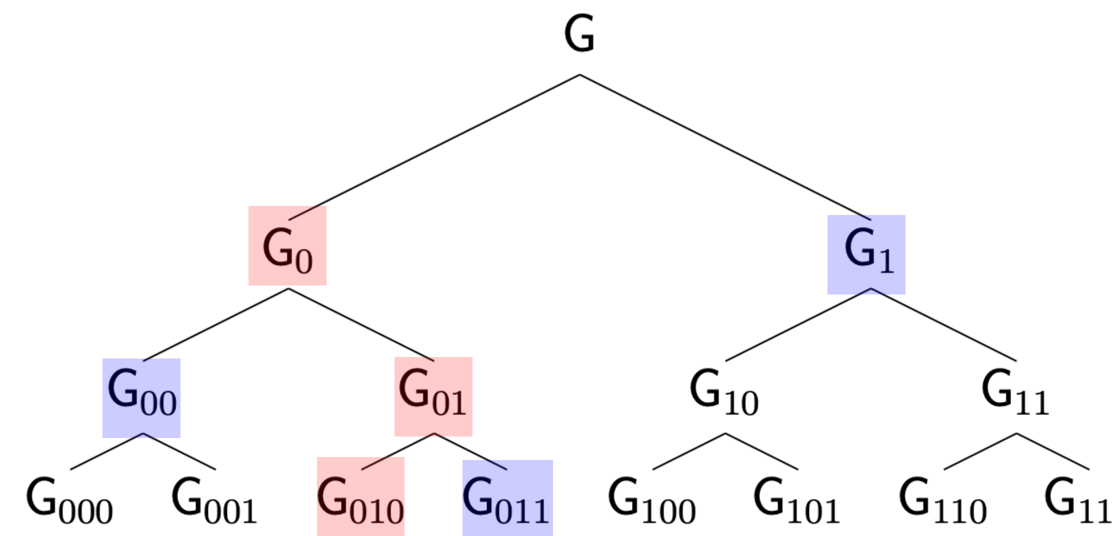
$$w := H \cdot (FullEval(k_0^1) + ... + FullEval(k_0^\ell))$$

$$v := H \cdot (FullEval(k_1^1) + ... + FullEval(k_1^\ell)), u := H \cdot e$$

$\boxed{H \leftarrow Ber} \times$ offline string

Relaxed Distributed Comparison Function

- RDCF: $f(x) = \begin{cases} 0 & x \leq \alpha \\ \beta & x > \alpha \end{cases}$ $\quad \mathsf{Expand}(k^0) \mapsto \alpha, y^0 \quad \mathsf{Expand}(k^0) \mapsto y^1$
- Example: $\alpha = 010$
- $\gamma_1 = H(G), \gamma_2 = H(G_0), \gamma_3 = H(G_{01})$
- $c_1 = \bar{\alpha}_1 \cdot \gamma_1, c_2 = \bar{\alpha}_2 \cdot \gamma_2, c_3 = \bar{\alpha}_3 \cdot \gamma_3$
- $B_i = c_1 + \ldots + c_{i-1} + \alpha_i \cdot \gamma_i + \alpha_i \cdot \beta$



$k^1 := G$

$k^0 := \langle \alpha, \{B_i\}, y = G_{010} + \sum c_i, G_1, G_{00}, G_{011} \rangle$

- Eval (x): Define $c_1^1 = \bar{x}_1 \cdot H(G), c_2^1 = \bar{x}_2 \cdot H(G_{x_1}), c_3^1 = \bar{x}_3 \cdot H(G_{x_1 x_2})$

$f^1(x) = G_{x_1 x_2 x_3} + \sum c_i^1$

$$f^0(x) = \begin{cases} y & x = \alpha \\ B_j + c_{j+1}^1 + \ldots + c_m^1 & x \neq \alpha \end{cases}$$

Offline Optimization: UPF

- Replace pseudorandomness in PPRF by unpredictability

$\underline{\mathsf{Exp}^{\mathsf{unp}}_{\mathsf{UPF},\mathcal{A}}(\lambda)}$:

$\alpha \xleftarrow{\$} \mathcal{X}_\lambda$

$k \leftarrow \mathsf{Setup}(1^\lambda)$

$k^* \leftarrow \mathsf{Puncture}(k, \alpha)$

$y \leftarrow \mathcal{A}(k^*, \alpha)$

If $y = \mathsf{Eval}(k, \alpha)$ **return** 1

Else return 0.

- Step1: a UPF that takes N ROs
- Step2: a PPRF by hashing the left leaves of UPF that takes $N/2$ ROs
- Computation saving: $2N \to 1.5N$

Summary

- Contribution 1: EA-LPN
- Contribution 2: Offline Optimization (checkout on Half-tree)

	Assump.	Corr.	Computation	Communication (bits) $P_0 \to P_1$	Communication (bits) $P_1 \to P_0$
[BCG+22]	ROM	sVOLE	m RO calls	$2t(\log \frac{m}{t} - 1)\lambda + 3t \log \lvert\mathbb{K}\rvert$	$t \log \lvert\mathbb{F}\rvert$
	Ad-hoc[1]	sVOLE	m RP calls + $0.5m$ RO calls		
This work	RPM	COT	m RP calls	$t(\log \frac{m}{t} - 1)\lambda + \lambda$	–
		sVOLE	m RP calls	$t(\log \frac{m}{t} - 1) \log \lvert\mathbb{K}\rvert + \lambda$	$t(\log \frac{m}{t} + 1) \log \lvert\mathbb{F}\rvert$
		sVOLE	$1.5m$ RP calls	$t(\log \frac{m}{t} - 2)\lambda + 3t \log \lvert\mathbb{K}\rvert + \lambda$	$t \log \lvert\mathbb{F}\rvert$

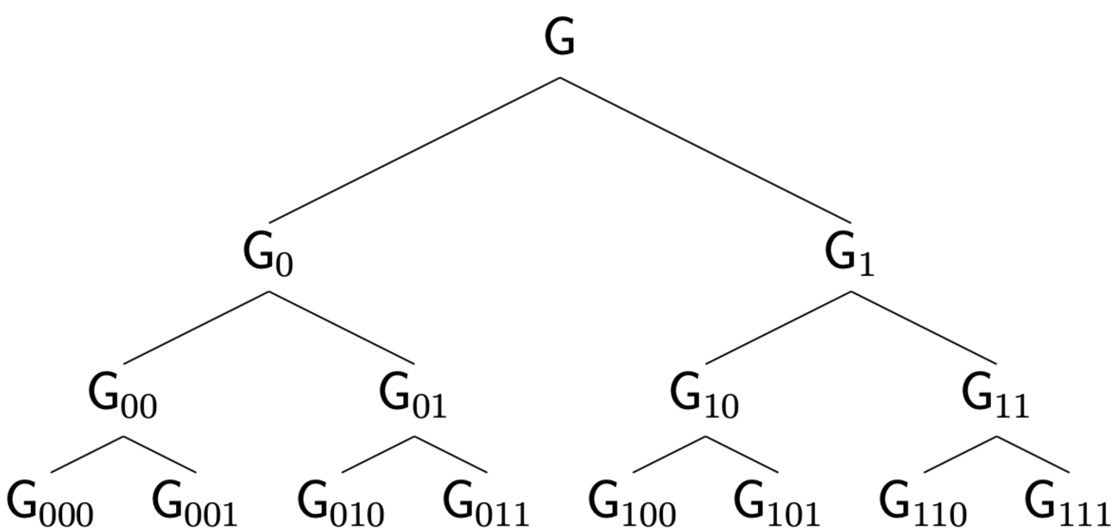
[1] Security relies on the conjecture that the adversary cannot evaluate the punctured result in their RPM-based UPF, where the GGM-style tree expansion uses $G(x) := \mathsf{H}_0(x) \,\|\, \mathsf{H}_1(x)$ for $\mathsf{H}_0(x) := \mathsf{H}(x) \oplus x$ and $\mathsf{H}_1(x) := \mathsf{H}(x) + x \bmod 2^\lambda$.

Table 2: **Comparison with the concurrent work.** "RO/ROM" (resp., "RP/RPM") is short for random oracle (resp., permutation) and the model. m denotes the length of sVOLE correlations. Computation is measured by the amount of symmetric-key operations. In practice, there is also some LPN-related computation cost. Assume weight-t regular LPN noises in sVOLE extension with field $\mathbb{F}$ and extension field $\mathbb{K}$.

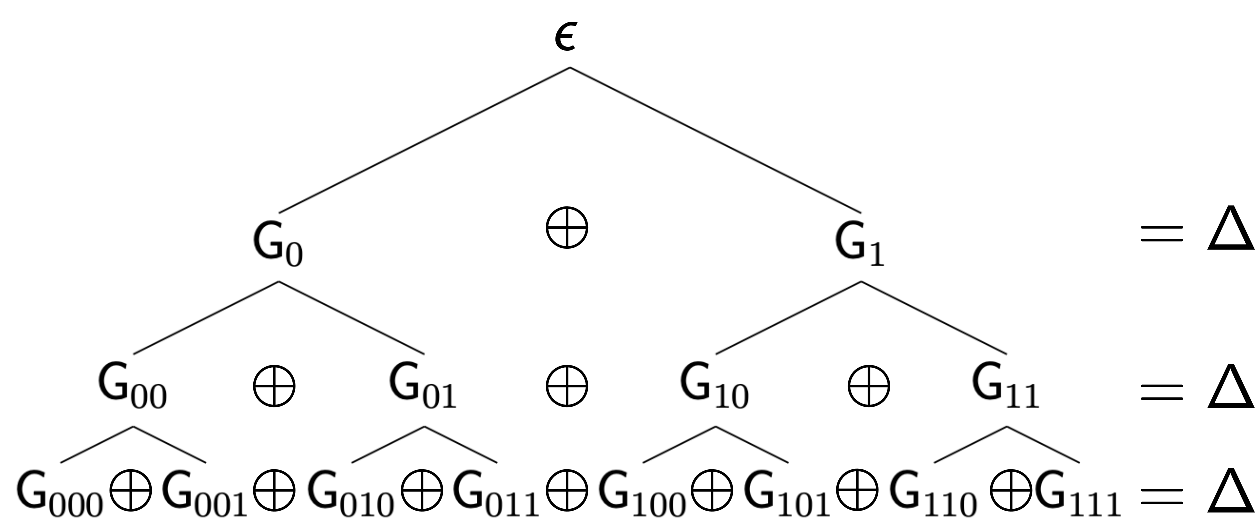
Half-tree Optimization

- Save computation/communication by introducing correlation at each level

GGM Tree

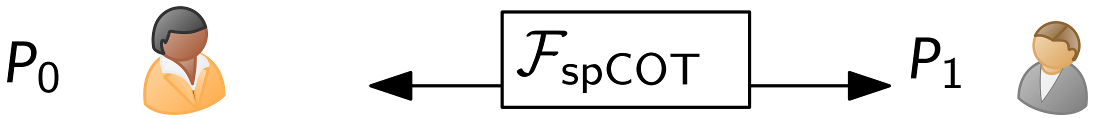


Correlated GGM Tree



- Expansion: $G_{00}||G_{01} = \mathsf{PRG}(G_0)$ $\qquad$ $G_{00} = \mathsf{H}(G_0), G_{01} = G_0 \oplus G_{00}$
- Costs: $N\times$ RO or $2N\times$ RP $\qquad$ $N\times$ RP
- Initial Setup: $G \leftarrow \mathbb{F}_2^\kappa$ $\qquad$ $G_0 = k \leftarrow \mathbb{F}_2^\kappa \quad G_1 = \Delta - k$

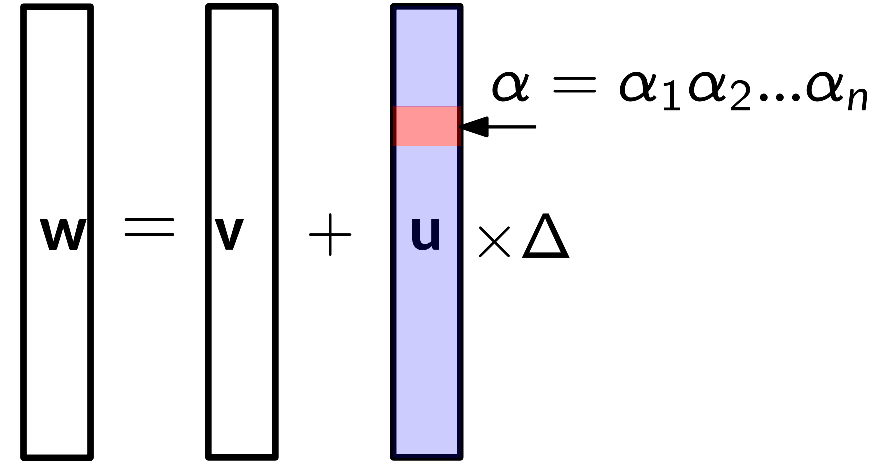
Example 1: Single Point COT



$\Delta \in \mathbb{F}_{2^\kappa}, \mathbf{v} \in \mathbb{F}_{2^\kappa}^N$

$\mathbf{u} \in \mathbb{F}_2^N, \mathbf{w} \in \mathbb{F}_{2^\kappa}^N$

$\|\mathbf{u}\| = 1$



Example 1: Single Point COT

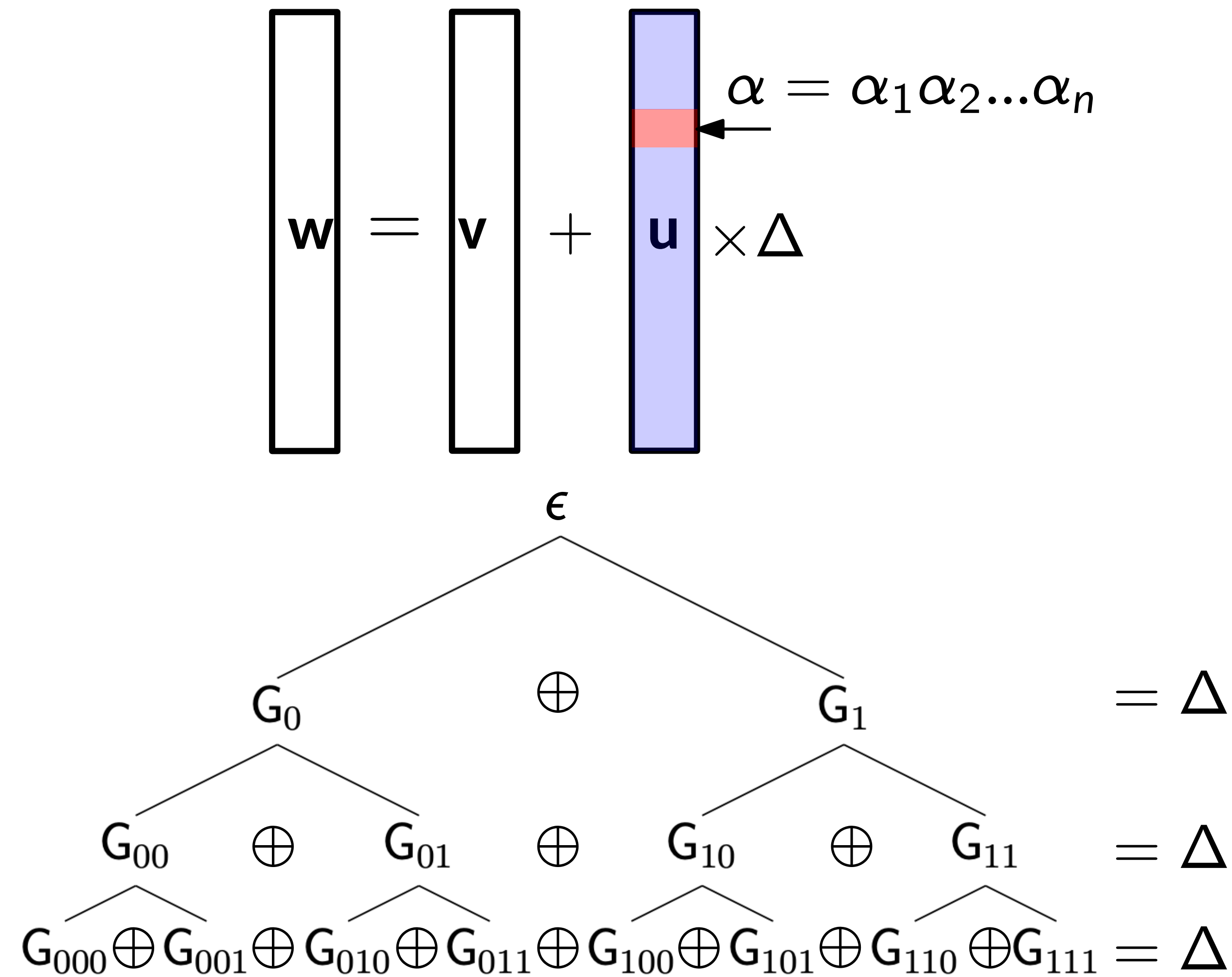
P_0 $\xleftarrow{\quad}$ $\boxed{\mathcal{F}_{\text{spCOT}}}$ $\xrightarrow{\quad}$ P_1

$\Delta \in \mathbb{F}_{2^\kappa}, \mathbf{v} \in \mathbb{F}_{2^\kappa}^N$

$\mathbf{u} \in \mathbb{F}_2^N, \mathbf{w} \in \mathbb{F}_{2^\kappa}^N$

$\|\mathbf{u}\| = 1$

- Setup: $\mathcal{F}_{\text{COT}}$ with Δ global key
- Prepare $[r_1], \ldots, [r_n] \in \mathbb{F}_2^n$



P_0 For $i \in [1, n]$ $\qquad$ P_1

$K_0^i := \bigoplus G_{*0}, K_1^i := \bigoplus G_{*1}$

$$\xrightarrow{c_i := K_0^i \oplus \mathsf{K}[r_i]}$$

For $i \in [1, n]$, $\alpha_i := \bar{r}_i$

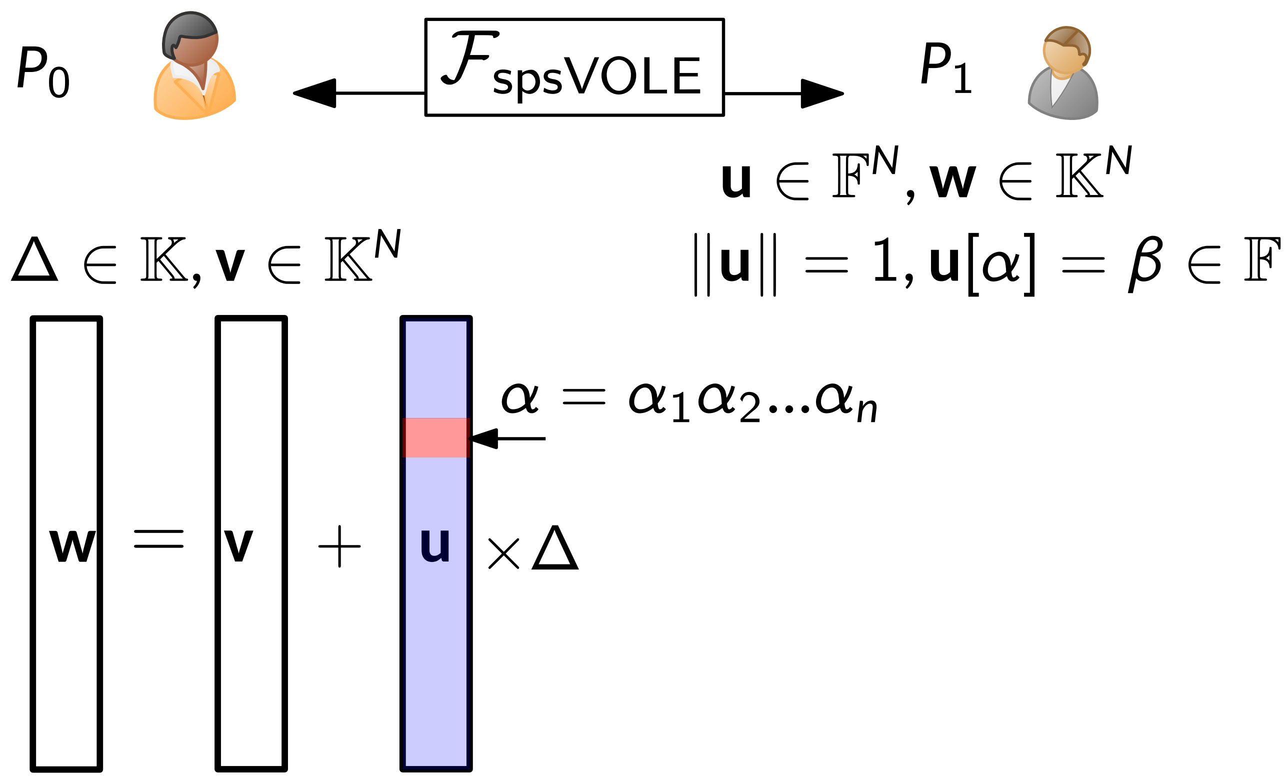
$K_{\bar{\alpha}_i}^i = c_i \oplus \mathsf{M}[r_i] = K_0^i \oplus r_i \cdot \Delta = K_0^i \oplus \bar{\alpha}_i \cdot \Delta$

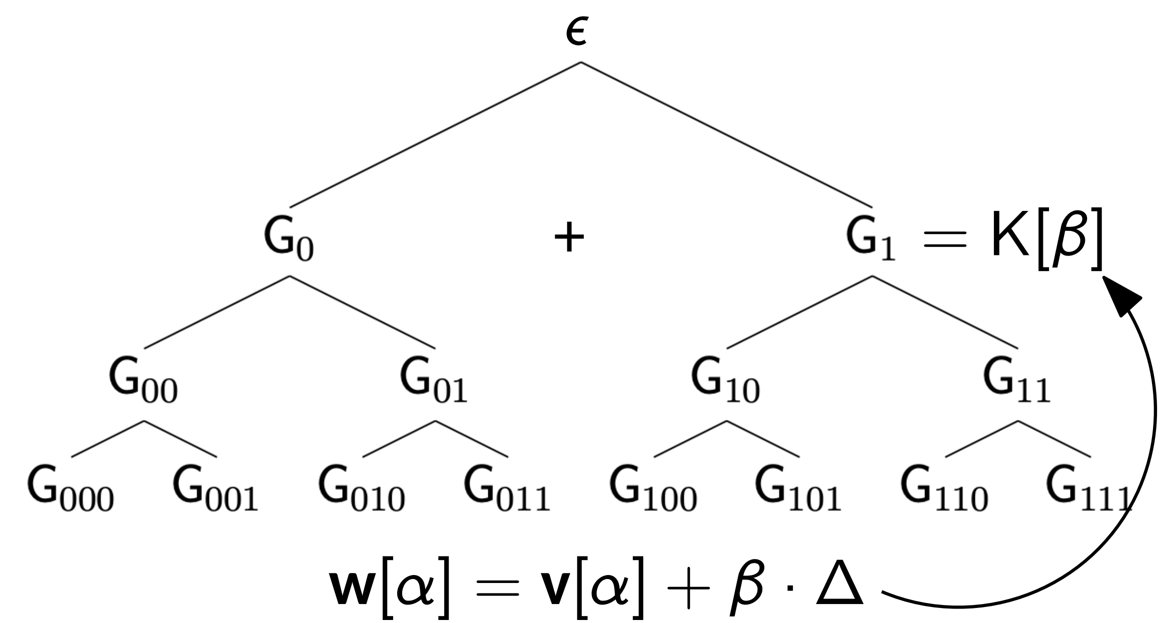
For $x \in [1, N]$

$\mathbf{v}[x] = G_x$

$$\mathbf{w}[x] = \begin{cases} G_x & x \neq \alpha \\ -\sum_{x \neq \alpha} G_x & x = \alpha \end{cases}$$

Example 2: Single Point sVOLE





Example 2: Single Point sVOLE

P_0 ← $\boxed{\mathcal{F}_{\mathsf{spsVOLE}}}$ → P_1

$\Delta \in \mathbb{K}, \mathbf{v} \in \mathbb{K}^N$

$\mathbf{u} \in \mathbb{F}^N, \mathbf{w} \in \mathbb{K}^N$

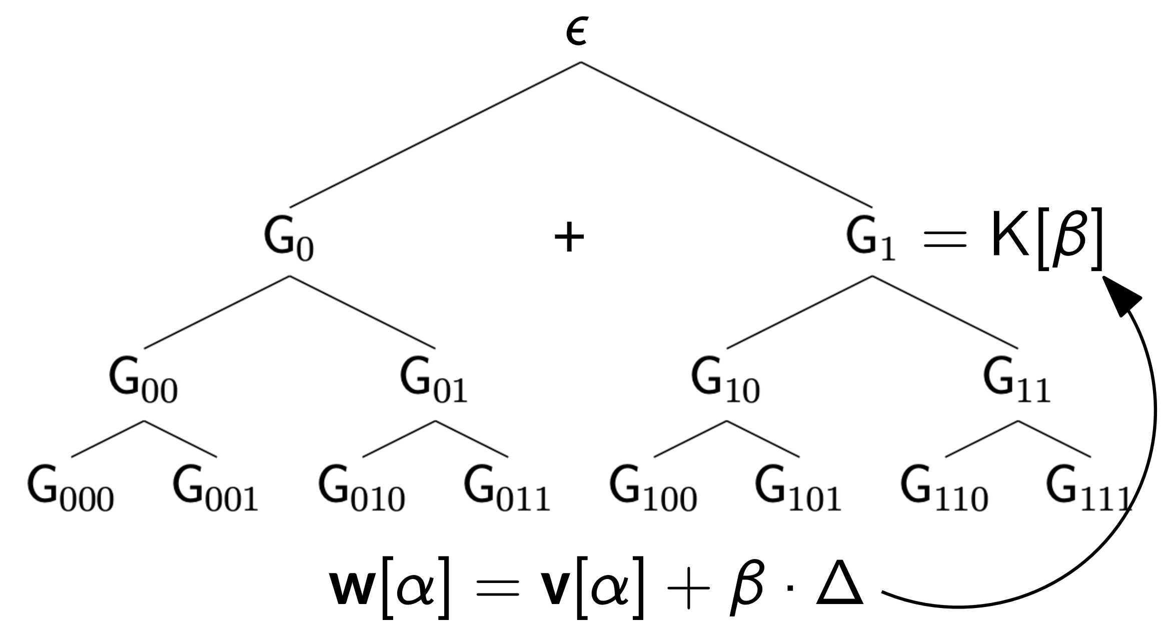
$||\mathbf{u}|| = 1, \mathbf{u}[\alpha] = \beta \in \mathbb{F}$

- Setup: $\mathcal{F}_{\mathsf{sVOLE}}$ with Δ global key
- Prepare $[s_0], [s_1], ..., [s_n]_\Delta$

$$P_0 \xleftarrow{(d_0, d_1, ..., d_n) := (s_0, ..., s_n) - (1, r_1, ..., r_n) \cdot \beta} P_1$$

$\mathsf{K}[\beta] := \mathsf{K}[s_0] + d_0 \cdot \Delta$

$\mathsf{M}[\beta] = \mathsf{M}[s_0]$



Example 2: Single Point sVOLE

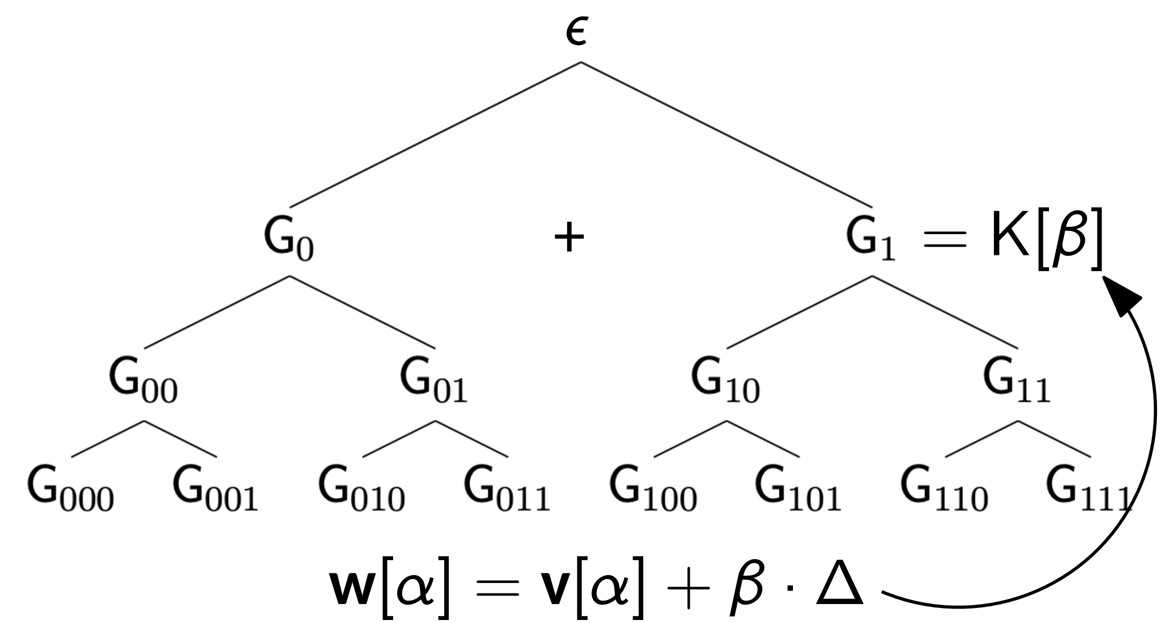
P_0 $\xleftarrow{}$ $\boxed{\mathcal{F}_{\text{spsVOLE}}}$ $\xrightarrow{}$ P_1

$\Delta \in \mathbb{K}, \mathbf{v} \in \mathbb{K}^N$

$\mathbf{u} \in \mathbb{F}^N, \mathbf{w} \in \mathbb{K}^N$

$||\mathbf{u}|| = 1, \mathbf{u}[\alpha] = \beta \in \mathbb{F}$

- Setup: $\mathcal{F}_{\text{sVOLE}}$ with Δ global key
- Prepare $[s_0], [s_1], ..., [s_n]_\Delta$



P_0 $\xleftarrow{(d_0, d_1, ..., d_n) := (s_0, ..., s_n) - (1, r_1, ..., r_n) \cdot \beta}$ P_1

$\mathsf{K}[\beta] := \mathsf{K}[s_0] + d_0 \cdot \Delta$ $\qquad$ $\mathsf{M}[\beta] = \mathsf{M}[s_0]$

- Setup cGGM using $k \leftarrow \mathbb{K}$ and $k - \mathsf{K}[\beta]$

$\mathsf{M}[r_i] = \mathsf{K}[r_i] + r_i \cdot \mathsf{K}[\beta]$

$\mathsf{M}[r_i] = \mathsf{K}[r_i] + r_i \cdot (\mathsf{M}[\beta] - \beta \cdot \Delta)$

$r_i \cdot \mathsf{M}[\beta] - \mathsf{M}[r_i] = -\mathsf{K}[r_i] + r_i \beta \cdot \Delta$

$\mathsf{K}[r_i] := -\mathsf{K}[s_i] - d_i \cdot \Delta$ $\qquad$ $\mathsf{M}[r_i] := r_i \cdot \mathsf{M}[\beta] - \mathsf{M}[s_i]$

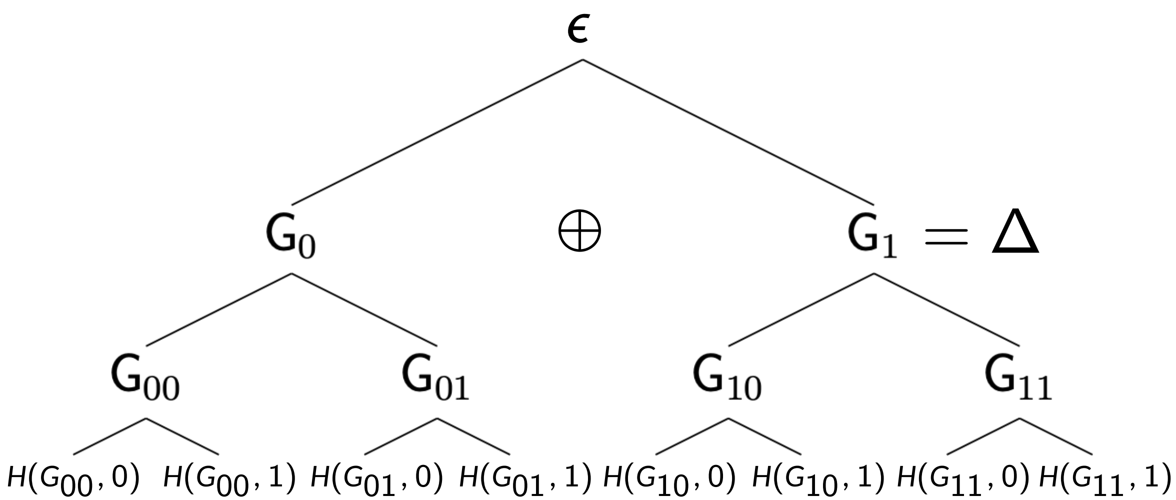
$\xrightarrow{c_i := K_0^i + \mathsf{K}[r_i]}$

For $x \in [1, N]$

$\mathbf{v}[x] = G_x$

$$\mathbf{w}[x] = \begin{cases} G_x & x \neq \alpha \\ -\sum_{x \neq \alpha} G_x + \mathsf{M}[\beta] & x = \alpha \end{cases}$$

Example 3: Single Point sVOLE from Pseudorandom cGGM



P_0 $\longleftarrow$ $\boxed{\mathcal{F}_{\text{spsVOLE}}}$ $\longrightarrow$ P_1

$\Delta \in \mathbb{K}, \mathbf{v} \in \mathbb{K}^N$

$\mathbf{u} \in \mathbb{F}^N, \mathbf{w} \in \mathbb{K}^N$

$||\mathbf{u}|| = 1, \mathbf{u}[\alpha] = \beta \in \mathbb{F}$

- Setup: $\mathcal{F}_{\text{sVOLE}}$ with Γ global key, $\mathcal{F}_{\text{COT}}$ with Δ global key
- Prepare $[r_1], \ldots, [r_n]_\Delta, [s]_\Gamma$

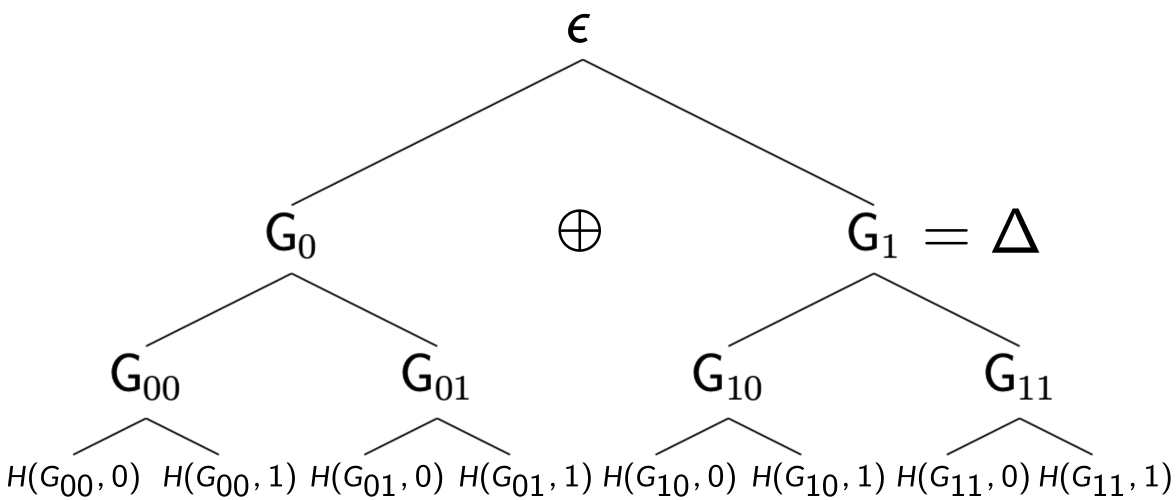
P_0 $\xleftarrow{d := s - \beta}$ P_1

Get $\mathrm{K}[\beta]$

Run pcGGM using $\mathrm{K}[\beta]$

Get $\mathrm{M}[\beta]$

Example 3: Single Point sVOLE from Pseudorandom cGGM



P_0 $\longleftarrow \boxed{\mathcal{F}_{\text{spsVOLE}}} \longrightarrow$ P_1

$\Delta \in \mathbb{K}, \mathbf{v} \in \mathbb{K}^N$

$\mathbf{u} \in \mathbb{F}^N, \mathbf{w} \in \mathbb{K}^N$

$\|\mathbf{u}\| = 1, \mathbf{u}[\alpha] = \beta \in \mathbb{F}$

- Setup: $\mathcal{F}_{\text{sVOLE}}$ with Γ global key, $\mathcal{F}_{\text{COT}}$ with Δ global key
- Prepare $[r_1], ..., [r_n]_\Delta, [s]_\Gamma$

P_0

Get $\mathsf{K}[\beta]$

Run pcGGM using $\mathsf{K}[\beta]$

P_1

Get $\mathsf{M}[\beta]$

$\xleftarrow{d := s - \beta}$

COT: $c_i := K_0^i + \mathsf{K}[r_i]$ for $i \in [1, n-1]$

$\longrightarrow$

OT: $c_n^0 := H(\mathsf{K}[r_n]) + K_0^n, c_n^1 := H(\mathsf{K}[r_n] \oplus \Delta) + K_1^n$

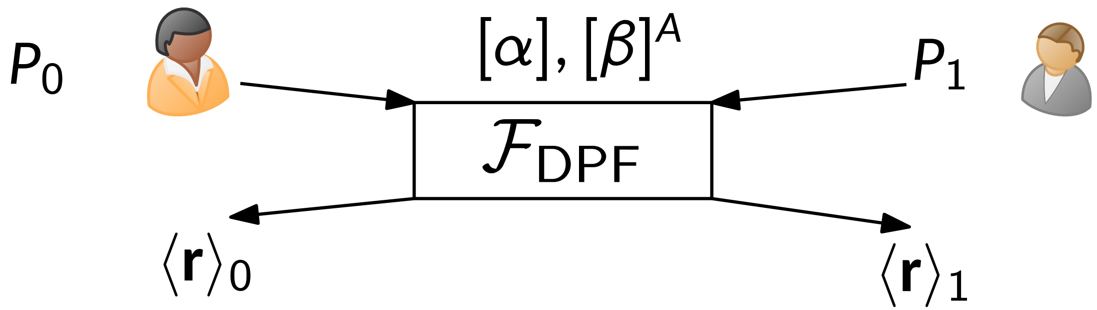
Diff: $\phi := K_0^n + K_1^n - \mathsf{K}[\beta]$

For $x \in [1, N]$

$\mathbf{v}[x] = G_x$

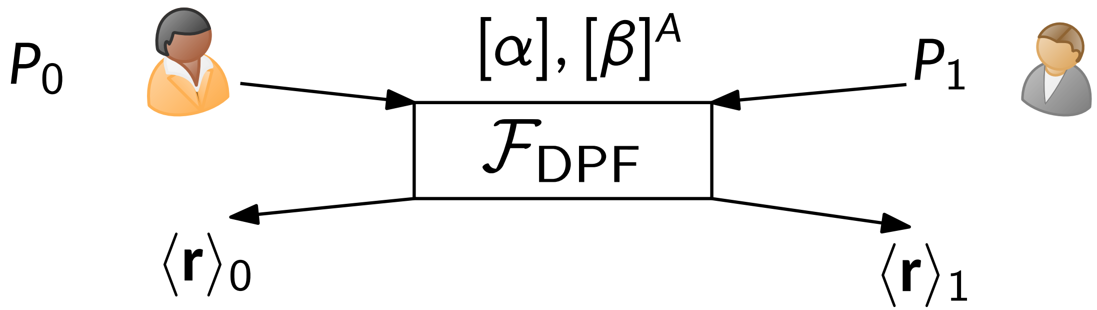
$$\mathbf{w}[x] = \begin{cases} G_x & x \neq \alpha \\ -\sum_{x \neq \alpha} G_x + \phi + \mathsf{M}[\beta] & x = \alpha \end{cases}$$

Example 4: Distributed DPF from pcGGM

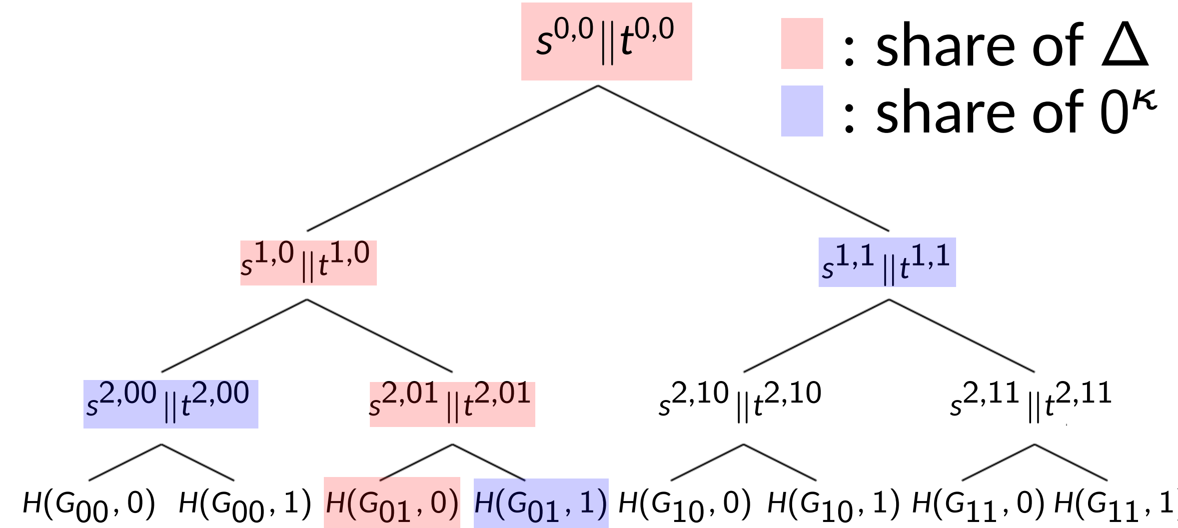


$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \begin{cases} 0 & x \neq \alpha \\ \beta & x = \alpha \end{cases}$$

Example 4: Distributed DPF from pcGGM

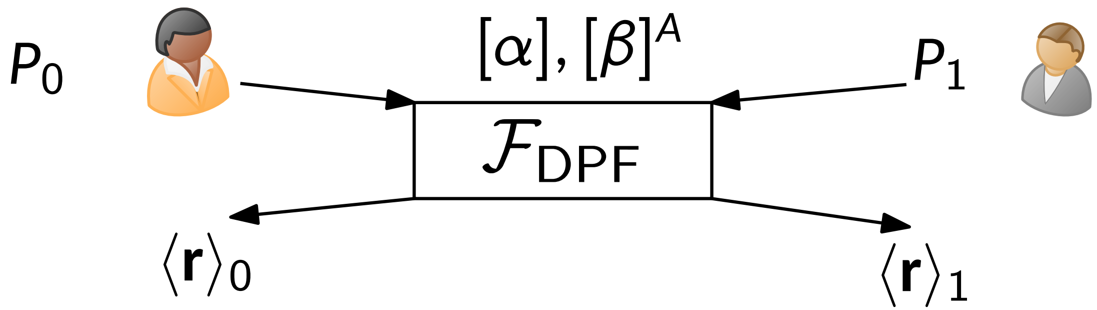


$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \begin{cases} 0 & x \neq \alpha \\ \beta & x = \alpha \end{cases}$$

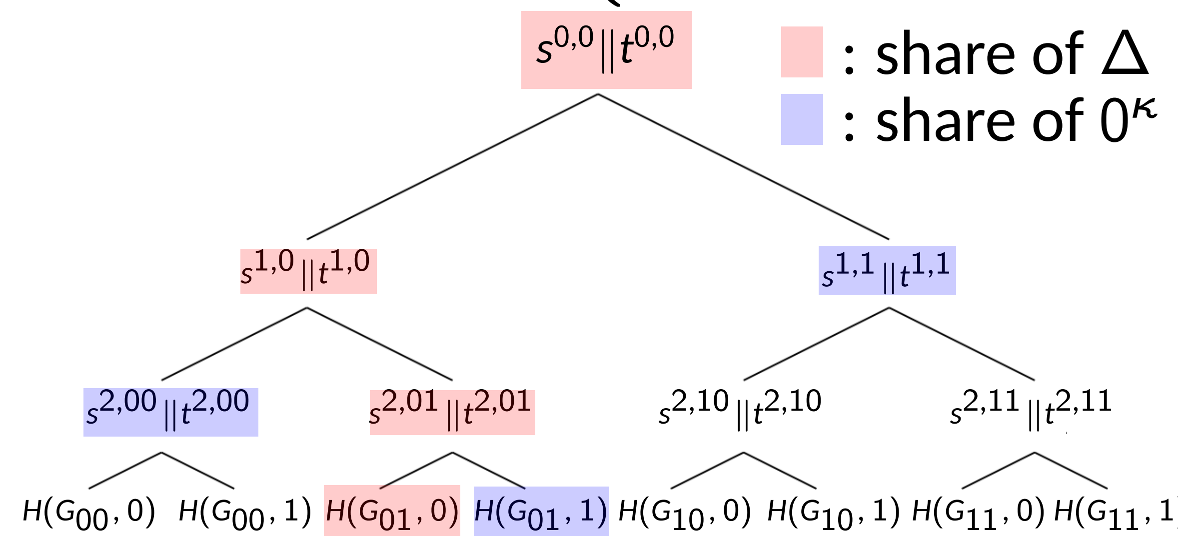


- Sample $\Delta = \langle \Delta \rangle_0 + \langle \Delta \rangle_1$ s.t. $\mathrm{lsb}(\Delta) = 1$
- Authenticate $\langle \alpha \rangle_0, \langle \alpha \rangle_1$
- Run $n + 2$ rounds to compute $CW_1, \ldots, CW_{n+1}$

Example 4: Distributed DPF from pcGGM



$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \begin{cases} 0 & x \neq \alpha \\ \beta & x = \alpha \end{cases}$$



- Sample $\Delta = \langle \Delta \rangle_0 + \langle \Delta \rangle_1$ s.t. $\mathsf{lsb}(\Delta) = 1$
- Authenticate $\langle \alpha \rangle_0, \langle \alpha \rangle_1$
- Run $n + 2$ rounds to compute $CW_1, ..., CW_{n+1}$

$$CW_i := \mathrm{H}(\langle K_0^i \rangle_0) \oplus \mathrm{H}(\langle K_0^i \rangle_1) \oplus \bar{\alpha}_i \cdot \Delta$$

$$CW_n := \mathrm{H}(\langle K_{\bar{\alpha}_n}^n \rangle_0) \oplus \mathrm{H}(\langle K_{\bar{\alpha}_n}^n \rangle_1)$$ NB: need 2-bit to correct LSB

$$CW_{n+1} := \left(\textstyle\sum_{i=1}^N t_0^i - \sum_{i=1}^N t_1^i\right) \cdot \left(\sum_{i=1}^N s_1^i - \sum_{i=1}^N s_0^i + \beta\right)$$

$$\langle \mathbf{r} \rangle_0[x] := (-1)^0 \cdot (s_0^x + t_0^x \cdot CW_{n+1})$$
$$\langle \mathbf{r} \rangle_1[x] := (-1)^1 \cdot (s_1^x + t_1^x \cdot CW_{n+1})$$

$\boxed{\mathcal{F}_{\mathrm{OLE}}}$ is required for non-$\mathbb{F}_{2^k}$ fields

$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \begin{cases} 0 & x \neq \alpha \\ s_0^\alpha - s_1^\alpha + (t_0^\alpha - t_1^\alpha)^2 \cdot CW_{n+1} & x = \alpha \end{cases}$$

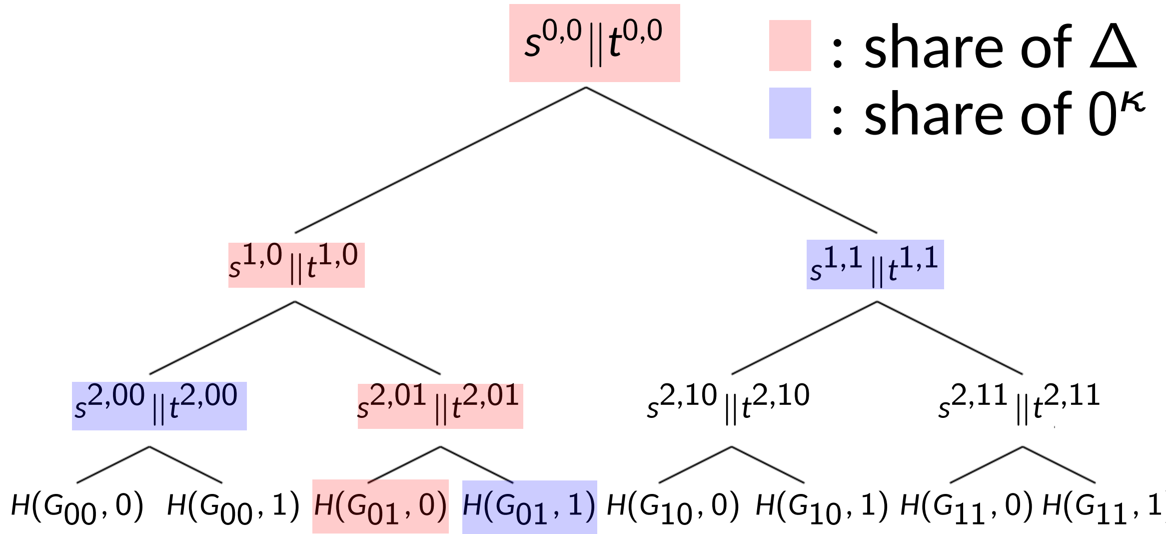
Example 5: Distributed DCF from pcGGM

$$\langle \mathbf{r}\rangle_0[x] + \langle \mathbf{r}\rangle_1[x] = \begin{cases} \beta & x < \alpha \\ 0 & x \geq \alpha \end{cases}$$

- First run DPF
- Then compute $VCW^{i+1} := (t_0^i - t_1^i) \cdot (v_1^{\alpha[1,i]} - v_0^{\alpha[1,i]} + (\alpha_{i+1} - \alpha_i) \cdot \beta)$

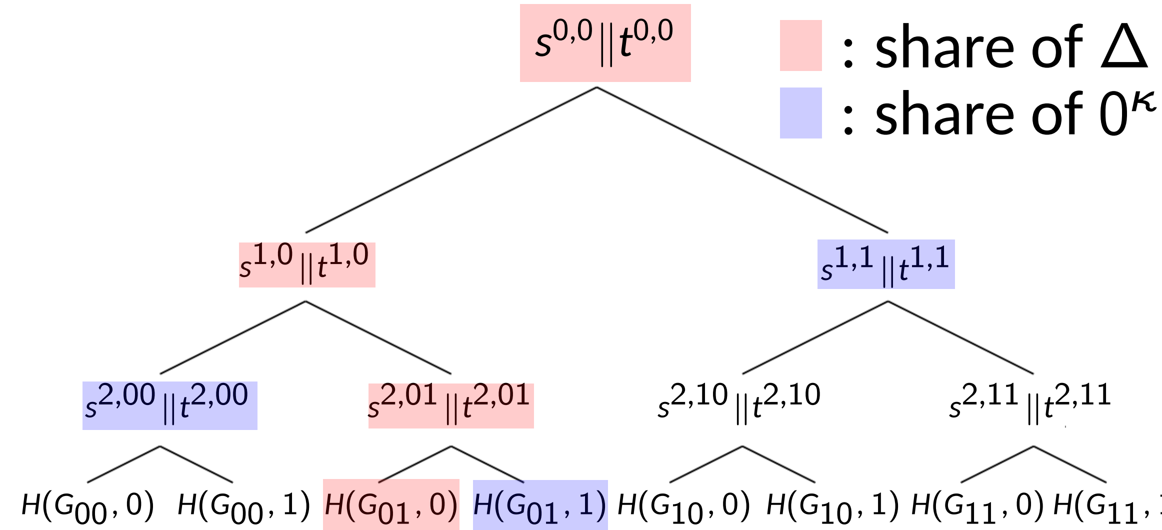
$$\langle \mathbf{r}\rangle_0[x] := \mathrm{DPF}_0(x) + \sum_{i=0}^{n-1}(v_0^{x[1,i]} + t_0^i \cdot VCW_{i+1})$$

$$\langle \mathbf{r}\rangle_1[x] := \mathrm{DPF}_1(x) - \sum_{i=0}^{n-1}(v_1^{x[1,i]} + t_1^i \cdot VCW_{i+1})$$



Example 5: Distributed DCF from pcGGM

$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \begin{cases} \beta & x < \alpha \\ 0 & x \geq \alpha \end{cases}$$



- First run DPF
- Then compute $VCW^{i+1} :=$ $(t_0^i - t_1^i) \cdot (v_1^{\alpha[1,i]} - v_0^{\alpha[1,i]} + (\alpha_{i+1} - \alpha_i) \cdot \beta)$

$$\langle \mathbf{r} \rangle_0[x] := \mathrm{DPF}_0(x) + \textstyle\sum_{i=0}^{n-1} (v_0^{x[1,i]} + t_0^i \cdot VCW_{i+1})$$

$$\langle \mathbf{r} \rangle_1[x] := \mathrm{DPF}_1(x) - \textstyle\sum_{i=0}^{n-1} (v_1^{x[1,i]} + t_1^i \cdot VCW_{i+1})$$

- If $x \neq \alpha$, let $x[1,j] = \alpha[1,j]$,
$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = \mathrm{DPF}_0(x) + \mathrm{DPF}_1(x) + \textstyle\sum_{i=0}^{n-1} (v_0^i - v_1^i + (t_0^i - t_1^i) \cdot VCW_{i+1})$$
$$= \textstyle\sum_{i=0}^{j} (v_0^i - v_1^i + (t_0^i - t_1^i) \cdot VCW_{i+1}) = \textstyle\sum_{i=0}^{j} (\alpha_{i+1} - \alpha_i) \cdot \beta = \alpha_{j+1} \cdot \beta$$
- If $x = \alpha$, we let $\mathrm{DPF}(\alpha) = -\alpha_n \cdot \beta$

$$\langle \mathbf{r} \rangle_0[x] + \langle \mathbf{r} \rangle_1[x] = -\alpha_n \cdot \beta + \textstyle\sum_{i=0}^{n-1} (v_0^i - v_1^i + (t_0^i - t_1^i) \cdot VCW_{i+1}) = -\alpha_n \cdot \beta + \textstyle\sum_{i=0}^{n-1} (\alpha_{i+1} - \alpha_i) \cdot \beta = 0$$