Actively Secure Half-Gates with Minimum Overhead under Duplex Networks

Malicious Half-Gates as Sleek as Semi-Honest

Hongrui Cui[1] Xiao Wang[2] Kang Yang[3] Yu Yu[1,4]

[1] Shanghai Jiao Tong University
[2] Northwestern University
[3] State Key Laboratory of Cryptology
[4] Shanghai Qi Zhi Institute

Apr. 25, 2023 · Eurocrypt 2023

* Some acknowledgments?

Background on Constant Round 2PC

- Garbled circuit is the canonical technique in constant round 2PC

Scheme	XOR	AND (bits)
Textbook Yao	8κ	8κ
Point&Permute	4κ	4κ
GRR3	3κ	3κ
GRR2	2κ	2κ
Free-XOR	0	3κ
fleXOR	$\{0,1,2\}\kappa$	2κ
Half-gates	0	2κ
Three-halves	0	$1.5\kappa + 5$

Background on Constant Round 2PC

- Garbled circuit is the canonical technique in constant round 2PC

Scheme	XOR	AND (bits)
Textbook Yao	8κ	8κ
Point&Permute	4κ	4κ
GRR3	3κ	3κ
GRR2	2κ	2κ
Free-XOR	0	3κ
fleXOR	$\{0, 1, 2\}\kappa$	2κ
Half-gates	0	2κ
Three-halves	0	$1.5\kappa + 5$

- How to boost GC to malicious security?
- AG [KRW17]: Use IT-MAC

$\boxed{\Pi_{\text{pre}}} + \boxed{\Pi_{\text{online}}} = \boxed{\text{Actively-secure constant-round 2PC}}$

Background on Constant Round 2PC

- Garbled circuit is the canonical technique in constant round 2PC

Scheme	XOR	AND (bits)
Textbook Yao	8κ	8κ
Point&Permute	4κ	4κ
GRR3	3κ	3κ
GRR2	2κ	2κ
Free-XOR	0	3κ
fleXOR	$\{0, 1, 2\}\kappa$	2κ
Half-gates	0	2κ
Three-halves	0	$1.5\kappa + 5$

- How to boost GC to malicious security?
- AG [KRW17]: Use IT-MAC

Π_{pre} TinyOT* + Π_{online} [KRRW18] $2\kappa + 1$ bits/AND = Actively-secure constant-round 2PC

Background on Constant Round 2PC

- Garbled circuit is the canonical technique in constant round 2PC

Scheme	XOR	AND (bits)
Textbook Yao	8κ	8κ
Point&Permute	4κ	4κ
GRR3	3κ	3κ
GRR2	2κ	2κ
Free-XOR	0	3κ
fleXOR	$\{0,1,2\}\kappa$	2κ
Half-gates	0	2κ
Three-halves	0	$1.5\kappa+5$

- How to boost GC to malicious security?
- AG [KRW17]: Use IT-MAC

Π_{pre} TinyOT* + Π_{online} [KRRW18] $2\kappa+1$ bits/AND = Actively-secure constant-round 2PC

[DILO22]

- Instantiate $\mathcal{F}_{\text{pre}}$ using PCG
- Optimize Π_{online} to minimize comm.
- $2\kappa+8\rho+O(1)$ bits/AND in $\mathcal{F}_{\text{COT}}$-hybrid
- $2\kappa+4\rho+O(1)$ bits/AND in $\mathcal{F}_{\text{DAMT}}$-hybrid

Background on Constant Round 2PC

- Garbled circuit is the canonical technique in constant round 2PC

Scheme	XOR	AND (bits)
Textbook Yao	8κ	8κ
Point&Permute	4κ	4κ
GRR3	3κ	3κ
GRR2	2κ	2κ
Free-XOR	0	3κ
fleXOR	$\{0, 1, 2\}\kappa$	2κ
Half-gates	0	2κ
Three-halves	0	$1.5\kappa + 5$

- How to boost GC to malicious security?
- AG [KRW17]: Use IT-MAC

Π_{pre} TinyOT* $+$ Π_{online} [KRRW18] $2\kappa + 1$ bits/AND $=$ Actively-secure constant-round 2PC

[DILO22]

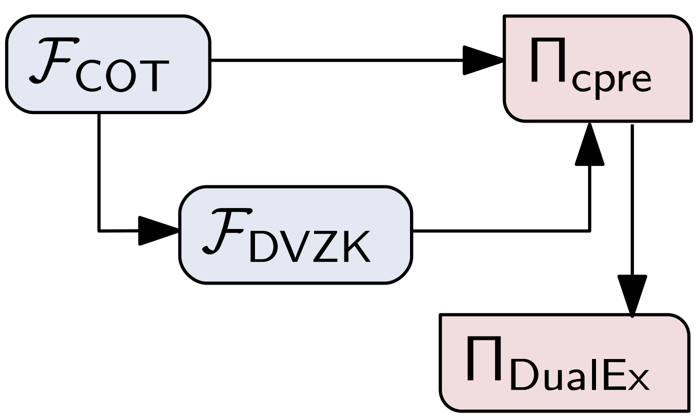
- Instantiate $\mathcal{F}_{\text{pre}}$ using PCG
- Optimize Π_{online} to minimize comm.
- $2\kappa + 8\rho + O(1)$ bits/AND in $\mathcal{F}_{\text{COT}}$-hybrid
- $2\kappa + 4\rho + O(1)$ bits/AND in $\mathcal{F}_{\text{DAMT}}$-hybrid

Can we do better?

Our Contributions

- Authenticated garbling with one-way comm. as small as semi-honest half-gates

2PC	Rounds		Communication per AND gate	
	Prep.	Online	one-way (bits)	two-way (bits)
Half-gates	1	2	2κ	2κ
HSS-PCG	8	2	$8\kappa + 11$ ($4.04\times$)	$16\kappa + 22$ ($8.09\times$)
KRRW-PCG	4	4	$5\kappa + 7$ ($2.53\times$)	$8\kappa + 14$ ($4.05\times$)
DILO	7	2	$2\kappa + 8\rho + 1$ ($2.25\times$)	$2\kappa + 8\rho + 5$ ($2.27\times$)
This work	8	3	$2\kappa + 5$ ($\approx \mathbf{1}\times$)	$4\kappa + 10$ ($2.04\times$)
This work+DILO	8	2	$2\kappa + 3\rho + 2$ ($1.48\times$)	$2\kappa + 3\rho + 4$ ($\approx \mathbf{1.48}\times$)



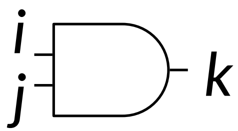
Contribution 1: Π_{cpre} with 2-bit comm. per AND gate

Contribution 2: Consistency checking via dual execution

Semi-Honest Garbled Circuit

$$\boxed{\Lambda_k := \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)}$$

i	j	k
0	0	0
0	1	0
1	0	0
1	1	1



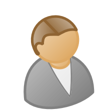
Encrypt & Sort $\longrightarrow$

Λ_i	Λ_j	ciphertext
0	0	$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$

$c_k := \mathsf{lsb}(L_{k,0})$

Sample $\{\lambda_w, L_{w,0}\}$ and Δ

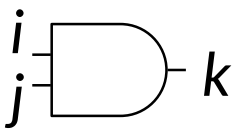
- $L_{w,0} \oplus L_{w,1} = \Delta$
- $\mathsf{lsb}(\Delta) = 1$



Semi-Honest Garbled Circuit

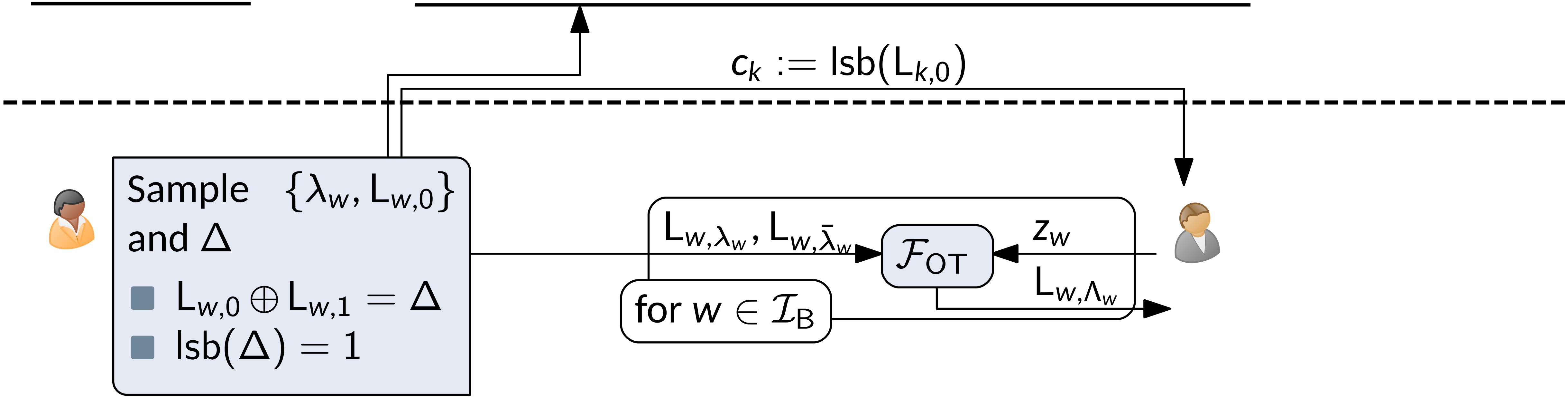
$$\Lambda_k := \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)$$

i	j	k
0	0	0
0	1	0
1	0	0
1	1	1



Encrypt & Sort →

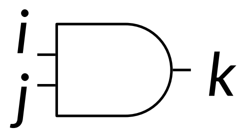
Λ_i	Λ_j	ciphertext
0	0	$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$



Semi-Honest Garbled Circuit

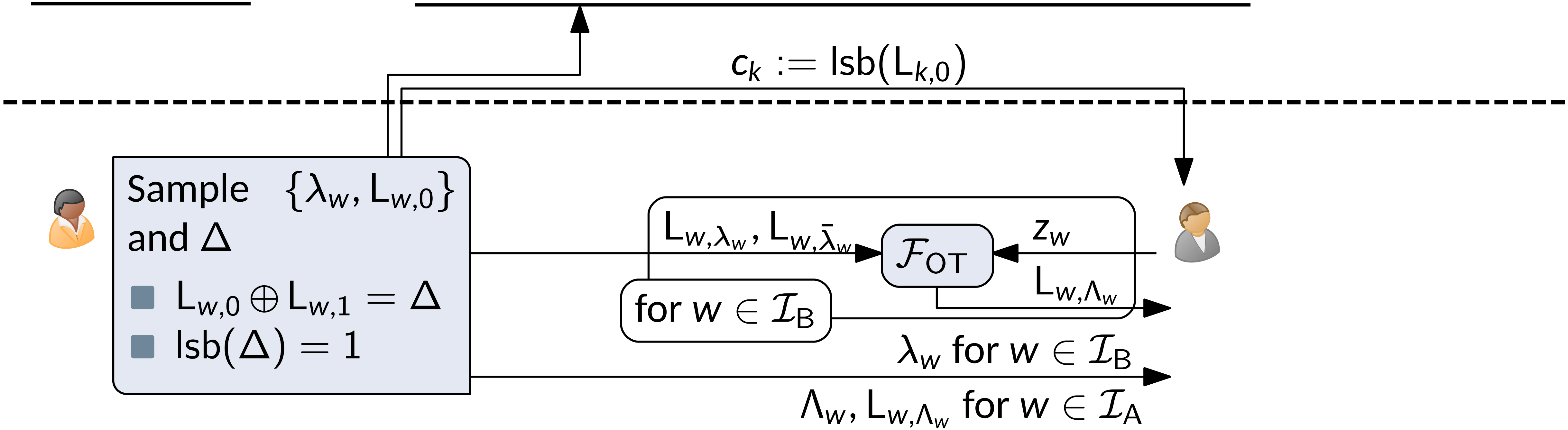
$$\Lambda_k := \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)$$

i	j	k
0	0	0
0	1	0
1	0	0
1	1	1



Encrypt & Sort

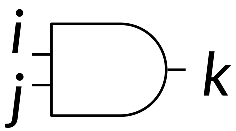
Λ_i	Λ_j	ciphertext
0	0	$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$



Semi-Honest Garbled Circuit

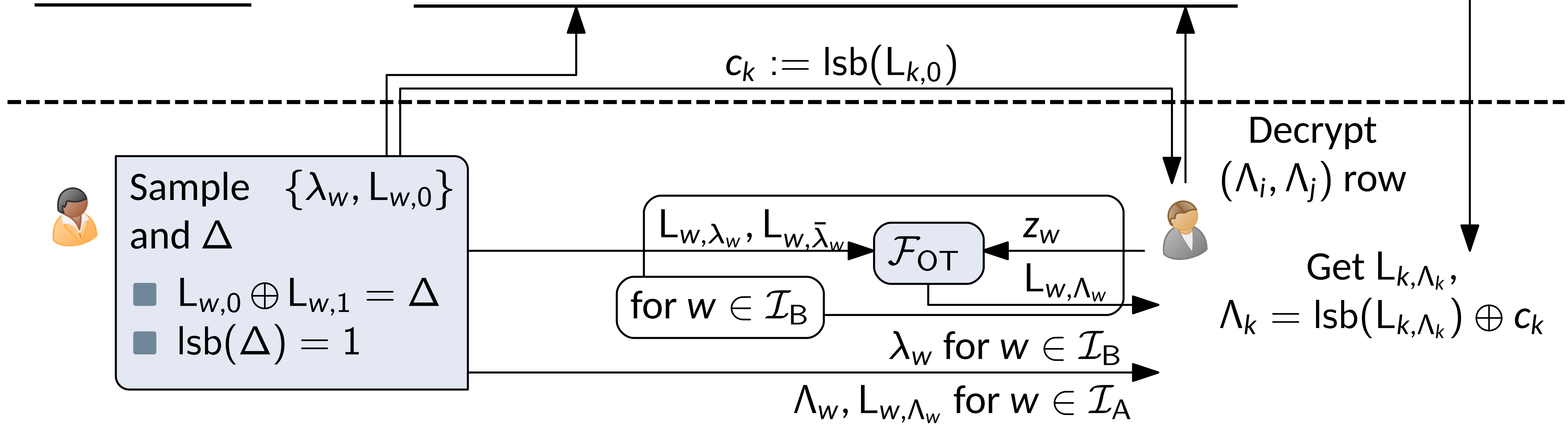
$$\Lambda_k := \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)$$

i	j	k
0	0	0
0	1	0
1	0	0
1	1	1



Encrypt & Sort

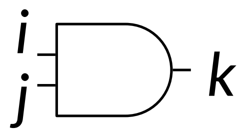
Λ_i	Λ_j	ciphertext
0	0	$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$



Semi-Honest Garbled Circuit

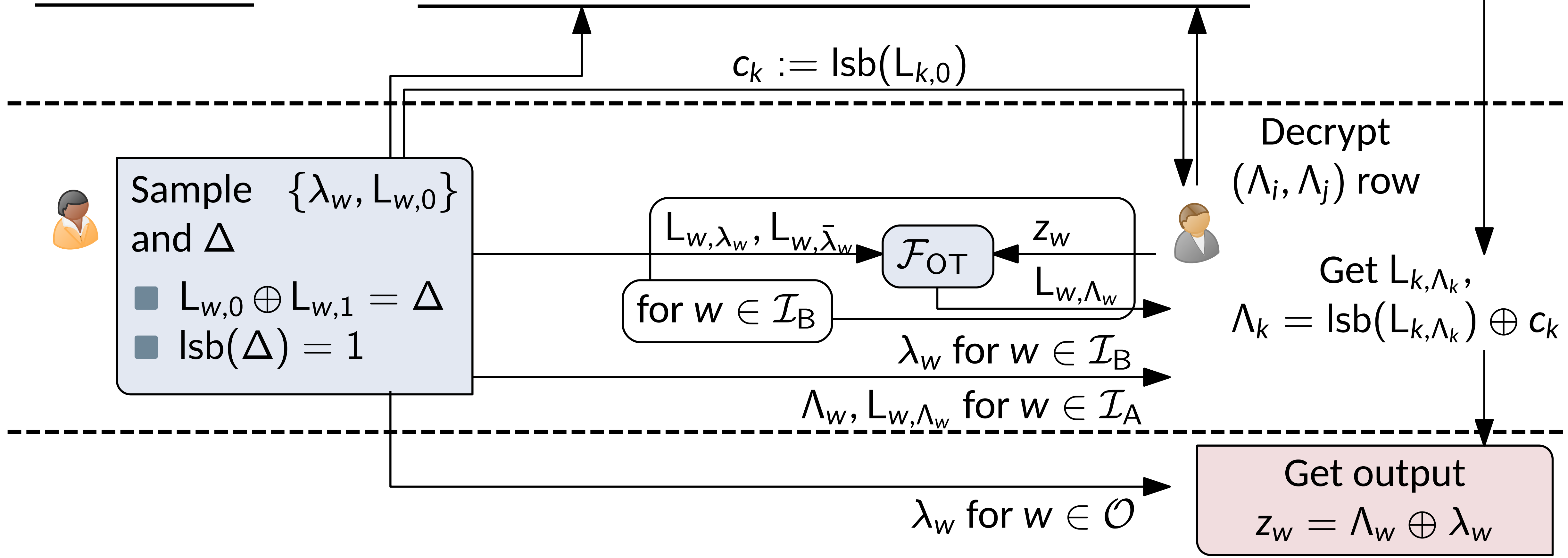
$$\Lambda_k := \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)$$

i	j	k
0	0	0
0	1	0
1	0	0
1	1	1



Encrypt & Sort

Λ_i	Λ_j	ciphertext
0	0	$H(L_{i,0}, L_{j,0}) \oplus L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$H(L_{i,0}, L_{j,1}) \oplus L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$H(L_{i,1}, L_{j,0}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$H(L_{i,1}, L_{j,1}) \oplus L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$



Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

Λ_i	Λ_j	Masked L_{k,Λ_k}
0	0	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
0	1	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	0	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	1	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$

- mount selective-failure attack on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble a different circuit without detection $\Rightarrow$ enforce AND correlation by IT-MAC, equality check, etc.

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

Λ_i	Λ_j	Masked L_{k,Λ_k}
0	0	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
0	1	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	0	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	1	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$

- mount selective-failure attack on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble a different circuit without detection $\Rightarrow$ enforce AND correlation by IT-MAC, equality check, etc.

needs preprocessing information to complete garbling

$\mathcal{F}_{\mathsf{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$, $\Delta_\mathsf{A}, \Delta_\mathsf{B}$

gets $\mathbf{a}, \hat{\mathbf{a}}, \mathsf{M}[\mathbf{a}], \mathsf{M}[\hat{\mathbf{a}}], \mathsf{K}[\mathbf{b}], \mathsf{K}[\hat{\mathbf{b}}]$

gets $\mathbf{b}, \hat{\mathbf{b}}, \mathsf{K}[\mathbf{a}], \mathsf{K}[\hat{\mathbf{a}}], \mathsf{M}[\mathbf{b}], \mathsf{M}[\hat{\mathbf{b}}]$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

Λ_i	Λ_j	Masked L_{k,Λ_k}
0	0	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
0	1	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	0	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	1	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$

- mount selective-failure attack on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble a different circuit without detection $\Rightarrow$ enforce AND correlation by IT-MAC, equality check, etc.

needs preprocessing information to complete garbling

$\mathcal{F}_{\mathsf{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$ $\Delta_\mathsf{A}, \Delta_\mathsf{B}$

gets **a**, M,], K$[\hat{\mathbf{b}}]$

gets **b**, K,], M$[\hat{\mathbf{b}}]$

$\mathsf{M}[\mathbf{a}] \oplus \mathsf{K}[\mathbf{a}] = \mathbf{a} \cdot \Delta_\mathsf{B}$

$\mathsf{K}[\mathbf{b}] \oplus \mathsf{M}[\mathbf{b}] = \mathbf{b} \cdot \Delta_\mathsf{A}$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

- mount selective-failure attack on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble a different circuit without detection $\Rightarrow$ enforce AND correlation by IT-MAC, equality check, etc.

Λ_i	Λ_j	Masked L_{k,Λ_k}
0	0	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
0	1	$\mathsf{L}_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	0	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_\mathsf{A}$
1	1	$\mathsf{L}_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_\mathsf{A}$

$$\Lambda_k \cdot \Delta_\mathsf{A} := \lambda_k \cdot \Delta_\mathsf{A} \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_\mathsf{A}$$
$$= \lambda_k \cdot \Delta_\mathsf{A} \oplus \ldots \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_\mathsf{A}$$

needs preprocessing information to complete garbling

$\mathcal{F}_{\mathsf{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$ $\Delta_\mathsf{A}, \Delta_\mathsf{B}$

gets $\mathbf{a}$, $\mathsf{M}[\mathbf{a}]$, ..., $\mathsf{K}[\mathbf{b}]$, ...,], $\mathsf{K}[\hat{\mathbf{b}}]$

gets $\mathbf{b}$, $\mathsf{K}[\mathbf{a}]$, ..., $\mathsf{M}[\mathbf{b}]$, ...,], $\mathsf{M}[\hat{\mathbf{b}}]$

$\mathsf{M}[\mathbf{a}] \oplus \mathsf{K}[\mathbf{a}] = \mathbf{a} \cdot \Delta_\mathsf{B}$

$\mathsf{K}[\mathbf{b}] \oplus \mathsf{M}[\mathbf{b}] = \mathbf{b} \cdot \Delta_\mathsf{A}$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Λ_i	Λ_j	Alice's GC	Bob's GC
0	0	$\mathsf{L}_{k,0} \oplus \mathsf{K}[\Lambda_{00}]$	$\mathsf{M}[\Lambda_{00}]$
0	1	$\mathsf{L}_{k,0} \oplus \mathsf{K}[\Lambda_{01}]$	$\mathsf{M}[\Lambda_{01}]$
1	0	$\mathsf{L}_{k,0} \oplus \mathsf{K}[\Lambda_{10}]$	$\mathsf{M}[\Lambda_{10}]$
1	1	$\mathsf{L}_{k,0} \oplus \mathsf{K}[\Lambda_{11}]$	$\mathsf{M}[\Lambda_{11}]$

Free-XOR GC $\Rightarrow$
$|\Delta_\mathsf{A}| = \kappa \approx 128$

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

- mount selective-failure attack on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble a different circuit without detection $\Rightarrow$ enforce AND correlation by IT-MAC, equality check, etc.

Λ_i	Λ_j	Masked L_{k,Λ_k}
0	0	$L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
0	1	$L_{k,0} \oplus (\lambda_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$
1	0	$L_{k,0} \oplus (\bar{\lambda}_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
1	1	$L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$

$$\Lambda_k \cdot \Delta_A := \lambda_k \cdot \Delta_A \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_A$$
$$= \lambda_k \cdot \Delta_A \oplus \ldots \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_A$$

needs preprocessing information to complete garbling

$\mathcal{F}_{\text{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$ Δ_A, Δ_B

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

gets $\mathbf{a}$, M], $K[\hat{\mathbf{b}}]$

gets $\mathbf{b}$, K], $M[\hat{\mathbf{b}}]$

$M[\mathbf{a}] \oplus K[\mathbf{a}] = \mathbf{a} \cdot \Delta_B$

$K[\mathbf{b}] \oplus M[\mathbf{b}] = \mathbf{b} \cdot \Delta_A$

Λ_i	Λ_j	Alice's GC	Bob's GC
0	0	$L_{k,0} \oplus K[\Lambda_{00}]$	$M[\Lambda_{00}]$
0	1	$L_{k,0} \oplus K[\Lambda_{01}]$	$M[\Lambda_{01}]$
1	0	$L_{k,0} \oplus K[\Lambda_{10}]$	$M[\Lambda_{10}]$
1	1	$L_{k,0} \oplus K[\Lambda_{11}]$	$M[\Lambda_{11}]$

Λ_i	Λ_j	Alice's AuthGC	Bob's AuthGC
0	0	$L_{k,0} \oplus M[\Lambda_{00}]$	$K[\Lambda_{00}]$
0	1	$L_{k,0} \oplus M[\Lambda_{01}]$	$K[\Lambda_{01}]$
1	0	$L_{k,0} \oplus M[\Lambda_{10}]$	$K[\Lambda_{10}]$
1	1	$L_{k,0} \oplus M[\Lambda_{11}]$	$K[\Lambda_{11}]$

$$\Lambda_k \cdot \Delta_B := \lambda_k \cdot \Delta_B \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_B$$
$$= \lambda_k \cdot \Delta_B \oplus \ldots \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_B$$

Free-XOR GC $\Rightarrow$ $|\Delta_A| = \kappa \approx 128$

IT-MAC Soundness $\Rightarrow$ $|\Delta_B| = \rho \approx 40$

KRRW18: Distributed Half-Gates Garbling + Equality Checking

- $G_{k,0} = H(L_{i,0}) \oplus H(L_{i,1}) \oplus \lambda_j \cdot \Delta_A$
- $G_{k,1} = H(L_{j,0}) \oplus H(L_{j,1}) \oplus \lambda_i \cdot \Delta_A \oplus L_{i,0}$
- $L_{k,0} = H(L_{i,0}) \oplus H(L_{j,0}) \oplus (\lambda_k \oplus \lambda_i\lambda_j) \cdot \Delta_A$

$$\Lambda_k \cdot \Delta_A := \lambda_k \cdot \Delta_A \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_A$$
$$= \underbrace{(\lambda_k \oplus \lambda_i\lambda_j) \cdot \Delta_A}_{\text{already shared}} \oplus \underbrace{\Lambda_i\lambda_j \cdot \Delta_A}_{G_{k,0}} \oplus \underbrace{\Lambda_j(\Lambda_i \oplus \lambda_i) \cdot \Delta_A}_{G_{k,1}}$$

Eval: $H(L_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_{k,0} \oplus H(L_{j,\Lambda_j}) \oplus \Lambda_j \cdot G_{k,1} \oplus L_{i,\Lambda_i}$
$$= H(L_{i,0}) \oplus \Lambda_i\lambda_j \cdot \Delta_A \oplus H(L_{j,0}) \oplus \Lambda_j(\lambda_i \oplus \Lambda_i) \cdot \Delta_A$$
$$= L_{k,0} \oplus \Lambda_k \cdot \Delta_A = L_{k,\Lambda_k}$$

With $\boxed{\mathcal{F}_{\text{pre}}}$ $G_{k,0}, G_{k,1}, L_{k,0}$ are already shared $\Rightarrow$ Evaluator can get $\{\Lambda_w, L_{w,\Lambda_w}\}$ with $2\kappa + 1$ bits amortized comm.

KRRW18: Distributed Half-Gates Garbling + Equality Checking

- $G_{k,0} = H(L_{i,0}) \oplus H(L_{i,1}) \oplus \lambda_j \cdot \Delta_A$
- $G_{k,1} = H(L_{j,0}) \oplus H(L_{j,1}) \oplus \lambda_i \cdot \Delta_A \oplus L_{i,0}$
- $L_{k,0} = H(L_{i,0}) \oplus H(L_{j,0}) \oplus (\lambda_k \oplus \lambda_i\lambda_j) \cdot \Delta_A$

$$\Lambda_k \cdot \Delta_A := \lambda_k \cdot \Delta_A \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_A$$
$$= \underbrace{(\lambda_k \oplus \lambda_i\lambda_j) \cdot \Delta_A}_{\text{already shared}} \oplus \underbrace{\Lambda_i\lambda_j \cdot \Delta_A}_{G_{k,0}} \oplus \underbrace{\Lambda_j(\Lambda_i \oplus \lambda_i) \cdot \Delta_A}_{G_{k,1}}$$

Eval: $H(L_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_{k,0} \oplus H(L_{j,\Lambda_j}) \oplus \Lambda_j \cdot G_{k,1} \oplus L_{i,\Lambda_i}$

$$= H(L_{i,0}) \oplus \Lambda_i\lambda_j \cdot \Delta_A \oplus H(L_{j,0}) \oplus \Lambda_j(\lambda_i \oplus \Lambda_i) \cdot \Delta_A$$
$$= L_{k,0} \oplus \Lambda_k \cdot \Delta_A = L_{k,\Lambda_k}$$

With $\mathcal{F}_{\text{pre}}$ $G_{k,0}, G_{k,1}, L_{k,0}$ are already shared $\Rightarrow$ Evaluator can get $\{\Lambda_w, L_{w,\Lambda_w}\}$ with $2\kappa + 1$ bits amortized comm.

- **b**-mask removes selective failure, now only need to check correct AND correlation

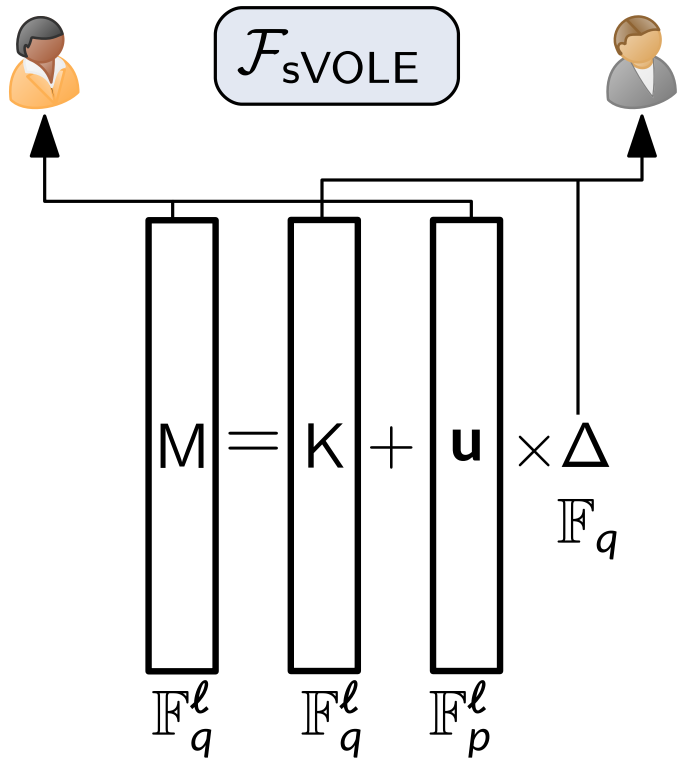
$$\boxed{\Lambda_k \oplus \lambda_k = (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j)} \Leftrightarrow e_k := \Lambda_k \oplus \lambda_k \oplus \Lambda_i\Lambda_j \oplus \Lambda_i\lambda_j \oplus \lambda_i\Lambda_j \oplus \lambda_i\lambda_j = 0$$

Check:
- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$\{\Lambda_k\}$

Defines $[e_k]$ and checks $\sum_k \chi^k e_k = 0$

Efficient COT/VOLE and Designated Verifier Zero Knowledge



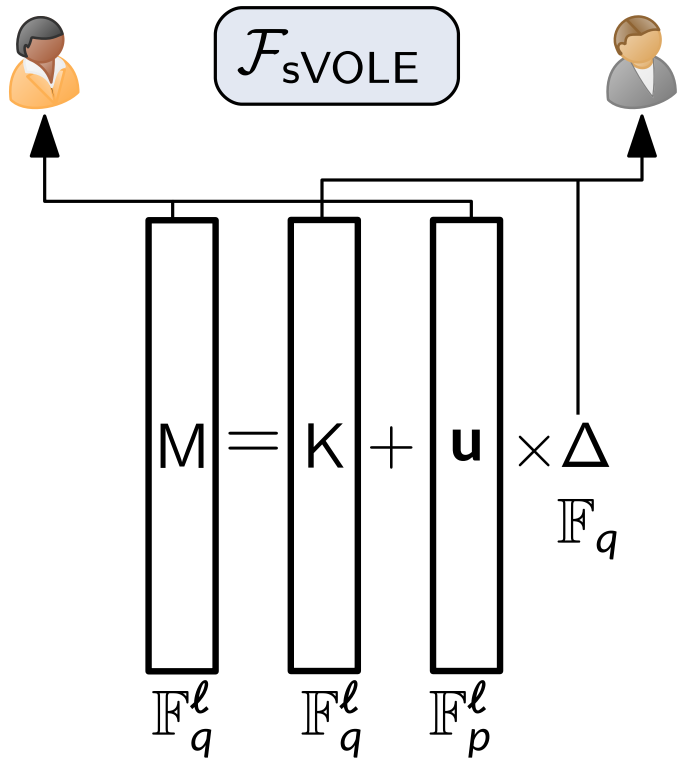
- Efficient protocol for $\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\text{sVOLE}}$ as $\mathcal{F}_{\text{COT}}$

Derandomization operation: Fix

$\xrightarrow{\delta := x \oplus u}$

$\mathsf{M}[x] := \mathsf{M}[u]$ $\qquad$ $\mathsf{K}[x] := \mathsf{K}[u] \oplus \delta \cdot \Delta$

Efficient COT/VOLE and Designated Verifier Zero Knowledge



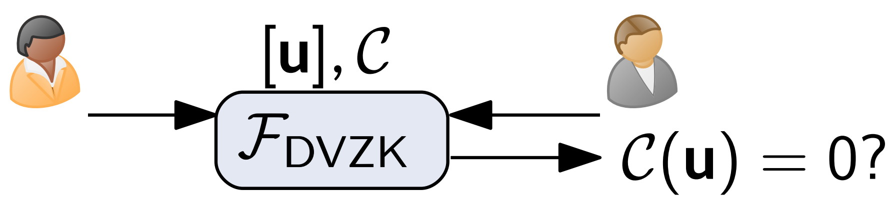
- Efficient protocol for $\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\mathsf{sVOLE}}$ as $\mathcal{F}_{\mathsf{COT}}$

Derandomization operation: Fix

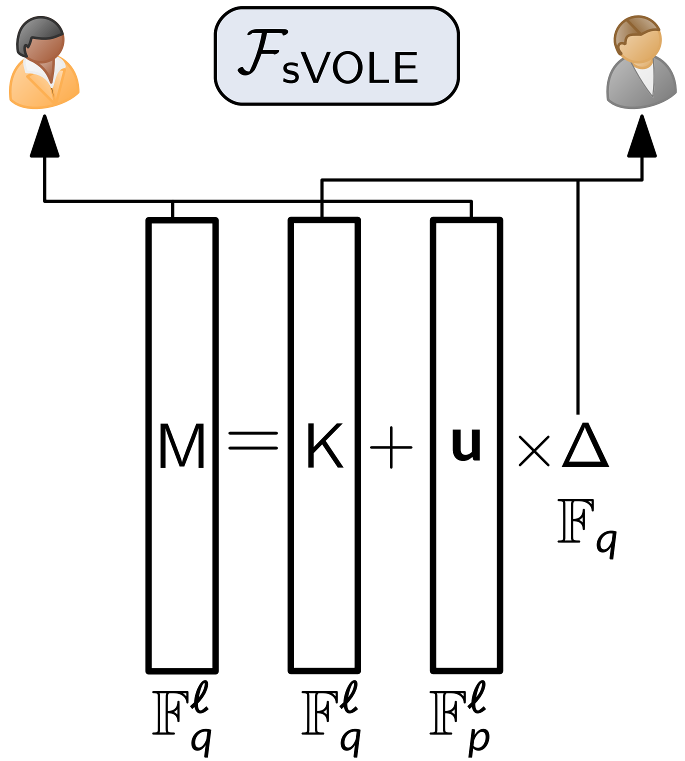
$\delta := x \oplus u$

$\mathsf{M}[x] := \mathsf{M}[u]$ $\qquad$ $\mathsf{K}[x] := \mathsf{K}[u] \oplus \delta \cdot \Delta$

- Efficient proof for deg-d relations on $\mathbf{u}$ [DIO21, YSWW21, ...]



Efficient COT/VOLE and Designated Verifier Zero Knowledge



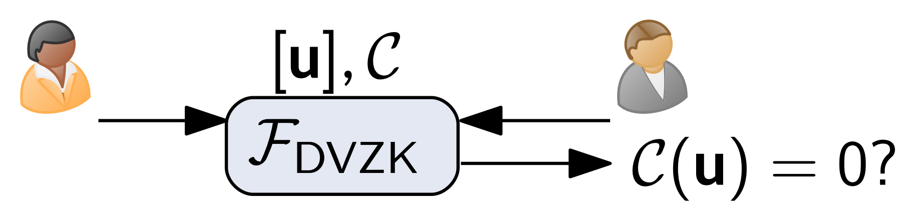
- Efficient protocol for $\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\mathsf{sVOLE}}$ as $\mathcal{F}_{\mathsf{COT}}$

Derandomization operation: Fix

$\delta := x \oplus u$

$\mathsf{M}[x] := \mathsf{M}[u]$ $\quad$ $\mathsf{K}[x] := \mathsf{K}[u] \oplus \delta \cdot \Delta$

- Efficient proof for deg-d relations on $\mathbf{u}$ [DIO21, YSWW21, ...]

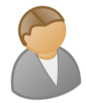


- In DILO, those PCG correlations are called “simple correlations”
- Unfortunately, we still don’t have a direct $\mathcal{F}_{\mathsf{pre}}$ PCG construction
- The closest is the $\mathcal{F}_{\mathsf{DAMT}}$ correlation generated from Ring-LPN, but with ρ-time overhead

AG from Simple Correlations

$\mathcal{F}_{\text{pre}}$

$\{G_{k,0}, G_{k,1}\}$



Eval:

$$\mathsf{H}(\mathsf{L}_{i,\Lambda_i}) \oplus \Lambda_i \cdot \mathsf{G}_{k,0} \oplus \mathsf{H}(\mathsf{L}_{j,\Lambda_i}) \oplus \Lambda_j \cdot \mathsf{G}_{k,1} \oplus \mathsf{L}_{i,\Lambda_i}$$

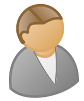
Possibly corrupted

Equivalent to guessing Λ

AG from Simple Correlations

$\mathcal{F}_{\text{pre}}$

$\{G_{k,0}, G_{k,1}\}$



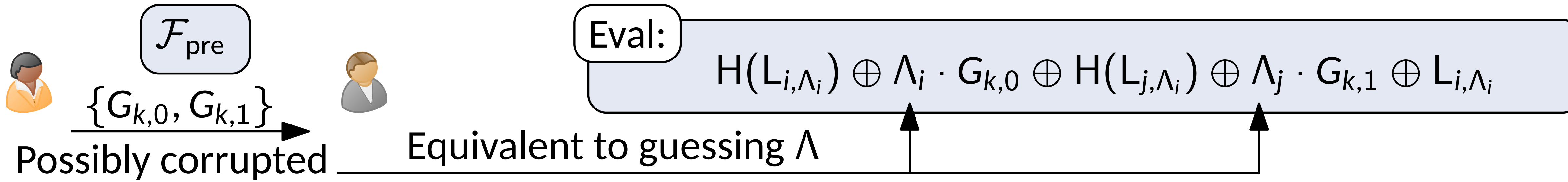
Eval: $H(L_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_{k,0} \oplus H(L_{j,\Lambda_i}) \oplus \Lambda_j \cdot G_{k,1} \oplus L_{i,\Lambda_i}$

Possibly corrupted

Equivalent to guessing Λ

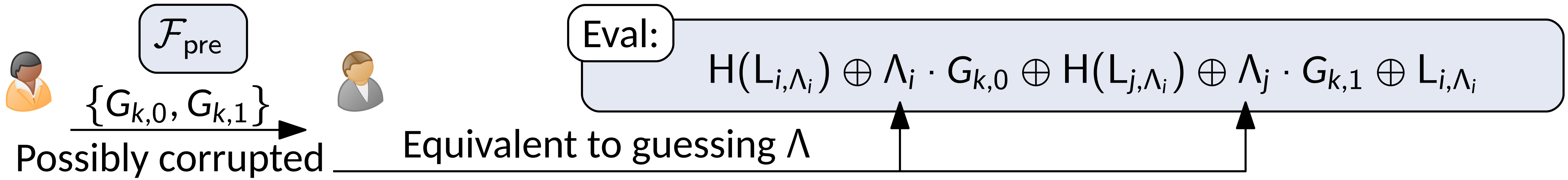
- Garbler can only guess once

AG from Simple Correlations



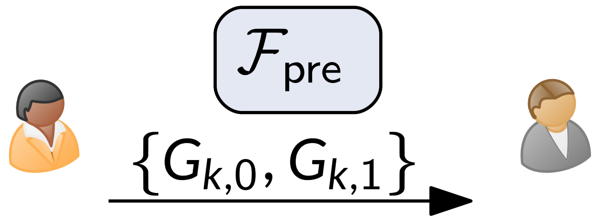
- Garbler can only guess once
- If **b** is uniformly random, then guessing leaks no information

AG from Simple Correlations



- Garbler can only guess once
- If **b** is uniformly random, then guessing leaks no information
- If #Guess is too large, then abort happens overwhelmingly, if #Guess is too little, then we don't require much entropy from **b**

AG from Simple Correlations



Possibly corrupted

Eval: $H(L_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_{k,0} \oplus H(L_{j,\Lambda_i}) \oplus \Lambda_j \cdot G_{k,1} \oplus L_{i,\Lambda_i}$

Equivalent to guessing Λ

- Garbler can only guess once
- If **b** is uniformly random, then guessing leaks no information
- If #Guess is too large, then abort happens overwhelmingly, if #Guess is too little, then we don't require much entropy from **b**

DILO Oberservation 1

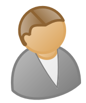
It suffices for **b** to be ρ-wise independent

- #Guess $\leq \rho$: Abort is input-independent
- #Guess $> \rho$: Abort is overwhelming

AG from Simple Correlations

$\mathcal{F}_{\text{pre}}$

$\{G_{k,0}, G_{k,1}\}$ $\longrightarrow$



Possibly corrupted

Eval: $H(L_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_{k,0} \oplus H(L_{j,\Lambda_i}) \oplus \Lambda_j \cdot G_{k,1} \oplus L_{i,\Lambda_i}$

Equivalent to guessing Λ

- Garbler can only guess once
- If **b** is uniformly random, then guessing leaks no information
- If #Guess is too large, then abort happens overwhelmingly, if #Guess is too little, then we don't require much entropy from **b**

DILO Oberservation 1

It suffices for **b** to be ρ-wise independent

- #Guess $\leq \rho$: Abort is input-independent
- #Guess $> \rho$: Abort is overwhelming

DILO Oberservation 2

We can construct ρ-wise independent **b** by linear expansion

$$\boxed{\mathbf{b}} = \boxed{\mathbf{M}} \times \boxed{\mathbf{b}}^*$$

- For $L = O(\rho \cdot \log(\frac{n}{\rho}))$, a uniformly random **M** suffices
- We can encode $\mathbf{b}^*$ in $\mathcal{F}_{\text{COT}}$ global keys

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Encoding $\mathbf{b}^*$ as Global Keys

$\mathcal{F}_{\text{pre}}$

samples

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$

Δ_A, Δ_B

$$\text{s.t. } \hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$$
$$= a_i a_j \oplus a_i b_j \oplus a_j b_i \oplus b_i b_j$$

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Encoding $\mathbf{b}^*$ as Global Keys

$\mathcal{F}_{\text{pre}}$

samples

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$

$\Delta_{\mathsf{A}}, \Delta_{\mathsf{B}}$

$$\text{s.t. } \hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$$
$$= a_i a_j \oplus a_i b_j \oplus a_j b_i \oplus b_i b_j$$

DILO Compression:

$\mathbf{b} = \mathbf{M} \cdot \mathbf{b}^*, \mathbf{b}^* \in \mathbb{F}_2^L$

suffices to compute $\mathbf{a} \otimes \mathbf{b}^*$

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Encoding $\mathbf{b}^*$ as Global Keys

$\mathcal{F}_{\text{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$, Δ_A, Δ_B

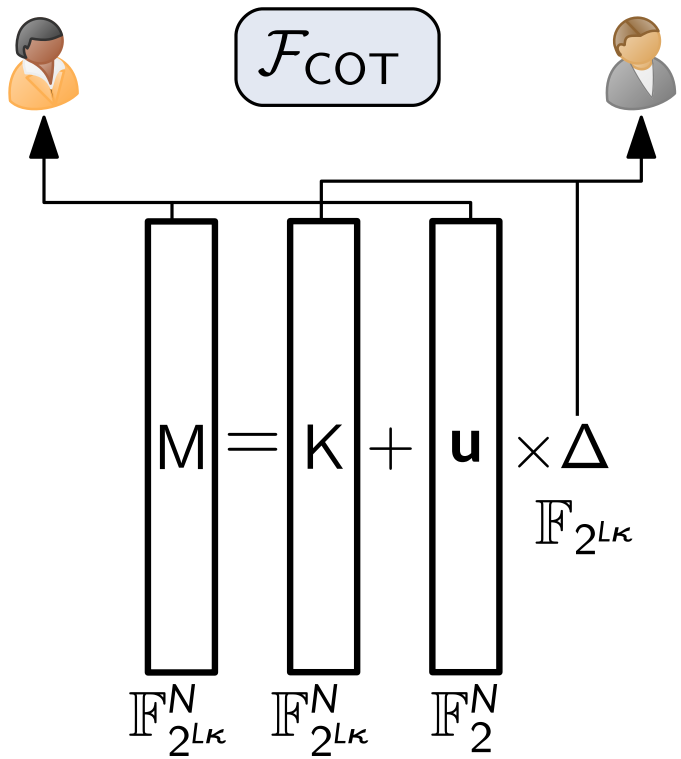
$$\text{s.t. } \hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$$
$$= a_i a_j \oplus a_i b_j \oplus a_j b_i \oplus b_i b_j$$

DILO Compression:
$\mathbf{b} = \mathbf{M} \cdot \mathbf{b}^*, \mathbf{b}^* \in \mathbb{F}_2^L$

suffices to compute $\mathbf{a} \otimes \mathbf{b}^*$

- COT can be extended to block COT, preserving PCG efficiency

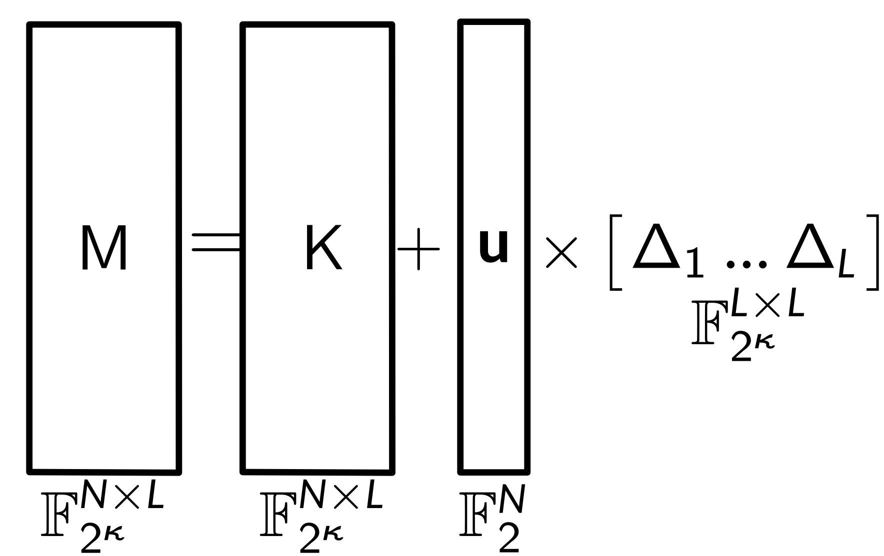
$\mathcal{F}_{\text{COT}}$



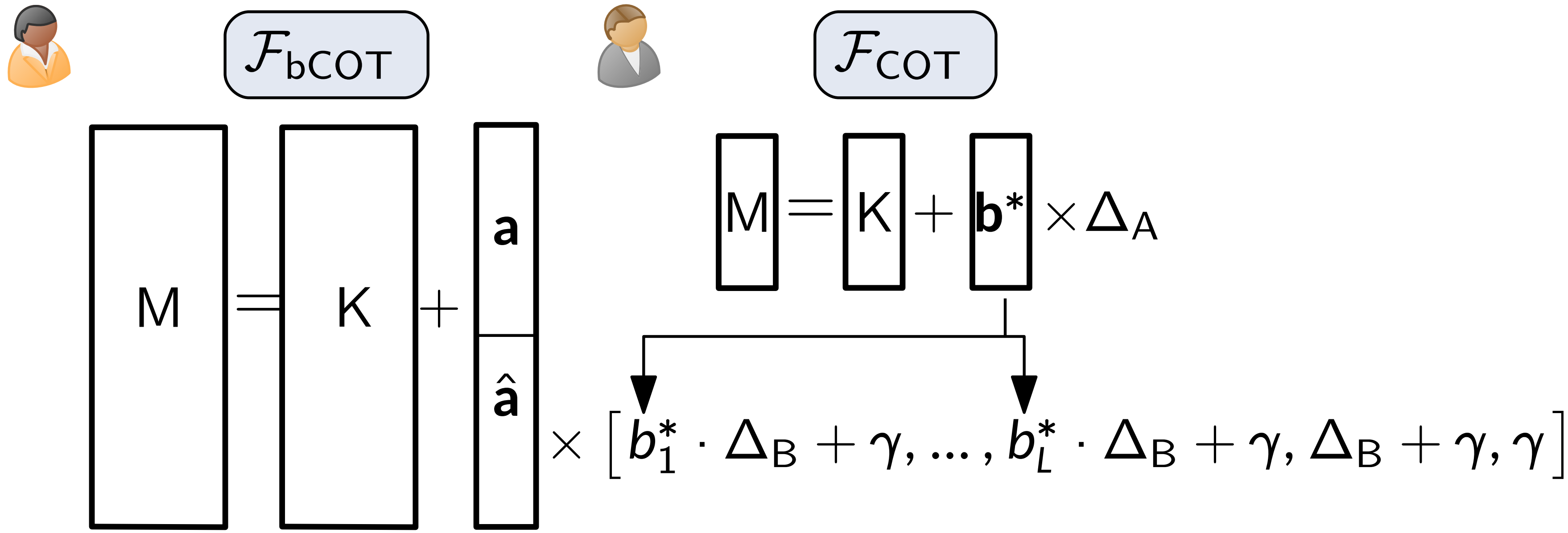
Using isomorphism
$\mathbb{F}_{2^{\kappa L}} \equiv \mathbb{F}_{2^\kappa}^L$

$\Leftrightarrow$

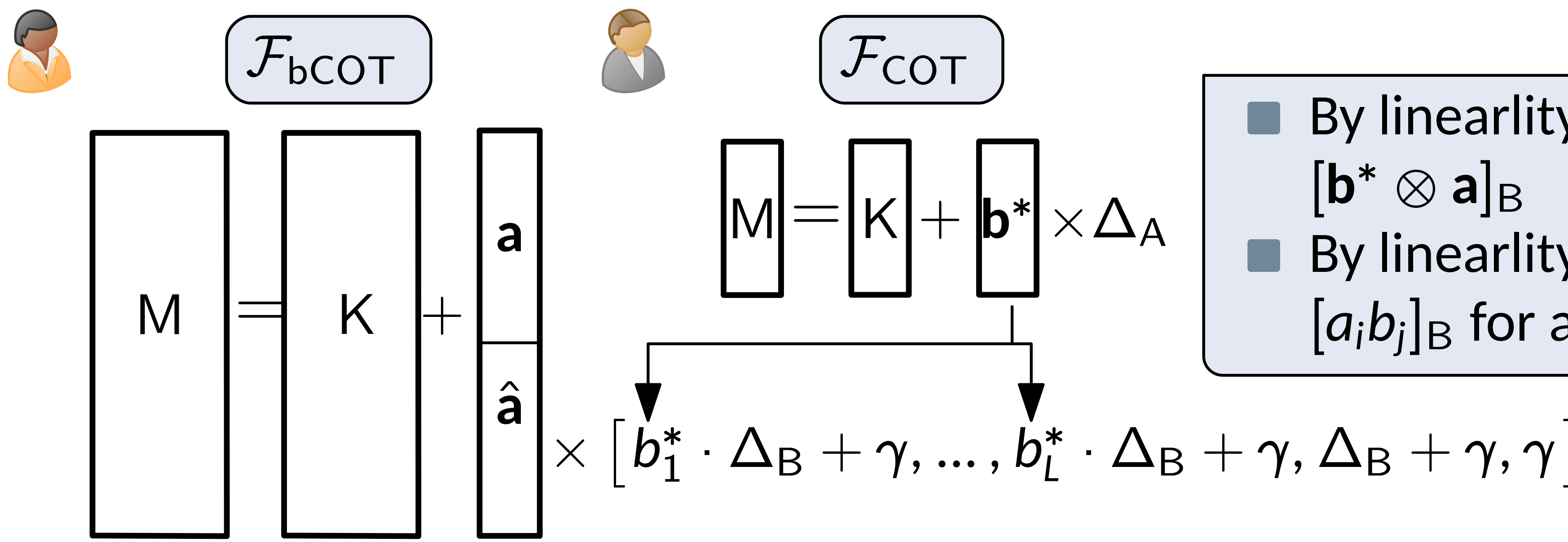
$\mathcal{F}_{\text{bCOT}}$



DILO Implementation of $\mathcal{F}_{\mathrm{cpre}}$: Generating $\tilde{b}_k$



DILO Implementation of $\mathcal{F}_{\mathsf{cpre}}$: Generating $\tilde{b}_k$



- By linearlity on M and K, we can get $[\mathbf{b}^* \otimes \mathbf{a}]_{\mathsf{B}}$
- By linearlity on IT-MAC, we can get $[a_i b_j]_{\mathsf{B}}$ for any i, j

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Generating $\tilde{b}_k$

$\mathcal{F}_{\text{bCOT}}$ $\mathcal{F}_{\text{COT}}$

$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_{\mathsf{A}}$$

$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \begin{bmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{bmatrix} \times \left[b_1^* \cdot \Delta_{\mathsf{B}} + \gamma, \ldots, b_L^* \cdot \Delta_{\mathsf{B}} + \gamma, \Delta_{\mathsf{B}} + \gamma, \gamma \right]$$

- By linearlity on M and K, we can get $[\mathbf{b}^* \otimes \mathbf{a}]_{\mathsf{B}}$
- By linearlity on IT-MAC, we can get $[a_i b_j]_{\mathsf{B}}$ for any i, j

---

$$\xrightarrow{\text{Fix}(\{a_i a_j\})}$$

$\mathcal{F}_{\text{DVZK}}$ Verify mult. correlations

$$\xrightarrow{\text{Fix}(\Delta_{\mathsf{A}})}$$

Verify $\mathbf{b}^* \cdot \Delta_{\mathsf{B}}$ correlation $\mathcal{F}_{\text{DVZK}}$

DILO Implementation of $\mathcal{F}_{\mathsf{cpre}}$: Generating $\tilde{b}_k$

$\mathcal{F}_{\mathsf{bCOT}}$

$$\mathsf{M} = \mathsf{K} + \begin{bmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{bmatrix} \times \left[b_1^* \cdot \Delta_\mathsf{B} + \gamma, \ldots, b_L^* \cdot \Delta_\mathsf{B} + \gamma, \Delta_\mathsf{B} + \gamma, \gamma \right]$$

$\mathcal{F}_{\mathsf{COT}}$

$$\mathsf{M} = \mathsf{K} + \mathbf{b}^* \times \Delta_\mathsf{A}$$

- By linearlity on M and K, we can get $[\mathbf{b}^* \otimes \mathbf{a}]_\mathsf{B}$
- By linearlity on IT-MAC, we can get $[a_i b_j]_\mathsf{B}$ for any i, j

---

$$\xrightarrow{\mathrm{Fix}(\{a_i a_j\})}$$

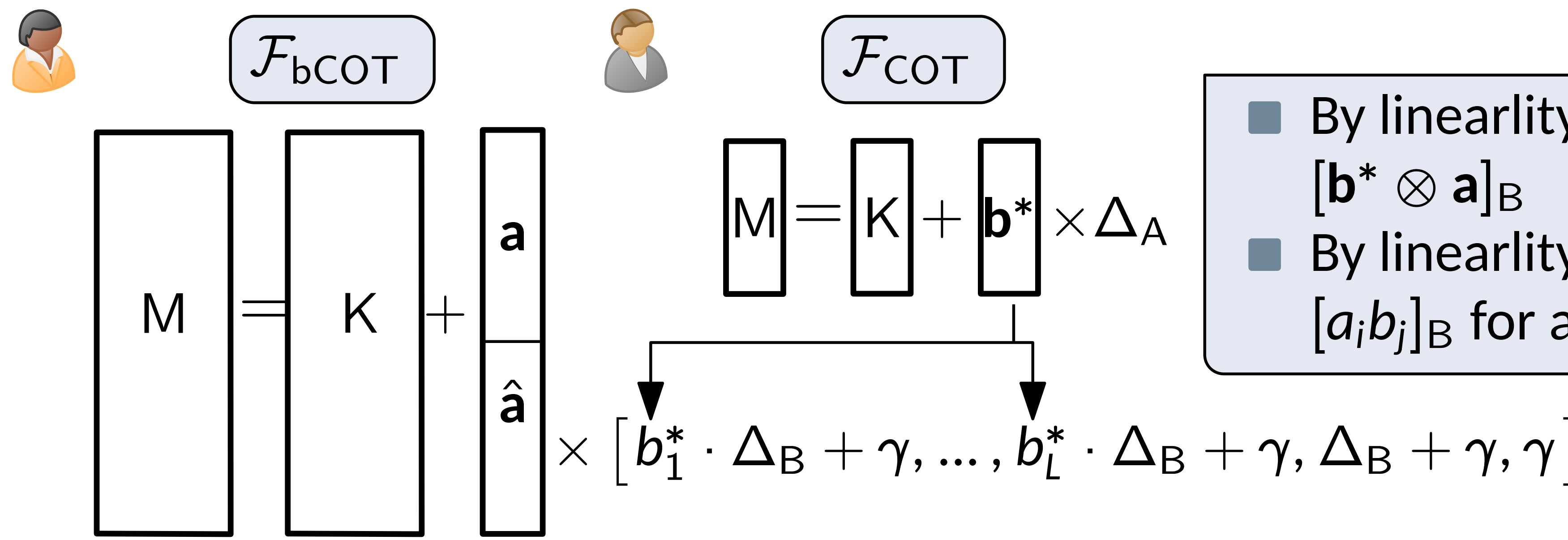
$\mathcal{F}_{\mathsf{DVZK}}$ Verify mult. correlations

$$\xrightarrow{\mathrm{Fix}(\Delta_\mathsf{A})}$$

Verify $\mathbf{b}^* \cdot \Delta_\mathsf{B}$ correlation $\mathcal{F}_{\mathsf{DVZK}}$

Define $[\tilde{b}_k]_\mathsf{B} := [\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i]_\mathsf{B}$

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Generating $\tilde{b}_k$



- By linearlity on M and K, we can get $[\mathbf{b}^* \otimes \mathbf{a}]_B$
- By linearlity on IT-MAC, we can get $[a_i b_j]_B$ for any i, j

$\xrightarrow{\text{Fix}(\{a_i a_j\})}$

$\mathcal{F}_{\text{DVZK}}$ Verify mult. correlations

$\xrightarrow{\text{Fix}(\Delta_A)}$

Verify $\mathbf{b}^* \cdot \Delta_B$ correlation $\mathcal{F}_{\text{DVZK}}$

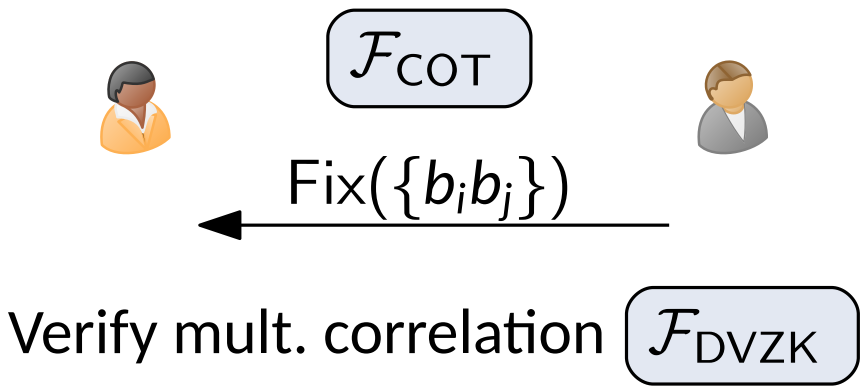
Define $[\tilde{b}_k]_B := [\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i]_B$

$\xrightarrow{m_{k,1} := \mathrm{M}[\tilde{b}_k]}$

- Define $\tilde{b}_k := (m_{k,1} \oplus \mathrm{K}[\tilde{b}_k]) \cdot \Delta_B^{-1}$
- Abort if $\tilde{b}_k \notin \{0, 1\}$
- Compute $\hat{b}_k = b_i b_j \oplus \tilde{b}_k$

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Generating $[\tilde{b}_k]_{\mathsf{A}}$

- It suffices to compute $\tilde{b}_k$ since $[\hat{b}_k]_{\mathsf{A}} = [\tilde{b}_k]_{\mathsf{A}} \oplus [b_i b_j]_{\mathsf{A}}$



- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\mathsf{M}[\tilde{b}_k] \oplus \mathsf{K}[\tilde{b}_k] = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_{\mathsf{A}}$

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Generating $[\tilde{b}_k]_{\mathsf{A}}$

- It suffices to compute $\tilde{b}_k$ since $[\hat{b}_k]_{\mathsf{A}} = [\tilde{b}_k]_{\mathsf{A}} \oplus [b_i b_j]_{\mathsf{A}}$

$\mathcal{F}_{\text{COT}}$

$\xleftarrow{\text{Fix}(\{b_i b_j\})}$

Verify mult. correlation $\mathcal{F}_{\text{DVZK}}$

- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\mathsf{M}[\tilde{b}_k] \oplus \mathsf{K}[\tilde{b}_k] = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_{\mathsf{A}}$

$\mathcal{F}_{\text{bCOT}}$

$$\text{Fix}\begin{pmatrix}\{a_i a_j \Delta_{\mathsf{A}}\} \\ \{\hat{a}_k \Delta_{\mathsf{A}}\} \\ \mathbf{a}\Delta_{\mathsf{A}}\end{pmatrix} \longrightarrow$$

Generate mask $\hat{a}_{k,2} \in \mathbb{F}_{2^\kappa}$ $\longrightarrow$

Locally comptue $[v_k]_{\mathsf{B}} := [\tilde{b}_k \cdot \Delta_{\mathsf{A}} \oplus \hat{a}_{k,2}]_{\mathsf{B}}$

DILO Implementation of $\mathcal{F}_{\mathrm{cpre}}$: Generating $[\tilde{b}_k]_\mathsf{A}$

- It suffices to compute $\tilde{b}_k$ since $[\hat{b}_k]_\mathsf{A} = [\tilde{b}_k]_\mathsf{A} \oplus [b_i b_j]_\mathsf{A}$

$\mathcal{F}_{\mathrm{COT}}$

$\xleftarrow{\mathrm{Fix}(\{b_i b_j\})}$

Verify mult. correlation $\mathcal{F}_{\mathrm{DVZK}}$

- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\mathsf{M}[\tilde{b}_k] \oplus \mathsf{K}[\tilde{b}_k] = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_\mathsf{A}$

$\mathcal{F}_{\mathrm{bCOT}}$

$$\mathrm{Fix}\begin{pmatrix}\{a_i a_j \Delta_\mathsf{A}\} \\ \{\hat{a}_k \Delta_\mathsf{A}\} \\ \mathbf{a}\Delta_\mathsf{A}\end{pmatrix} \longrightarrow$$

Generate mask $\hat{a}_{k,2} \in \mathbb{F}_{2^\kappa}$ $\longrightarrow$

Locally comptue $[v_k]_\mathsf{B} := [\tilde{b}_k \cdot \Delta_\mathsf{A} \oplus \hat{a}_{k,2}]_\mathsf{B}$

$\xrightarrow{m_{k,2} := \mathsf{M}[v_k]}$

Define $\mathsf{K}[\tilde{b}_k] := \hat{a}_{k,2}$

Define $\mathsf{M}[\tilde{b}_k] := (m_{k,2} \oplus \mathsf{K}[v_k]) \cdot \Delta_\mathsf{B}^{-1}$

By the linearity of IT-MAC, $[\hat{b}_k]_\mathsf{A} := [b_i b_j]_\mathsf{A} \oplus [\tilde{b}_k]_\mathsf{A}$

The Online Protocol

KRRW Check:

- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$$\boldsymbol{\Lambda} = \mathbf{a} \oplus \mathbf{b} \oplus \mathbf{z}$$

The Online Protocol

KRRW Check:

- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$$\Lambda = \mathbf{a} \oplus \mathbf{b} \oplus \mathbf{z} \qquad \mathbf{b} = \mathbf{M} \times \mathbf{b}^*$$

The Online Protocol

KRRW Check:

- Evaluator sends $\{\Lambda_w\}$ for all AND gates ✗
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$$\boldsymbol{\Lambda} = \mathbf{a} \oplus \mathbf{b} \oplus \mathbf{z} \qquad \mathbf{b} = \mathbf{M} \times \mathbf{b}^*$$

DILO-WRK Check

$$\begin{aligned}\Lambda_k \cdot \Delta_B &:= \lambda_k \cdot \Delta_B \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_B \\ &= \lambda_k \cdot \Delta_B \oplus \Lambda_i \Lambda_j \cdot \Delta_B \oplus \Lambda_i \lambda_j \cdot \Delta_B \oplus \Lambda_j \lambda_i \cdot \Delta_B \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_B\end{aligned}$$

The Online Protocol

KRRW Check:

- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$$\mathbf{\Lambda} = \mathbf{a} \oplus \mathbf{b} \oplus \mathbf{z} \qquad \mathbf{b} = \mathbf{M} \times \mathbf{b}^*$$

DILO-WRK Check

$$\Lambda_k \cdot \Delta_\mathsf{B} := \lambda_k \cdot \Delta_\mathsf{B} \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_\mathsf{B}$$

$$\boxed{\Lambda_i(a_j \oplus b_j)\Delta_\mathsf{B} = \Lambda_i b_j \Delta_\mathsf{B} \oplus \Lambda_i \mathsf{K}[a_j] \oplus \Lambda_i \mathsf{M}[a_j]}$$

$$= \lambda_k \cdot \Delta_\mathsf{B} \oplus \Lambda_i\Lambda_j \cdot \Delta_\mathsf{B} \oplus \Lambda_i\lambda_j \cdot \Delta_\mathsf{B} \oplus \Lambda_j\lambda_i \cdot \Delta_\mathsf{B} \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_\mathsf{B}$$

The Online Protocol

KRRW Check:

- Evaluator sends $\{\Lambda_w\}$ for all AND gates ✗
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.

$$\boldsymbol{\Lambda} = \mathbf{a} \oplus \mathbf{b} \oplus \mathbf{z} \qquad \mathbf{b} = \mathbf{M} \times \boxed{\mathbf{b}^*}$$

DILO-WRK Check

$$\Lambda_k \cdot \Delta_{\mathrm{B}} := \lambda_k \cdot \Delta_{\mathrm{B}} \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_{\mathrm{B}}$$

$$\Lambda_i(a_j \oplus b_j)\Delta_{\mathrm{B}} = \Lambda_i b_j \Delta_{\mathrm{B}} \oplus \Lambda_i \mathsf{K}[a_j] \oplus \Lambda_i \mathsf{M}[a_j]$$

$$= \lambda_k \cdot \Delta_{\mathrm{B}} \oplus \Lambda_i\Lambda_j \cdot \Delta_{\mathrm{B}} \oplus \Lambda_i\lambda_j \cdot \Delta_{\mathrm{B}} \oplus \Lambda_j\lambda_i \cdot \Delta_{\mathrm{B}} \oplus (\hat{a}_k \oplus \hat{b}_k) \cdot \Delta_{\mathrm{B}}$$

GC

$$\mathsf{G}_0 = \mathsf{H}(\mathsf{L}_{i,0}) \oplus \mathsf{H}(\mathsf{L}_{i,1}) \oplus a_j \cdot \Delta_{\mathrm{A}} \oplus \mathsf{K}[b_j]$$
$$\mathsf{G}_1 = \mathsf{H}(\mathsf{L}_{j,0}) \oplus \mathsf{H}(\mathsf{L}_{j,1}) \oplus a_i \cdot \Delta_{\mathrm{A}} \oplus \mathsf{K}[b_i] \oplus \mathsf{L}_{i,0}$$

Use $\mathsf{G}_{k,0}, \mathsf{G}_{k,1}$ to recover L_{k,Λ_k} without checking

AuthGC

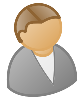
$$\mathsf{G}'_{k,0} = \mathsf{H}'(\mathsf{L}_{i,0}) \oplus \mathsf{H}'(\mathsf{L}_{j,0}) \oplus \mathsf{M}[a_k] \oplus \mathsf{M}[\hat{a}_k]$$
$$\mathsf{G}'_{k,1} = \mathsf{H}'(\mathsf{L}_{i,0}) \oplus \mathsf{H}'(\mathsf{L}_{i,1}) \oplus \mathsf{M}[a_j]$$
$$\mathsf{G}'_{k,2} = \mathsf{H}'(\mathsf{L}_{j,0}) \oplus \mathsf{H}'(\mathsf{L}_{j,1}) \oplus \mathsf{M}[a_i]$$

Use $\mathsf{G}'_{k,0}, \mathsf{G}'_{k,1}, \mathsf{G}'_{k,2}$ to recover Λ_k
Abort if $\mathsf{M}[\Lambda_k] \oplus \mathsf{K}[\Lambda_k] \notin \{0, \Delta_{\mathrm{B}}\}$

$$\mathsf{L}_{k,0} = \mathsf{H}(\mathsf{L}_{i,0}) \oplus \mathsf{H}(\mathsf{L}_{j,0}) \oplus (a_k \oplus \hat{a}_k) \cdot \Delta_{\mathrm{A}} \oplus \mathsf{K}[b_k \oplus \hat{b}_k]$$

Optimizing the Compressed Preprocessing Protocol

The overhead of DILO is
$5\rho + 2$ bits per AND gate

1 bit

$\mathsf{Fix}(\{b_i b_j\})$

$\rho + 1$ bits

$\mathsf{Fix}(\{a_i a_j\})$

$m_{k,1} := \mathsf{M}[\tilde{b}_k]$

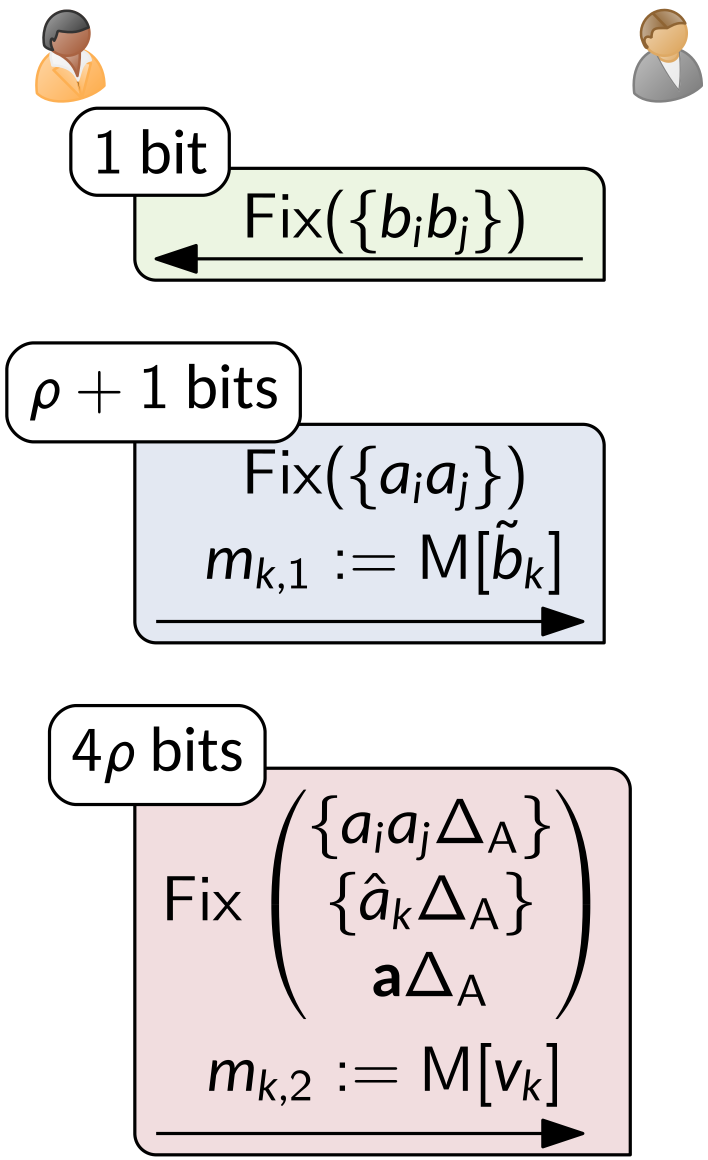
4ρ bits

$$\mathsf{Fix}\begin{pmatrix}\{a_i a_j \Delta_{\mathsf{A}}\} \\ \{\hat{a}_k \Delta_{\mathsf{A}}\} \\ \mathbf{a}\Delta_{\mathsf{A}}\end{pmatrix}$$

$m_{k,2} := \mathsf{M}[v_k]$

Optimizing the Compressed Preprocessing Protocol

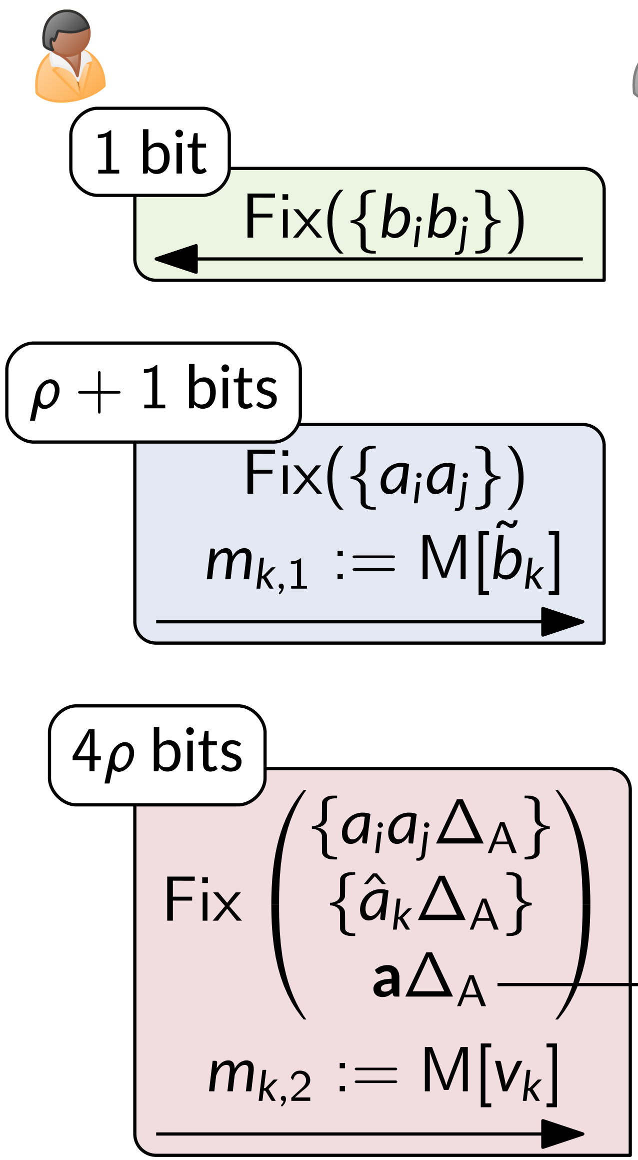
The overhead of DILO is
$5\rho + 2$ bits per AND gate



- Why not call $\mathsf{Fix}(\tilde{b}_k)$ directly?
- We need to detect against dishonest $\mathsf{Fix}()$ input

Optimizing the Compressed Preprocessing Protocol

The overhead of DILO is
$5\rho + 2$ bits per AND gate



- Why not call $\mathsf{Fix}(\tilde{b}_k)$ directly?
- We need to detect against dishonest Fix() input

- $[\mathbf{a}\Delta_\mathsf{A}]_\mathsf{B} \equiv [\mathbf{a}]_{\Delta_\mathsf{A} \cdot \Delta_\mathsf{B}}$

Dual Key Authentication

- $\mathsf{M}[\mathbf{a}\Delta_\mathsf{A}] \oplus \mathsf{K}[\mathbf{a}\Delta_\mathsf{A}] = \mathbf{a}\Delta_\mathsf{A}\Delta_\mathsf{B}$
- We denote it as $\langle \mathbf{a} \rangle$

Optimizing the Compressed Preprocessing Protocol

The overhead of DILO is
$5\rho + 2$ bits per AND gate

1 bit

$\mathsf{Fix}(\{b_i b_j\})$

$\rho + 1$ bits

$\mathsf{Fix}(\{a_i a_j\})$

$m_{k,1} := \mathsf{M}[\tilde{b}_k]$

4ρ bits

$$\mathsf{Fix}\begin{pmatrix}\{a_i a_j \Delta_\mathsf{A}\} \\ \{\hat{a}_k \Delta_\mathsf{A}\} \\ \mathbf{a}\Delta_\mathsf{A}\end{pmatrix}$$

$m_{k,2} := \mathsf{M}[v_k]$

- Why not call $\mathsf{Fix}(\tilde{b}_k)$ directly?
- We need to detect against dishonest Fix() input

Dual Key Authentication

- $[\mathbf{a}\Delta_\mathsf{A}]_\mathsf{B} \equiv [\mathbf{a}]_{\Delta_\mathsf{A} \cdot \Delta_\mathsf{B}}$
- $\mathsf{M}[\mathbf{a}\Delta_\mathsf{A}] \oplus \mathsf{K}[\mathbf{a}\Delta_\mathsf{A}] = \mathbf{a}\Delta_\mathsf{A}\Delta_\mathsf{B}$
- We denote it as $\langle \mathbf{a} \rangle$

- Suppose we generate $\langle \tilde{b}_k \rangle$ and $\langle r \rangle$, $[r]_\mathsf{B}$ (mask for)
- can open $y := \sum_k \chi^k \cdot \tilde{b}_k \oplus r$ and convince
- calls $\mathsf{Fix}(\tilde{b}_k)$ and checks $\sum_k \chi^k [\tilde{b}_k] \oplus [r] \oplus y = 0$

Optimizing the Compressed Preprocessing Protocol

The overhead of DILO is
$5\rho + 2$ bits per AND gate

1 bit

$\mathsf{Fix}(\{b_i b_j\})$

$\rho + 1$ bits

$\mathsf{Fix}(\{a_i a_j\})$

$m_{k,1} := \mathsf{M}[\tilde{b}_k]$

4ρ bits

$$\mathsf{Fix}\begin{pmatrix}\{a_i a_j \Delta_\mathsf{A}\} \\ \{\hat{a}_k \Delta_\mathsf{A}\} \\ \mathbf{a}\Delta_\mathsf{A}\end{pmatrix}$$

$m_{k,2} := \mathsf{M}[v_k]$

- Why not call $\mathsf{Fix}(\tilde{b}_k)$ directly?
- We need to detect against dishonest Fix() input

Dual Key Authentication

- $[\mathbf{a}\Delta_\mathsf{A}]_\mathsf{B} \equiv [\mathbf{a}]_{\Delta_\mathsf{A}\cdot\Delta_\mathsf{B}}$
- $\mathsf{M}[\mathbf{a}\Delta_\mathsf{A}] \oplus \mathsf{K}[\mathbf{a}\Delta_\mathsf{A}] = \mathbf{a}\Delta_\mathsf{A}\Delta_\mathsf{B}$
- We denote it as $\langle \mathbf{a} \rangle$

- Suppose we generate $\langle \tilde{b}_k \rangle$ and $\langle r \rangle$, $[r]_\mathsf{B}$ (mask for)
- can open $y := \sum_k \chi^k \cdot \tilde{b}_k \oplus r$ and convince
- calls $\mathsf{Fix}(\tilde{b}_k)$ and checks $\sum_k \chi^k [\tilde{b}_k] \oplus [r] \oplus y = 0$

If so we can reduce 4ρ bits to 1 bit

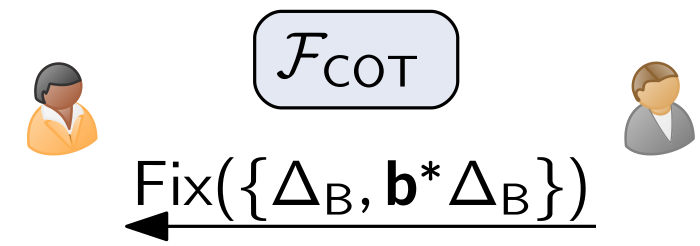
Our goal is to enerate $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

Optimizing the Compressed Preprocessing Protocol, Continued

- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

- $\mathsf{D}_{\mathsf{A}}[\hat{a}_k] \oplus \mathsf{D}_{\mathsf{B}}[\hat{a}_k] = \hat{a}_k \Delta_{\mathsf{A}} \Delta_{\mathsf{B}}$
- $\mathsf{D}_{\mathsf{A}}[a_i b_j] \oplus \mathsf{D}_{\mathsf{B}}[a_i b_j] = a_i b_j \Delta_{\mathsf{A}} \Delta_{\mathsf{B}}$

The compression technique allows encoding **b** in $\mathcal{F}_{\mathsf{bCOT}}$ global keys



Optimizing the Compressed Preprocessing Protocol, Continued

- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

- $\mathsf{D_A}[\hat{a}_k] \oplus \mathsf{D_B}[\hat{a}_k] = \hat{a}_k \Delta_\mathsf{A} \Delta_\mathsf{B}$
- $\mathsf{D_A}[a_i b_j] \oplus \mathsf{D_B}[a_i b_j] = a_i b_j \Delta_\mathsf{A} \Delta_\mathsf{B}$

The compression technique allows encoding **b** in $\mathcal{F}_{\mathsf{bCOT}}$ global keys

$\mathcal{F}_{\mathsf{COT}}$

$$\xleftarrow{\mathsf{Fix}(\{\Delta_\mathsf{B}, \mathbf{b}^* \Delta_\mathsf{B}\})}$$

$$\mathsf{K}[\Delta_\mathsf{B}] \quad \oplus \quad \mathsf{M}[\Delta_\mathsf{B}] = \Delta_\mathsf{A} \Delta_\mathsf{B}$$

$$\mathsf{K}[b_i^* \Delta_\mathsf{B}] \quad \oplus \quad \mathsf{M}[b_i^* \Delta_\mathsf{B}] = b_i^* \Delta_\mathsf{A} \Delta_\mathsf{B}$$

Optimizing the Compressed Preprocessing Protocol, Continued

- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

- $\mathsf{D}_\mathsf{A}[\hat{a}_k] \oplus \mathsf{D}_\mathsf{B}[\hat{a}_k] = \hat{a}_k \Delta_\mathsf{A} \Delta_\mathsf{B}$
- $\mathsf{D}_\mathsf{A}[a_i b_j] \oplus \mathsf{D}_\mathsf{B}[a_i b_j] = a_i b_j \Delta_\mathsf{A} \Delta_\mathsf{B}$

$\mathcal{F}_{\mathsf{bCOT}}^{L+1}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_\mathsf{B}]$$

$\mathcal{F}_{\mathsf{bCOT}}^{2}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_\mathsf{B}]$$

The compression technique allows encoding **b** in $\mathcal{F}_{\mathsf{bCOT}}$ global keys

$\mathcal{F}_{\mathsf{COT}}$

$\xleftarrow{\mathsf{Fix}(\{\Delta_\mathsf{B}, \mathbf{b}^* \Delta_\mathsf{B}\})}$

$$\alpha_0 \oplus \beta_0 = \Delta_\mathsf{A} \Delta_\mathsf{B}$$

$$\alpha_i \oplus \beta_i = b_i^* \Delta_\mathsf{A} \Delta_\mathsf{B}$$

Optimizing the Compressed Preprocessing Protocol, Continued

- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

- $\mathsf{D}_\mathsf{A}[\hat{a}_k] \oplus \mathsf{D}_\mathsf{B}[\hat{a}_k] = \hat{a}_k \Delta_\mathsf{A} \Delta_\mathsf{B}$
- $\mathsf{D}_\mathsf{A}[a_i b_j] \oplus \mathsf{D}_\mathsf{B}[a_i b_j] = a_i b_j \Delta_\mathsf{A} \Delta_\mathsf{B}$

$\mathcal{F}^{L+1}_{\mathsf{bCOT}}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_\mathsf{B}]$$

$\mathcal{F}^{2}_{\mathsf{bCOT}}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_\mathsf{B}]$$

The compression technique allows encoding **b** in $\mathcal{F}_{\mathsf{bCOT}}$ global keys

$\mathcal{F}_{\mathsf{COT}}$

$\mathsf{Fix}(\{\Delta_\mathsf{B}, \mathbf{b}^* \Delta_\mathsf{B}\})$

$$\alpha_0 \oplus \beta_0 = \Delta_\mathsf{A} \Delta_\mathsf{B}$$
$$\alpha_i \oplus \beta_i = b_i^* \Delta_\mathsf{A} \Delta_\mathsf{B}$$

$\Leftrightarrow$

$$\alpha_0 \cdot \Delta_\mathsf{A}^{-1} \oplus \beta_0 \cdot \Delta_\mathsf{A}^{-1} = \Delta_\mathsf{B}$$
$$\alpha_i \cdot \Delta_\mathsf{A}^{-1} \oplus \beta_i \cdot \Delta_\mathsf{A}^{-1} = b_i^* \Delta_\mathsf{B}$$

By $\mathsf{Fix}(\Delta'_\mathsf{A})$

$$\mathsf{K}[\beta_0] \oplus \beta_0 \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_0]$$
$$\mathsf{K}[\beta_i] \oplus \beta_i \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_i]$$

Optimizing the Compressed Preprocessing Protocol, Continued

- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

- $\mathsf{D}_\mathsf{A}[\hat{a}_k] \oplus \mathsf{D}_\mathsf{B}[\hat{a}_k] = \hat{a}_k \Delta_\mathsf{A} \Delta_\mathsf{B}$
- $\mathsf{D}_\mathsf{A}[a_i b_j] \oplus \mathsf{D}_\mathsf{B}[a_i b_j] = a_i b_j \Delta_\mathsf{A} \Delta_\mathsf{B}$

$\mathcal{F}_{\mathsf{bCOT}}^{L+1}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_\mathsf{B}]$$

$\mathcal{F}_{\mathsf{bCOT}}^{2}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_\mathsf{B}]$$

The compression technique allows encoding **b** in $\mathcal{F}_{\mathsf{bCOT}}$ global keys

$\mathcal{F}_{\mathsf{COT}}$

$\mathsf{Fix}(\{\Delta_\mathsf{B}, \mathbf{b}^* \Delta_\mathsf{B}\})$

$$\alpha_0 \oplus \beta_0 = \Delta_\mathsf{A} \Delta_\mathsf{B}$$
$$\alpha_i \oplus \beta_i = b_i^* \Delta_\mathsf{A} \Delta_\mathsf{B}$$

$\Leftrightarrow$

$$\alpha_0 \cdot \Delta_\mathsf{A}^{-1} \oplus \beta_0 \cdot \Delta_\mathsf{A}^{-1} = \Delta_\mathsf{B}$$
$$\alpha_i \cdot \Delta_\mathsf{A}^{-1} \oplus \beta_i \cdot \Delta_\mathsf{A}^{-1} = b_i^* \Delta_\mathsf{B}$$

By $\mathsf{Fix}(\Delta'_\mathsf{A})$

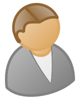
$$\mathsf{K}[\beta_0] \oplus \beta_0 \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_0]$$
$$\mathsf{K}[\beta_i] \oplus \beta_i \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_i]$$

[DIO22] gives a modular way of proving equality under independent keys

Optimizing the Compressed Preprocessing Protocol, Completed

Π_{Samp} $\Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$ \text{ s.t. } \mathsf{lsb}(\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}) = 1$

$\mathcal{F}_{\mathsf{COT}}$



$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_{\mathsf{A}}$$

Optimizing the Compressed Preprocessing Protocol, Completed

Π_{Samp} $\Delta_\mathsf{A}, \Delta_\mathsf{B} \leftarrow \$$ s.t. $\mathsf{lsb}(\Delta_\mathsf{A}\Delta_\mathsf{B}) = 1$

$\mathcal{F}_{\mathsf{COT}}$

$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_\mathsf{A}$$

$\mathsf{Fix}(\{\Delta_\mathsf{B}, \mathbf{b}^*\Delta_\mathsf{B}\})$

$\mathsf{Fix}(r, r \cdot \Delta_\mathsf{B}\}, \{b_i b_j\})$

$\mathcal{F}^{L+1}_{\mathsf{bCOT}}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, ..., \beta_L, \Delta_\mathsf{B}]$$

$\mathcal{F}^{2}_{\mathsf{bCOT}}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_\mathsf{B}]$$

$\mathsf{Fix}(\{a_i a_j\})$

Optimizing the Compressed Preprocessing Protocol, Completed

$\boxed{\Pi_{\mathsf{Samp}}}$ $\Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$ \text{ s.t. } \mathsf{lsb}(\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}) = 1$

$\boxed{\mathcal{F}_{\mathsf{COT}}}$

$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_{\mathsf{A}}$$

$\mathsf{Fix}(\{\Delta_{\mathsf{B}}, \mathbf{b}^*\Delta_{\mathsf{B}}\})$

$\mathsf{Fix}(r, r \cdot \Delta_{\mathsf{B}}\}, \{b_i b_j\})$

$\boxed{\mathcal{F}_{\mathsf{bCOT}}^{L+1}}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_{\mathsf{B}}]$$

$\boxed{\mathcal{F}_{\mathsf{bCOT}}^{2}}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_{\mathsf{B}}]$$

$\mathsf{Fix}(\{a_i a_j\})$

$$\boxed{\text{Define } \langle \tilde{b}_k \rangle := \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle \oplus \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle}$$

Optimizing the Compressed Preprocessing Protocol, Completed

Π_{Samp} $\Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$$ s.t. $\mathsf{lsb}(\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}) = 1$

$\mathcal{F}_{\mathsf{COT}}$

$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_{\mathsf{A}}$$

$\mathsf{Fix}(\{\Delta_{\mathsf{B}}, \mathbf{b}^*\Delta_{\mathsf{B}}\})$

$\mathsf{Fix}(r, r \cdot \Delta_{\mathsf{B}}\}, \{b_i b_j\})$

$\mathcal{F}_{\mathsf{bCOT}}^{L+1}$

$$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, ..., \beta_L, \Delta_{\mathsf{B}}]$$

$\mathcal{F}_{\mathsf{bCOT}}^{2}$

$$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_{\mathsf{B}}]$$

$\mathsf{Fix}(\{a_i a_j\})$

Define $\langle \tilde{b}_k \rangle := \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle \oplus \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle$

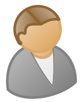
$m_{k,1} := \mathsf{lsb}(\mathsf{D}_{\mathsf{A}}[\tilde{b}_k])$

$\tilde{b}_k = m_{k,1} \oplus \mathsf{lsb}(\mathsf{D}_{\mathsf{A}}[\tilde{b}_k])$

Optimizing the Compressed Preprocessing Protocol, Completed

Π_{Samp} $\Delta_A, \Delta_B \leftarrow \$$ s.t. $\mathsf{lsb}(\Delta_A \Delta_B) = 1$

$\mathcal{F}_{\mathsf{COT}}$



$\boxed{M} = \boxed{K} + \boxed{\mathbf{b}^*} \times \Delta_A$

$\mathsf{Fix}(\{\Delta_B, \mathbf{b}^* \Delta_B\})$

$\mathsf{Fix}(r, r \cdot \Delta_B\}, \{b_i b_j\})$

$\longleftarrow$

$\mathcal{F}_{\mathsf{bCOT}}^{L+1}$

$\boxed{M[\mathbf{a}]} = \boxed{K[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_B]$

$\mathcal{F}_{\mathsf{bCOT}}^{2}$

$\boxed{M[\hat{\mathbf{a}}]} = \boxed{K[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_B]$

$\mathsf{Fix}(\{a_i a_j\})$

$\longrightarrow$

Define $\langle \tilde{b}_k \rangle := \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle \oplus \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle$

$m_{k,1} := \mathsf{lsb}(D_A[\tilde{b}_k])$

$\longrightarrow$

$\tilde{b}_k = m_{k,1} \oplus \mathsf{lsb}(D_A[\tilde{b}_k])$

$\mathsf{Fix}(\tilde{b}_k)$

$\longleftarrow$

Optimizing the Compressed Preprocessing Protocol, Completed

Π_{Samp} $\Delta_A, \Delta_B \leftarrow \$$ s.t. $\mathsf{lsb}(\Delta_A \Delta_B) = 1$

$\mathcal{F}_{\mathsf{COT}}$

$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_A$

$\mathsf{Fix}(\{\Delta_B, \mathbf{b}^*\Delta_B\})$

$\mathsf{Fix}(r, r \cdot \Delta_B\}, \{b_i b_j\})$

$\mathcal{F}_{\mathsf{bCOT}}^{L+1}$

$\boxed{\mathsf{M}[\mathbf{a}]} = \boxed{\mathsf{K}[\mathbf{a}]} + \boxed{\mathbf{a}} \times [\beta_1, \ldots, \beta_L, \Delta_B]$

$\mathcal{F}_{\mathsf{bCOT}}^{2}$

$\boxed{\mathsf{M}[\hat{\mathbf{a}}]} = \boxed{\mathsf{K}[\hat{\mathbf{a}}]} + \boxed{\hat{\mathbf{a}}} \times [\beta_0, \Delta_B]$

$\mathsf{Fix}(\{a_i a_j\})$

Define $\langle \tilde{b}_k \rangle := \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle \oplus \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle$

$m_{k,1} := \mathsf{lsb}(\mathsf{D}_A[\tilde{b}_k])$

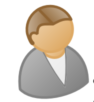
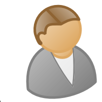
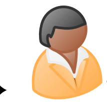
$\tilde{b}_k = m_{k,1} \oplus \mathsf{lsb}(\mathsf{D}_A[\tilde{b}_k])$

$\mathsf{Fix}(\tilde{b}_k)$

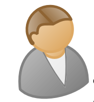
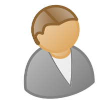
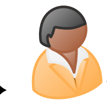
$y := r + \sum_k \chi^k \cdot \tilde{b}_k$

- Check $\{\beta_i\}$ consistency by $\mathsf{Fix}(\Delta'_A)$
- Check $\mathbf{b}^*\Delta_B, \{a_i a_j\}, \{b_i b_j\}$ consistency by $\mathcal{F}_{\mathsf{DVZK}}$
- Check $m_{k,1}$ consistency by $\mathsf{CheckZero}(\langle y \rangle - y)$
- Check $\mathsf{Fix}(\tilde{b}_k)$ consistency by $\mathsf{CheckZero}([y]_B - y)$

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\text{cpre}}$ + DILO-WRK =  $\rightarrow$ : $2\kappa + 3\rho + 2$ bits,  $\rightarrow$ : 2 bits
- How about optimizing one-way communication? Maybe dual execution?

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\text{cpre}}$ + DILO-WRK =  $\rightarrow$ : $2\kappa + 3\rho + 2$ bits,  $\rightarrow$ : 2 bits
- How about optimizing one-way communication? Maybe dual execution?

$\mathbf{x}$ for $\mathcal{I}_{\text{A}}$

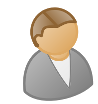
Π_{GC}

$\mathbf{y}$ for $\mathcal{I}_{\text{B}}$

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$

$\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\{z_w\}$ for $w \in \mathcal{O}$



$\mathbf{y}$ for $\mathcal{I}_{\text{B}}$

Π_{GC}

$\mathbf{x}$ for $\mathcal{I}_{\text{A}}$

$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$

$\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

$\{z'_w\}$ for $w \in \mathcal{O}$

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\text{cpre}}$ + DILO-WRK = $\rightarrow$: $2\kappa + 3\rho + 2$ bits, $\rightarrow$: 2 bits
- How about optimizing one-way communication? Maybe dual execution?

$\mathbf{x}$ for $\mathcal{I}_{\mathsf{A}}$ | Π_{GC} | $\mathbf{y}$ for $\mathcal{I}_{\mathsf{B}}$

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$

$\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\{z_w\}$ for $w \in \mathcal{O}$

$\mathbf{y}$ for $\mathcal{I}_{\mathsf{B}}$ | Π_{GC} | $\mathbf{x}$ for $\mathcal{I}_{\mathsf{A}}$

$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$

$\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

$\{z'_w\}$ for $w \in \mathcal{O}$

- Semi-honest GC + DualEx [HEK12, HsV20] : Check $z_w = z'_w$ for $w \in \mathcal{O}$

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\mathsf{cpre}}$ + DILO-WRK = $\to$: $2\kappa + 3\rho + 2$ bits, $\to$: 2 bits
- How about optimizing one-way communication? Maybe dual execution?

x for $\mathcal{I}_\mathsf{A}$ $\quad$ Π_GC $\quad$ **y** for $\mathcal{I}_\mathsf{B}$

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$ $\quad$ $\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\{z_w\}$ for $w \in \mathcal{O}$

y for $\mathcal{I}_\mathsf{B}$ $\quad$ Π_GC $\quad$ **x** for $\mathcal{I}_\mathsf{A}$

$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$ $\quad$ $\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

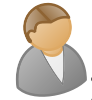
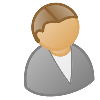
$\{z'_w\}$ for $w \in \mathcal{O}$

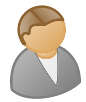
- Semi-honest GC + DualEx [HEK12, HsV20] : Check $z_w = z'_w$ for $w \in \mathcal{O}$

$$\mathcal{A} \text{ may} \begin{cases} \text{garble a different circuit } \mathcal{C}' \\ \text{use different input } \mathbf{x} \text{ or } \mathbf{y} \\ \text{launch selective failure} \end{cases} \Rightarrow$$

- Even if we use a secure $\mathcal{F}_{\mathsf{EQ}}$
- $\mathcal{A}$ can still gain 1-bit leakage

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\mathsf{cpre}}$ + DILO-WRK =  $\rightarrow$ : $2\kappa + 3\rho + 2$ bits,  $\rightarrow$ : 2 bits
- How about optimizing one-way communication? Maybe dual execution?

$\mathcal{F}_{\mathsf{cpre}}$

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}], \Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$ $\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\mathcal{F}_{\mathsf{cpre}}$

$[\mathbf{a}'], [\hat{\mathbf{a}}'], [\mathbf{b}'], [\hat{\mathbf{b}}'], \Delta'_{\mathsf{A}}, \Delta'_{\mathsf{B}} \leftarrow \$$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$ $\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

Optimizing the One-way Communication Via Dual Execution

- Optimized $\mathcal{F}_{\text{cpre}}$ + DILO-WRK = $\rightarrow$: $2\kappa + 3\rho + 2$ bits, $\rightarrow$: 2 bits
- How about optimizing one-way communication? Maybe dual execution?

$\mathcal{F}_{\text{cpre}}$

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}], \Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$ $\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\mathsf{L}_{k,\Lambda_k} = \mathsf{L}_{k,0} \oplus \Lambda_k \cdot \Delta_{\mathsf{A}}$

$\mathcal{F}_{\text{cpre}}$

$[\mathbf{a}'], [\hat{\mathbf{a}}'], [\mathbf{b}'], [\hat{\mathbf{b}}'], \Delta'_{\mathsf{A}}, \Delta'_{\mathsf{B}} \leftarrow \$$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

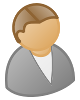
$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$ $\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

$\mathsf{L}'_{k,\Lambda'_k} = \mathsf{L}'_{k,0} \oplus \Lambda'_k \cdot \Delta'_{\mathsf{B}}$

- Color bits and wire masks are authenticated for every wire
- This enables checking equality for every wire across two executions

[HK21] Garbled Sharing

Optimizing the One-way Communication Via Dual Execution

$\mathcal{F}_{\mathsf{cpre}}$

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}], \Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$$

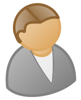
s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

$\{\lambda_k, \mathsf{L}_{k,0}, \mathsf{L}_{k,1}\}$

$\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$

$\mathsf{L}_{k,\Lambda_k} = \mathsf{L}_{k,0} \oplus \Lambda_k \cdot \Delta_{\mathsf{A}}$




$\mathcal{F}_{\mathsf{cpre}}$

$[\mathbf{a}'], [\hat{\mathbf{a}}'], [\mathbf{b}'], [\hat{\mathbf{b}}'], \Delta_{\mathsf{A}}, \Delta_{\mathsf{B}} \leftarrow \$$

s.t. $\hat{a}_k \oplus \hat{b}_k = (a_i \oplus b_i) \cdot (a_j \oplus b_j)$

Π_{DG}

$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$

$\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ for $k \in \mathcal{W}$

$\mathsf{L}'_{k,\Lambda'_k} = \mathsf{L}'_{k,0} \oplus \Lambda'_k \cdot \Delta'_{\mathsf{B}}$

Checks $(a_w \oplus b_w \oplus \Lambda_w) \cdot (\Delta_{\mathsf{A}} \oplus \Delta_{\mathsf{B}}) = (a'_w \oplus b'_w \oplus \Lambda'_w) \cdot (\Delta_{\mathsf{A}} \oplus \Delta_{\mathsf{B}})$

$$V_w^{\mathsf{A}} = (a_w \oplus a'_w \oplus \Lambda'_w)\Delta_{\mathsf{A}} \oplus \mathsf{M}_{\mathsf{A}}[a_w] \oplus \mathsf{M}_{\mathsf{A}}[a'_w] \oplus \mathsf{M}_{\mathsf{A}}[\Lambda'_w] \oplus \mathsf{K}_{\mathsf{A}}[b_w] \oplus \mathsf{K}_{\mathsf{A}}[b'_w] \oplus \mathsf{K}_{\mathsf{A}}[\Lambda_w],$$
$$V_w^{\mathsf{B}} = (b_w \oplus b'_w \oplus \Lambda_w)\Delta_{\mathsf{B}} \oplus \mathsf{M}_{\mathsf{B}}[b_w] \oplus \mathsf{M}_{\mathsf{B}}[b'_w] \oplus \mathsf{M}_{\mathsf{B}}[\Lambda_w] \oplus \mathsf{K}_{\mathsf{B}}[a_w] \oplus \mathsf{K}_{\mathsf{B}}[a'_w] \oplus \mathsf{K}_{\mathsf{B}}[\Lambda'_w].$$

Conclusion

- Further optimization on the compression technique of [DILO22]
- Dual-key authentication and efficient generation
- Dual execution upon distribution garbling eliminates 1-bit leakage
- Malicious 2PC with one-way comm. of $2\kappa + 5$ bits, two way comm. of $2\kappa + 3\rho + 2$ bits

2PC	Rounds		Communication per AND gate	
	Prep.	Online	one-way (bits)	two-way (bits)
Half-gates	1	2	2κ	2κ
HSS-PCG	8	2	$8\kappa + 11$ ($4.04\times$)	$16\kappa + 22$ ($8.09\times$)
KRRW-PCG	4	4	$5\kappa + 7$ ($2.53\times$)	$8\kappa + 14$ ($4.05\times$)
DILO	7	2	$2\kappa + 8\rho + 1$ ($2.25\times$)	$2\kappa + 8\rho + 5$ ($2.27\times$)
This work	8	3	$2\kappa + 5$ ($\approx \mathbf{1}\times$)	$4\kappa + 10$ ($2.04\times$)
This work+DILO	8	2	$2\kappa + 3\rho + 2$ ($1.48\times$)	$2\kappa + 3\rho + 4$ ($\approx \mathbf{1.48}\times$)

Thanks for your listening

Merci beaucoup