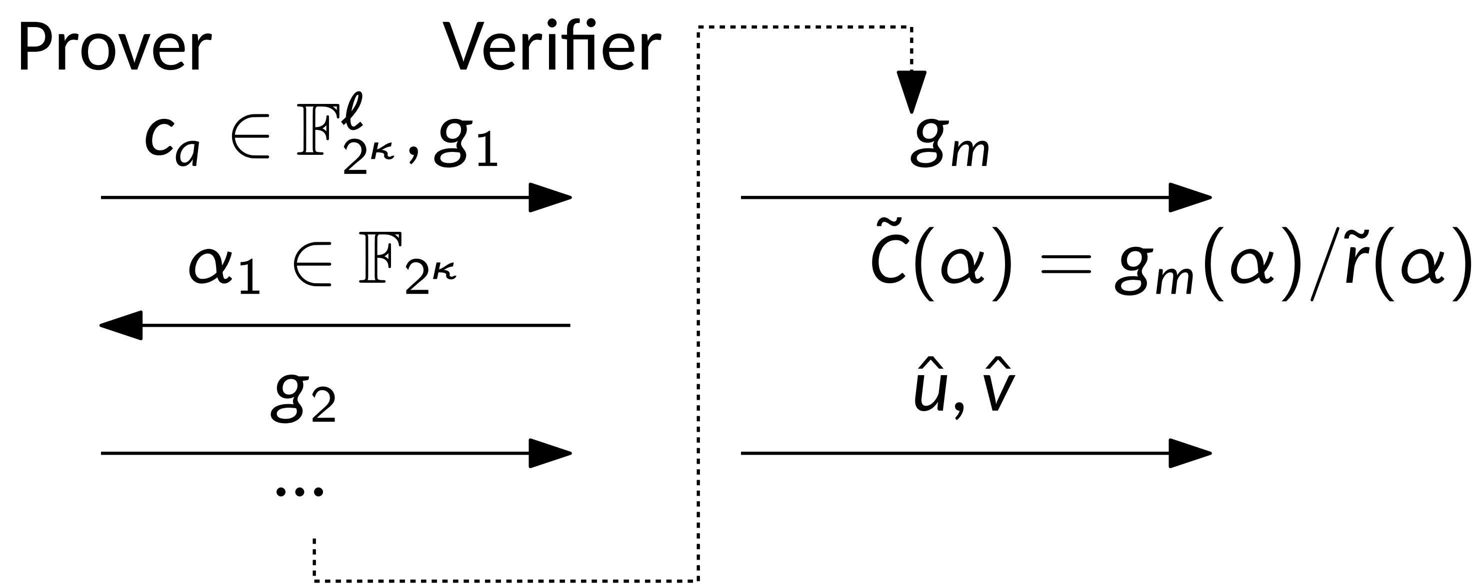
- In QuickSilver, the verifier only needs to access $\mathsf{K}[a], \mathsf{K}[b], \mathsf{K}[c]$
- Where $\mathsf{K}[a] = \mathbf{r}_a^T \cdot \mathsf{K}[\mathbf{x}]$, etc. ($\mathbf{x}$ is the witness)
- Recall that with SoftSpokenOT, $\mathsf{K}[\mathbf{x}] = W' - [0 \| C] \cdot \mathrm{diag}(\Delta)$
- C is the main communication cost (randomness alignment)
- Therefore, we can let the prover send $c_a = \mathbf{r}_a^T \cdot [0 \| C]$ directly and prove its correctness using sumcheck.

Let $\tilde{r}$, $\tilde{C}$ be the multi-linear extension of $\mathbf{r}_a$ and C

$$\sum_{b_1,\ldots,b_m} \tilde{r}(\mathbf{b}) \cdot \tilde{C}(\mathbf{b}) = c_a$$



Accept if $\tilde{W}'(\alpha) - [0 \| \tilde{C}(\alpha)] \cdot \mathrm{diag}(\Delta) = \hat{v} + [1 \ldots 1] \cdot \hat{u} \cdot \mathrm{diag}(\Delta)$

Background on Constant Round 2PC

- Steady improvement in the semi-honest world

Textbook [Yao86]	P&P [BMR90]	GRR3 [NPS99]	GRR2 [PSSW90]	Free-XOR [KS08]	FleXOR [KMR14]	Half-Gates [ZRE15]	Three-Halves [RR21]
XOR: 8κ AND: 8κ	XOR: 4κ AND: 4κ	XOR: 3κ AND: 3κ	XOR: 2κ AND: 2κ	XOR: 0 AND: 3κ	$\{0, 1, 2\}\kappa$	2κ	$1.5\kappa + 5$

Background on Constant Round 2PC

- Steady improvement in the semi-honest world

Textbook [Yao86]	P&P [BMR90]	GRR3 [NPS99]	GRR2 [PSSW90]	Free-XOR [KS08]	FleXOR [KMR14]	Half-Gates [ZRE15]	Three-Halves [RR21]
XOR: 8κ AND: 8κ	XOR: 4κ AND: 4κ	XOR: 3κ AND: 3κ	XOR: 2κ AND: 2κ	XOR: 0 AND: 3κ	$\{0, 1, 2\}\kappa$	2κ	$1.5\kappa + 5$

- What about the malicious world?

Cut-and-Choose [LP07,NO09,HKE13,NST17,...]

$O(\rho\kappa)$ or $O(\frac{\rho\kappa}{\log C})$

Background on Constant Round 2PC

- Steady improvement in the semi-honest world

Textbook [Yao86]	P&P [BMR90]	GRR3 [NPS99]	GRR2 [PSSW90]	Free-XOR [KS08]	FleXOR [KMR14]	Half-Gates [ZRE15]	Three-Halves [RR21]
XOR: 8κ AND: 8κ	XOR: 4κ AND: 4κ	XOR: 3κ AND: 3κ	XOR: 2κ AND: 2κ	XOR: 0 AND: 3κ	$\{0,1,2\}\kappa$	2κ	$1.5\kappa + 5$

- What about the malicious world?

Cut-and-Choose [LP07,NO09,HKE13,NST17,...]	Authenticated Garbling [WRK17,KRRW18]
$O(\rho\kappa)$ or $O(\frac{\rho\kappa}{\log C})$	Π_{pre}: $13\kappa + 8\rho$ Π_{online}: $2\kappa + 1$

Π_{pre} TinyOT* $+$ Π_{online} Checking $=$ Actively-secure constant-round 2PC

Background on Constant Round 2PC

- Steady improvement in the semi-honest world

Textbook [Yao86]	P&P [BMR90]	GRR3 [NPS99]	GRR2 [PSSW90]	Free-XOR [KS08]	FleXOR [KMR14]	Half-Gates [ZRE15]	Three-Halves [RR21]
XOR: 8κ AND: 8κ	XOR: 4κ AND: 4κ	XOR: 3κ AND: 3κ	XOR: 2κ AND: 2κ	XOR: 0 AND: 3κ	$\{0,1,2\}\kappa$	2κ	$1.5\kappa + 5$

- What about the malicious world?

Cut-and-Choose [LP07,NO09,HKE13,NST17,...]	Authenticated Garbling [WRK17,KRRW18]	PCGs [BCG+19, YWL+20, CRR21,...]
$O(\rho\kappa)$ or $O(\frac{\rho\kappa}{\log C})$	Π_{pre}: $13\kappa + 8\rho$ Π_{online}: $2\kappa + 1$	

Π_{pre} TinyOT* $+$ Π_{online} Checking $=$ Actively-secure constant-round 2PC

Background on Constant Round 2PC

- Steady improvement in the semi-honest world

Textbook [Yao86]	P&P [BMR90]	GRR3 [NPS99]	GRR2 [PSSW90]	Free-XOR [KS08]	FleXOR [KMR14]	Half-Gates [ZRE15]	Three-Halves [RR21]
XOR: 8κ AND: 8κ	XOR: 4κ AND: 4κ	XOR: 3κ AND: 3κ	XOR: 2κ AND: 2κ	XOR: 0 AND: 3κ	$\{0,1,2\}\kappa$	2κ	$1.5\kappa + 5$

- What about the malicious world?

Cut-and-Choose [LP07,NO09,HKE13,NST17,...]	Authenticated Garbling [WRK17,KRRW18]	PCGs [BCG+19, YWL+20, CRR21,...]	AG from PCG [DILO22]
$O(\rho\kappa)$ or $O(\frac{\rho\kappa}{\log C})$	Π_{pre}: $13\kappa + 8\rho$ Π_{online}: $2\kappa + 1$		$\mathcal{F}_{\mathrm{VOLE}}$-hyb. $2\kappa + 8\rho$ $\mathcal{F}_{\mathrm{DAMT}}$-hyb. $2\kappa + 4\rho$

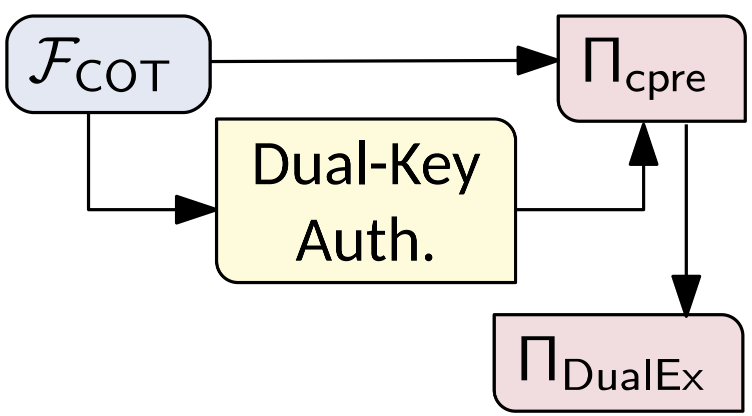
Π_{pre} TinyOT* $+$ Π_{online} Checking $=$ Actively-secure constant-round 2PC

Can we close the gap?

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

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

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	$L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$

- selective-failure on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble different logic $\Rightarrow$ Add 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	$L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
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	$L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$

- selective-failure on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble different logic $\Rightarrow$ Add IT-MAC, equality check, etc.
- We need preprocessing information to complete garbling

$$\mathbf{a}_{\Delta_B} \oplus \mathbf{a}_{\Delta_B} = \mathbf{a} \cdot \Delta_B$$

$\mathbf{a}, \hat{\mathbf{a}}, \Delta_A$

$\mathbf{a}$ $\hat{\mathbf{a}}$ $\mathbf{b}$ $\hat{\mathbf{b}}$

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

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

Δ_A, Δ_B

$\hat{a}_k \oplus \hat{b}_k = \lambda_i \cdot \lambda_j$ for $(i, j, k, \wedge)$

$\mathbf{b}, \hat{\mathbf{b}}, \Delta_B$

$\mathbf{a}$ $\hat{\mathbf{a}}$ $\mathbf{b}$ $\hat{\mathbf{b}}$

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

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	$L_{k,0} \oplus (\bar{\lambda}_i \cdot \bar{\lambda}_j \oplus \lambda_k)\Delta_A$

- selective-failure on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble different logic $\Rightarrow$ Add IT-MAC, equality check, etc.
- We need preprocessing information to complete garbling

$\mathbf{a}$ Δ_B $\oplus$ $\mathbf{a}$ Δ_B $= \mathbf{a} \cdot \Delta_B$

$\mathbf{a}, \hat{\mathbf{a}}, \Delta_A$

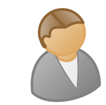
$\mathbf{a}$ $\hat{\mathbf{a}}$ $\mathbf{b}$ $\hat{\mathbf{b}}$

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

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

Δ_A, Δ_B

$\hat{a}_k \oplus \hat{b}_k = \lambda_i \cdot \lambda_j$ for $(i, j, k, \wedge)$



$\mathbf{b}, \hat{\mathbf{b}}, \Delta_B$

$\mathbf{a}$ $\hat{\mathbf{a}}$ $\mathbf{b}$ $\hat{\mathbf{b}}$

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	$L_{k,0} \oplus K[\Lambda_{11}]$	$M[\Lambda_{11}]$

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

Free-XOR GC $\Rightarrow$

$|\Delta_A| = \kappa \approx 128$

Authenticated Garbling = Distributed Garbling + Checking

controls garbling so it can

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

- selective-failure on $\Lambda := z \oplus \lambda \Rightarrow$ Secret share $\lambda := a \oplus b$
- garble different logic $\Rightarrow$ Add IT-MAC, equality check, etc.
- We need preprocessing information to complete garbling

$$\mathbf{a}\ \Delta_B \oplus \mathbf{a}\ \Delta_B = \mathbf{a} \cdot \Delta_B$$

$\mathbf{a}, \hat{\mathbf{a}}, \Delta_A$

a **â** **b** $\hat{\mathbf{b}}$

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

Δ_A, Δ_B

$\hat{a}_k \oplus \hat{b}_k = \lambda_i \cdot \lambda_j$ for $(i, j, k, \wedge)$

$\mathbf{b}, \hat{\mathbf{b}}, \Delta_B$

a **â** **b** $\hat{\mathbf{b}}$

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 $\Rightarrow$

$|\Delta_A| = \kappa \approx 128$

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

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

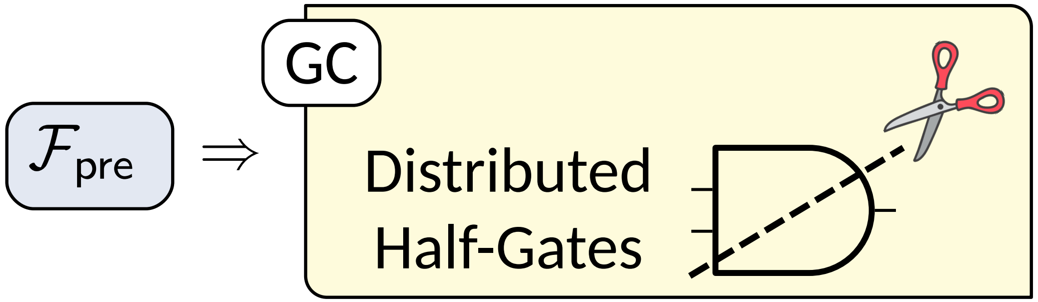
Λ_i	Λ_j	Alice's AuthGC	Bob's AuthGC
0	0	$\mathsf{L}_{k,0} \oplus \mathsf{M}[\Lambda_{00}]$	$\mathsf{K}[\Lambda_{00}]$
0	1	$\mathsf{L}_{k,0} \oplus \mathsf{M}[\Lambda_{01}]$	$\mathsf{K}[\Lambda_{01}]$
1	0	$\mathsf{L}_{k,0} \oplus \mathsf{M}[\Lambda_{10}]$	$\mathsf{K}[\Lambda_{10}]$
1	1	$\mathsf{L}_{k,0}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

IT-MAC Soundness $\Rightarrow$

$|\Delta_B| = \rho \approx 40$

KRRW18: Distributed Half-Gates Garbling + Equality Checking

- Distributed half-gates garbling is fully compatible with $\boxed{\mathcal{F}_{\text{pre}}}$



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

4κ bits/AND WRK17	$\Rightarrow$	$2\kappa + 1$ bits/AND KRRW18

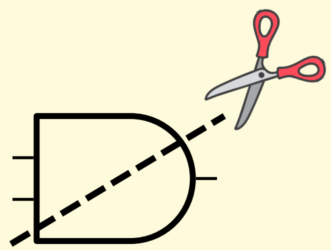
KRRW18: Distributed Half-Gates Garbling + Equality Checking

- Distributed half-gates garbling is fully compatible with $\boxed{\mathcal{F}_{\text{pre}}}$

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

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

Distributed Half-Gates



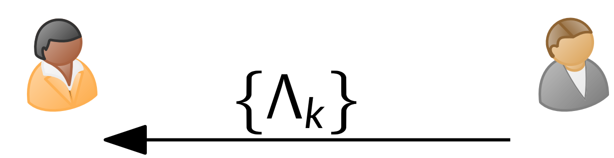
4κ bits/AND	$\Rightarrow$	$2\kappa + 1$ bits/AND
WRK17		KRRW18

- **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:

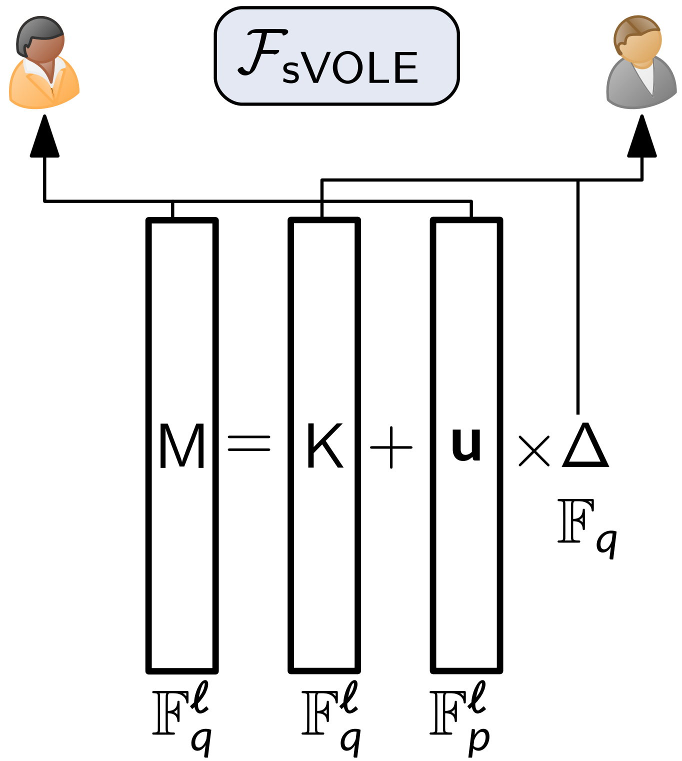
- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.



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

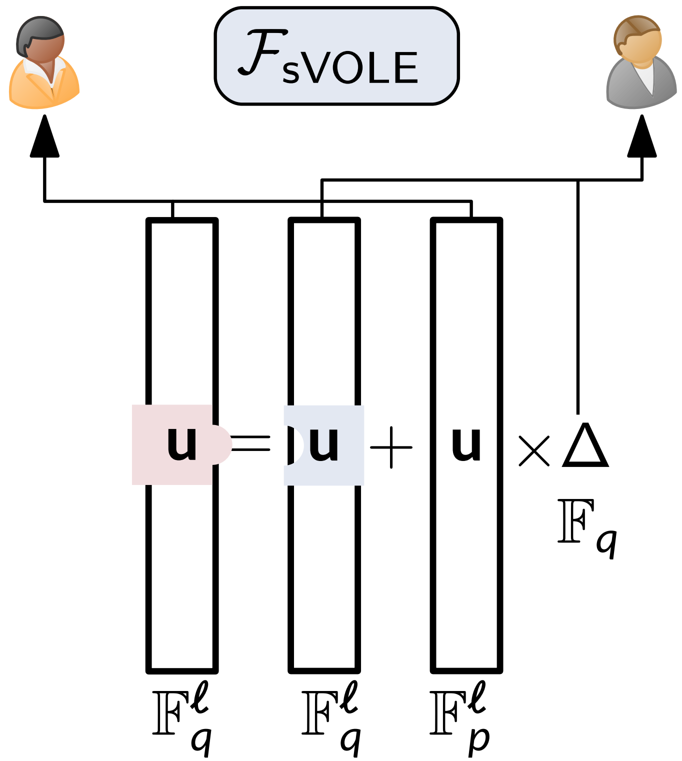
4ρ bits/AND	$\Rightarrow$	0 bits/AND
WRK17		KRRW18

Efficient COT/sVOLE 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 $\mathcal{F}_{\mathsf{COT}}$

Efficient COT/sVOLE and Designated Verifier Zero Knowledge



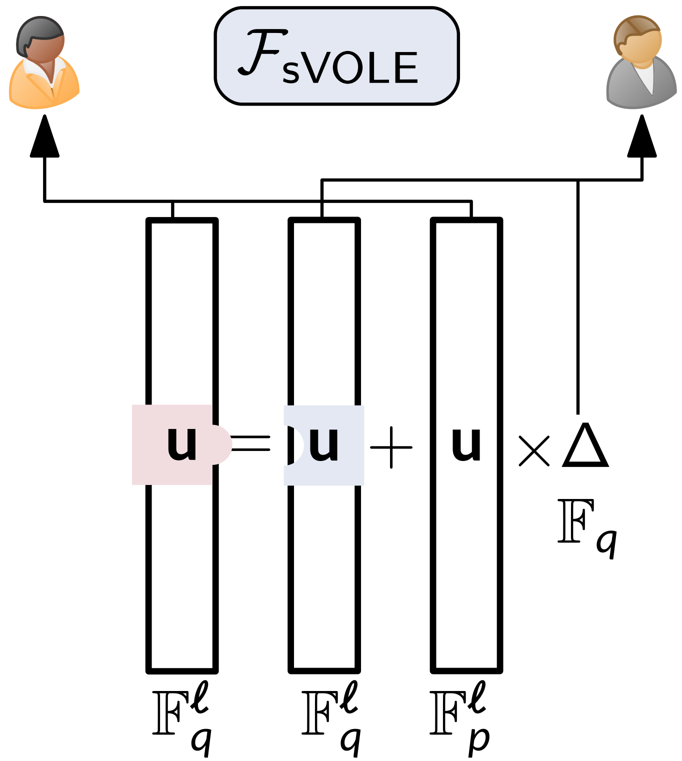
- Efficient protocol for $\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\text{sVOLE}}$ as $\mathcal{F}_{\text{COT}}$

Derandomization operation: Fix

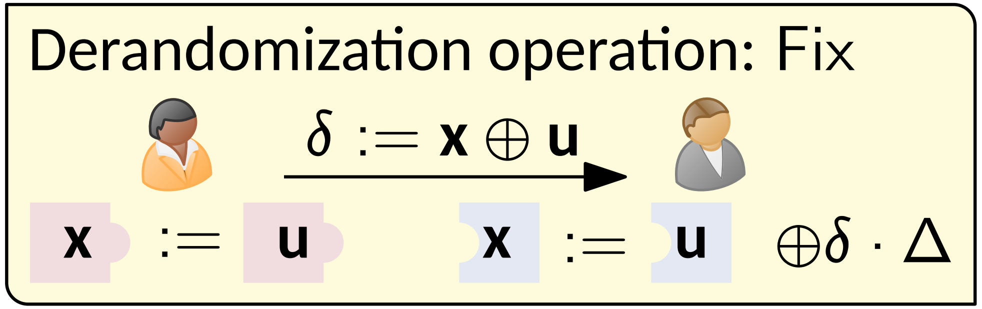
$\delta := \mathbf{x} \oplus \mathbf{u}$

$\mathbf{x} := \mathbf{u}$ $\quad\quad$ $\mathbf{x} := \mathbf{u} \oplus \delta \cdot \Delta$

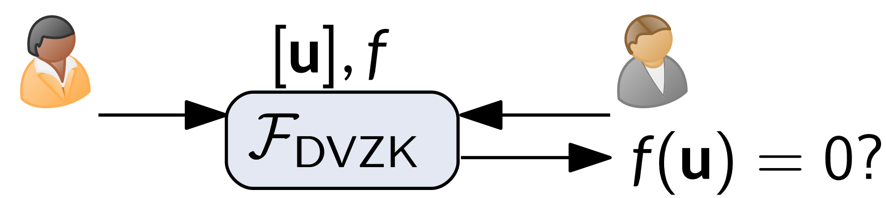
Efficient COT/sVOLE and Designated Verifier Zero Knowledge



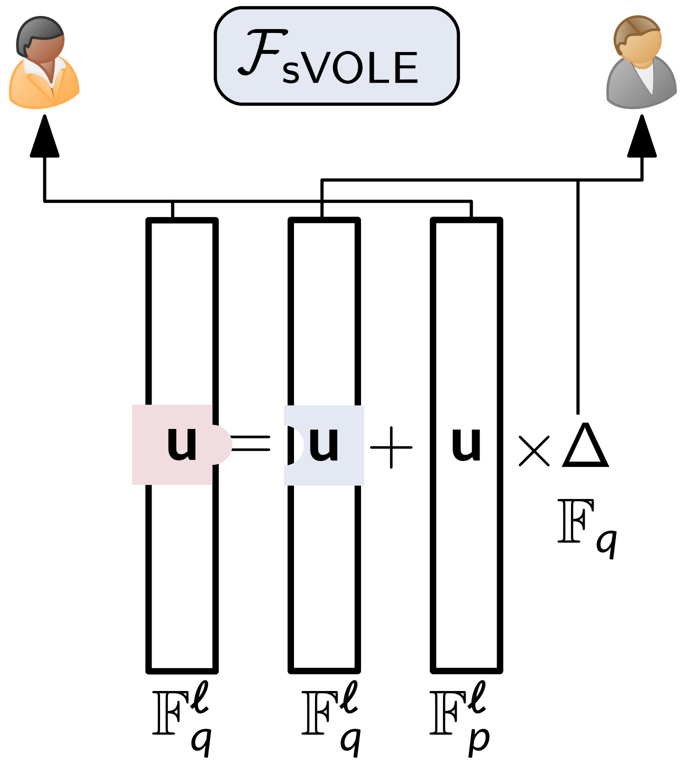
- Efficient protocol for $\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\mathsf{sVOLE}}$ as $\mathcal{F}_{\mathsf{COT}}$



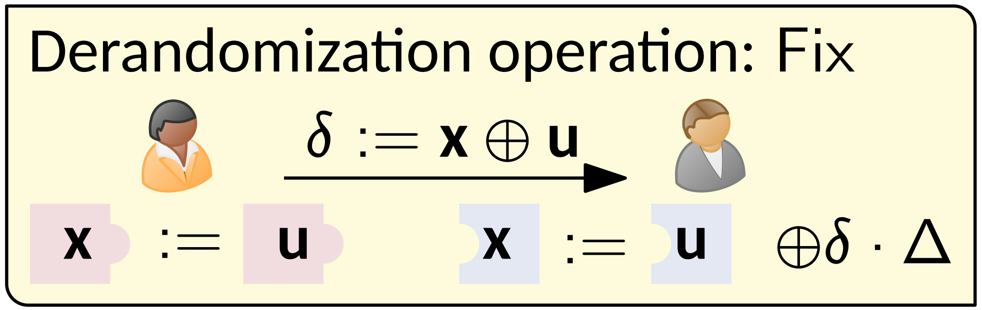
- Efficient proof for deg-d relations on $\mathbf{u}$ [DIO21, YSWW21, ...]



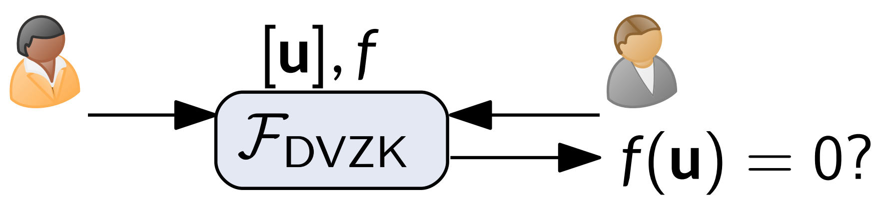
Efficient COT/sVOLE and Designated Verifier Zero Knowledge



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

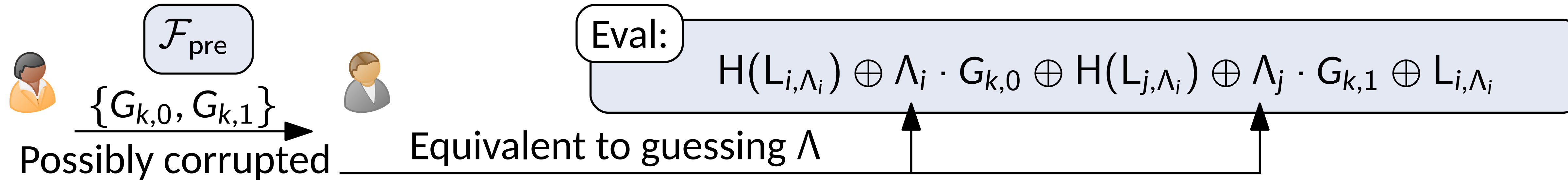


- Efficient proof for deg-d relations on $\mathbf{u}$ [DIO21, YSWW21, ...]



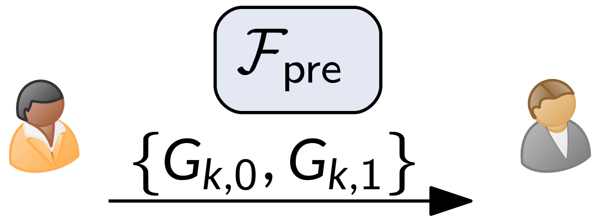
- In DILO, those PCG correlations are called "simple correlations"
- Unfortunately, we still don't have an efficient direct $\mathcal{F}_{\mathsf{pre}}$ PCG construction
- The closest is the $\mathcal{F}_{\mathsf{DAMT}}$ correlation generated from Ring-LPN, but with ρ-time overhead

Prior Art: DILO



- 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**

Prior Art: DILO



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

Possibly corrupted

Equivalent to guessing Λ

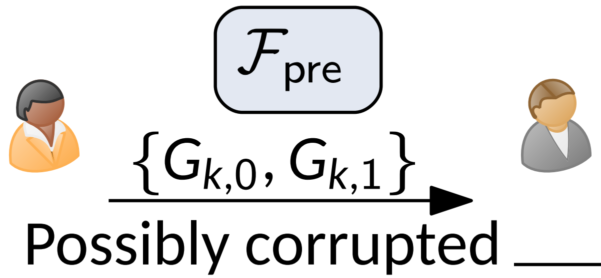
- Garbler can only guess once
- If **b** is uniformly random, then guessing leaks no information
- If #Guess is too large, then abort happens overwhelmingly, if #Guess is too little, then we don't require much entropy from **b**

DILO Oberservation 1

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

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

Prior Art: DILO



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

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

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

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

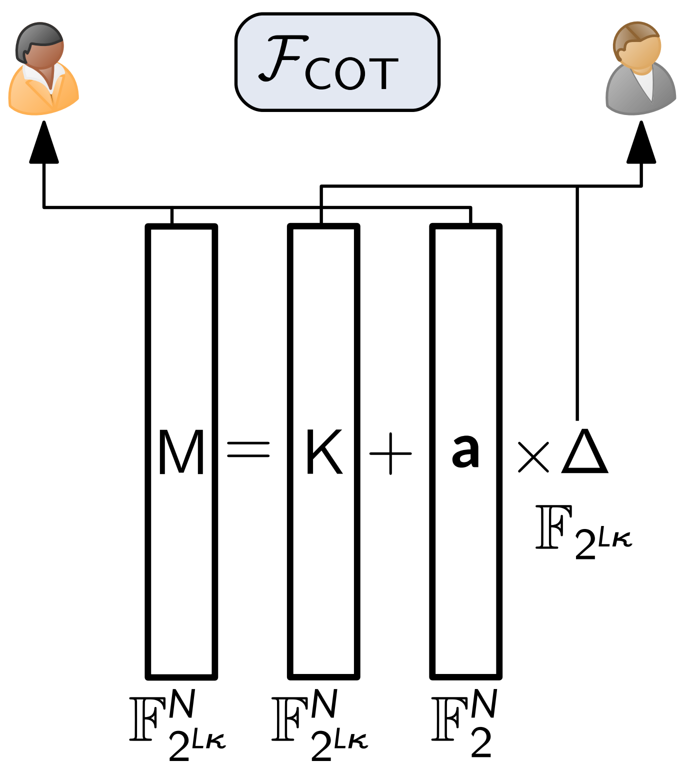
$\mathcal{F}_{\text{pre}}$ samples $[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}]$, Δ_A, Δ_B

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

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

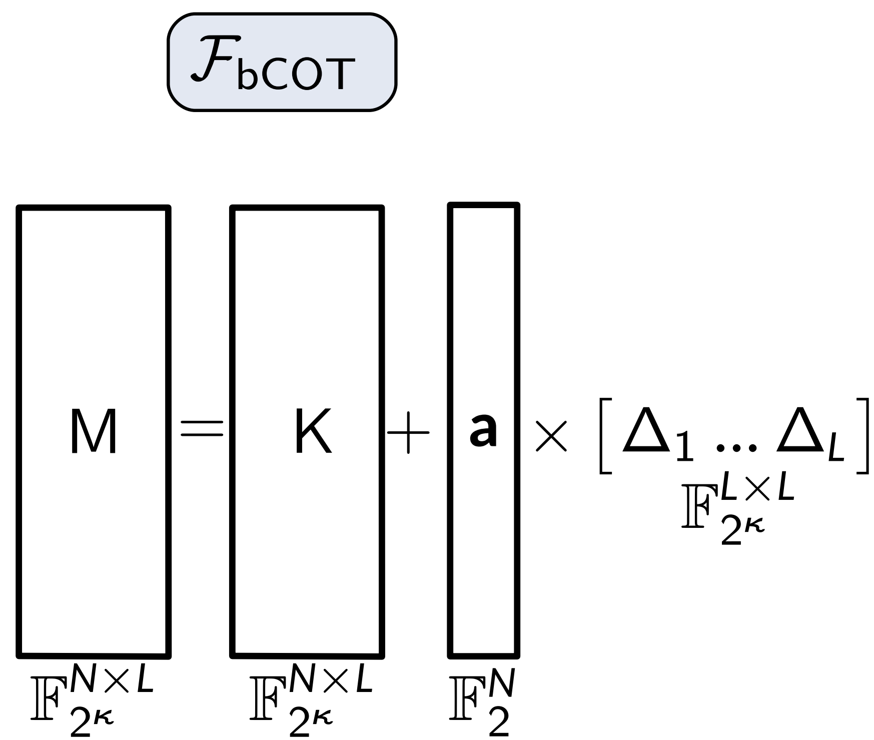
suffices to compute $\mathbf{a} \otimes \mathbf{b}^*$

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



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

$\Leftrightarrow$



DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Computing $\hat{b}_k$

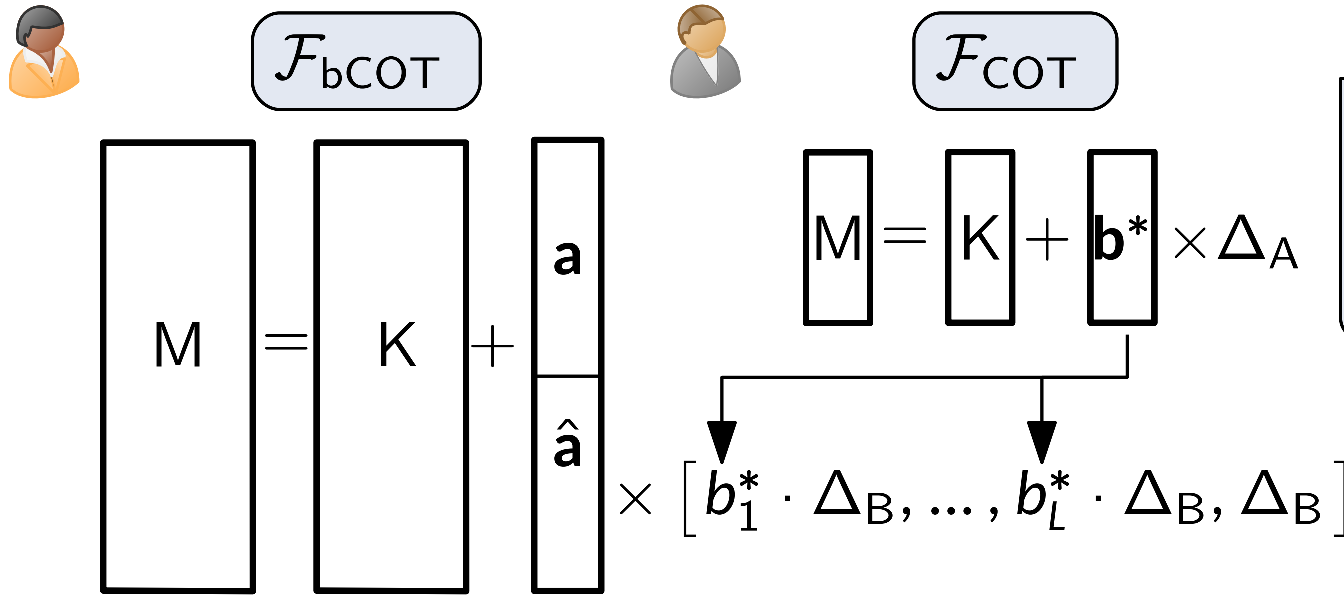
$\boxed{\mathcal{F}_{\text{bCOT}}}$

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

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

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

DILO Implementation of $\mathcal{F}_{\mathrm{cpre}}$: Computing $\hat{b}_k$



- $[\mathbf{a}]_{b_j^* \Delta_{\mathrm{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\mathrm{B}}}$
- By linearity on IT-MAC, we can get $[a_i b_j]_{\Delta_{\mathrm{B}}}$ for any i, j

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Computing $\hat{b}_k$

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

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

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

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

- $[\mathbf{a}]_{b_j^* \Delta_{\text{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\text{B}}}$
- By linearity on IT-MAC, we can get $[a_i b_j]_{\Delta_{\text{B}}}$ for any i, j

---

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

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

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

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

DILO Implementation of $\mathcal{F}_{\mathsf{cpre}}$: Computing $\hat{b}_k$

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

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

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

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

- $[\mathbf{a}]_{b_j^* \Delta_{\mathsf{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\mathsf{B}}}$
- By linearity on IT-MAC, we can get $[a_i b_j]_{\Delta_{\mathsf{B}}}$ for any i, j

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

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

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

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

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

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Computing $\hat{b}_k$

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

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

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

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

- $[\mathbf{a}]_{b_j^* \Delta_{\mathrm{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\mathrm{B}}}$
- By linearity on IT-MAC, we can get $[a_i b_j]_{\Delta_{\mathrm{B}}}$ for any i, j

---

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

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

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

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

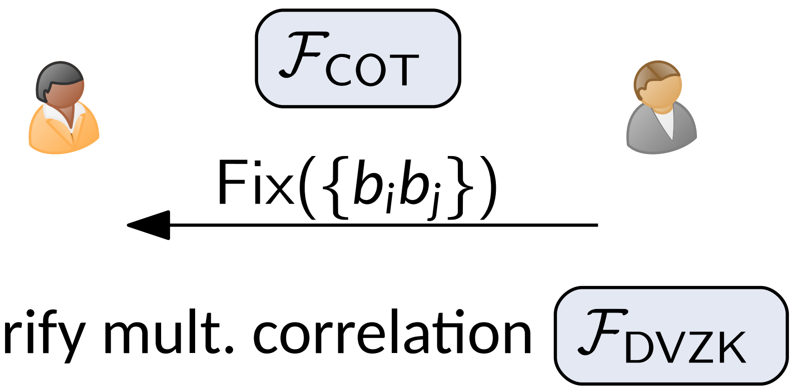
Define $[\tilde{b}_k]_{\Delta_{\mathrm{B}}} := [\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i]_{\Delta_{\mathrm{B}}}$

$\xrightarrow{m_{k,1} := \tilde{b}_k}$

- $\tilde{b}_k := (\tilde{b}_k \oplus \tilde{b}_k) \cdot \Delta_{\mathrm{B}}^{-1}$
- $\hat{b}_k = b_i b_j \oplus \tilde{b}_k$

DILO Implementation of $\mathcal{F}_{\mathrm{cpre}}$: Authenticating $\hat{b}_k$ (Under Δ_{A})

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



- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\tilde{b}_k \oplus \tilde{b}_k = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_{\mathrm{A}}$

DILO Implementation of $\mathcal{F}_{\mathsf{cpre}}$: Authenticating $\hat{b}_k$ (Under Δ_A)

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

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

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

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

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

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

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

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

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

DILO Implementation of $\mathcal{F}_{\text{cpre}}$: Authenticating $\hat{b}_k$ (Under Δ_A)

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

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

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

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

- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\tilde{b}_k \oplus \tilde{b}_k = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_A$

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

$$\xrightarrow{\text{Fix}\begin{pmatrix}\{a_i a_j \Delta_A\} \\ \{\hat{a}_k \Delta_A\} \\ \mathbf{a}\Delta_A\end{pmatrix}}$$

$\xrightarrow{\text{Generate mask } \hat{a}_{k,2} \in \mathbb{F}_{2^\rho}}$

Locally comptue $[v_k]_{\Delta_B} := [\tilde{b}_k \cdot \Delta_A \oplus \hat{a}_{k,2}]_{\Delta_B}$

$\xrightarrow{m_{k,2} := \tilde{b}_k}$

- $\tilde{b}_k := (\tilde{b}_k \oplus \tilde{b}_k) \cdot \Delta_B^{-1}$
- $\hat{b}_k = \tilde{b}_k \oplus b_i b_j$

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.

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

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.

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

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.

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

DILO-WRK Check

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

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

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.

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

DILO-WRK Check

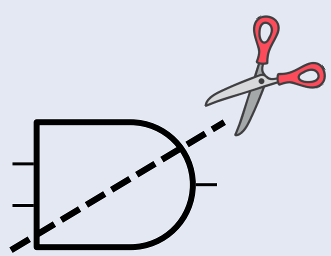
$$\Lambda_k \cdot \Delta_B := \lambda_k \cdot \Delta_B \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_B$$

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

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

GC

Distributed Half-Gates



2κ bits/AND

$+$

AuthGC

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

3ρ bits/AND

$= 2\kappa + 3\rho$ bits/AND

Optimizing the Compressed Preprocessing Protocol

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

1 bit

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

$\rho + 1$ bits

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

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

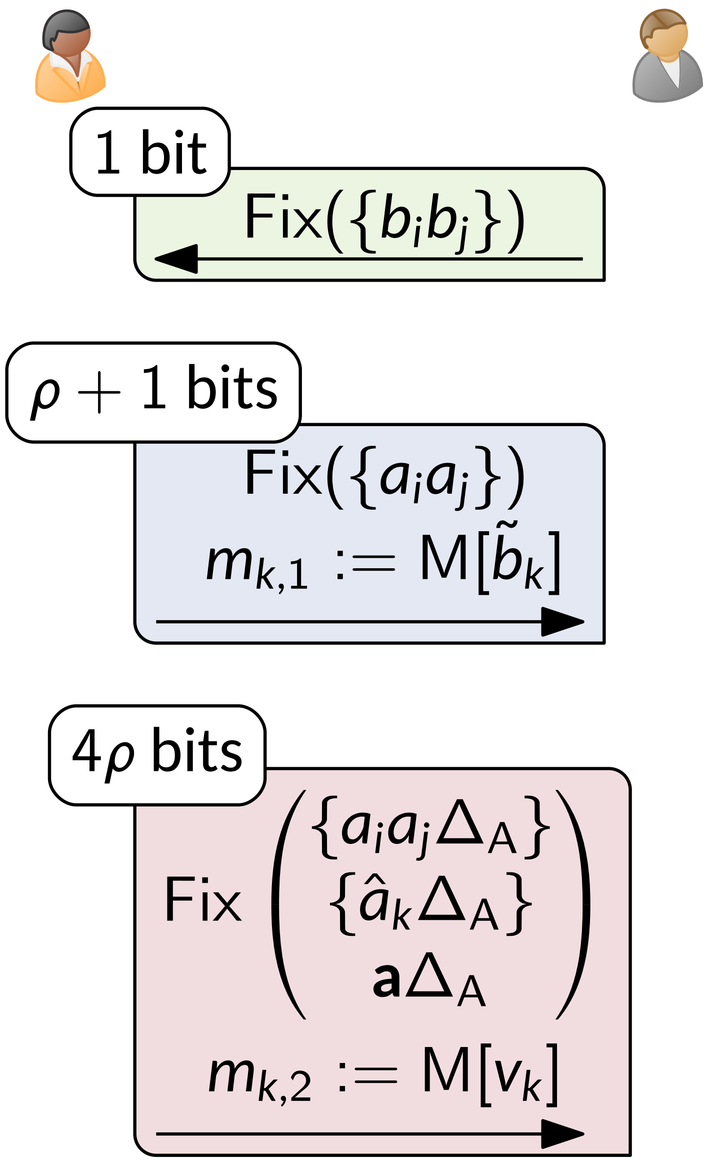
4ρ bits

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

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

Optimizing the Compressed Preprocessing Protocol

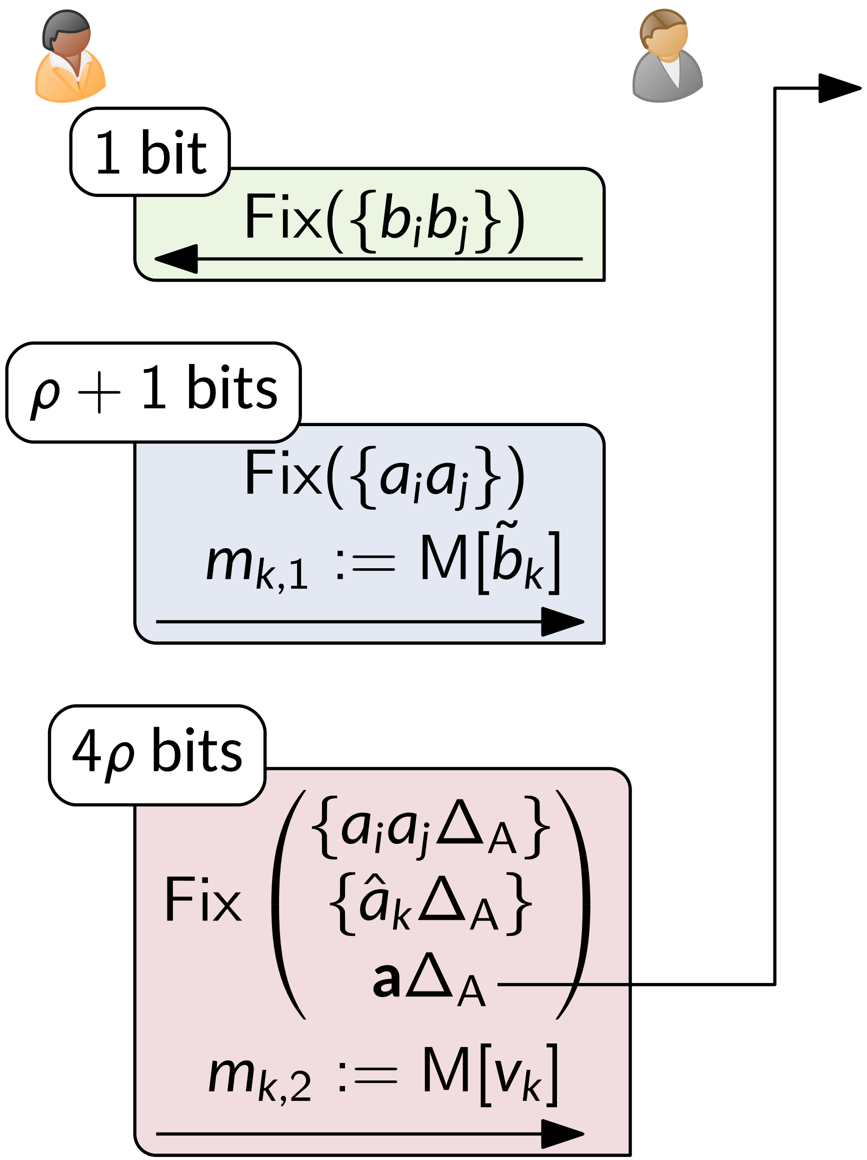
The overhead of DILO is
$5\rho + 2$ bits per AND gate



- 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

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

  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 $\mathsf{Fix}()$ input

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

- 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 $\mathsf{Fix}()$ input

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

Dual Key Authentication

- 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

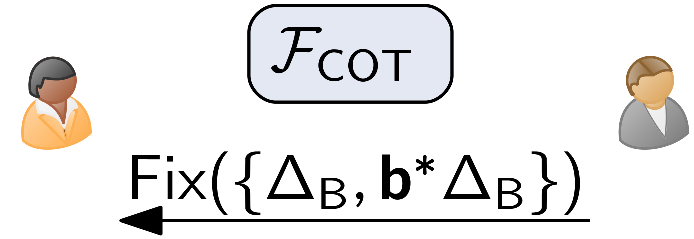
Our goal is to generate $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

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 $\mathbf{b}$ in $\mathcal{F}_{\mathsf{bCOT}}$ global keys

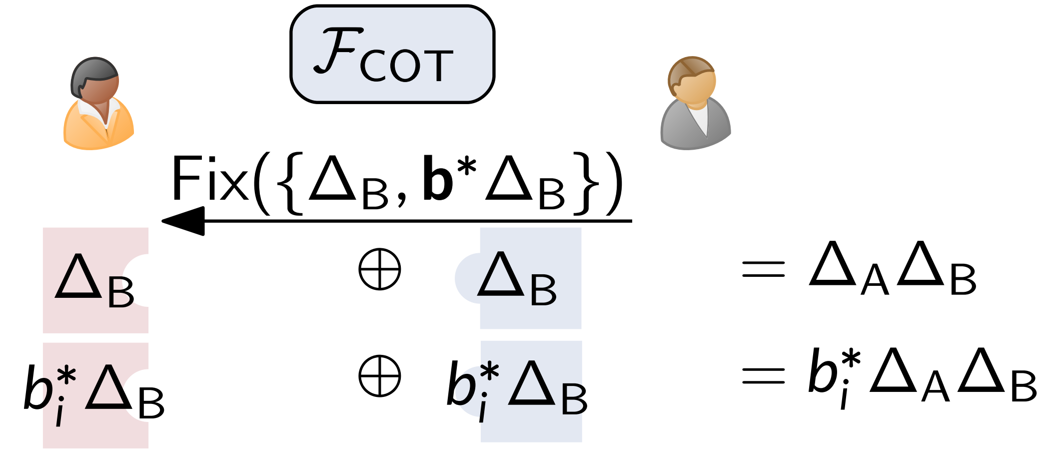


Optimizing the Compressed Preprocessing Protocol, Continued

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

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

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

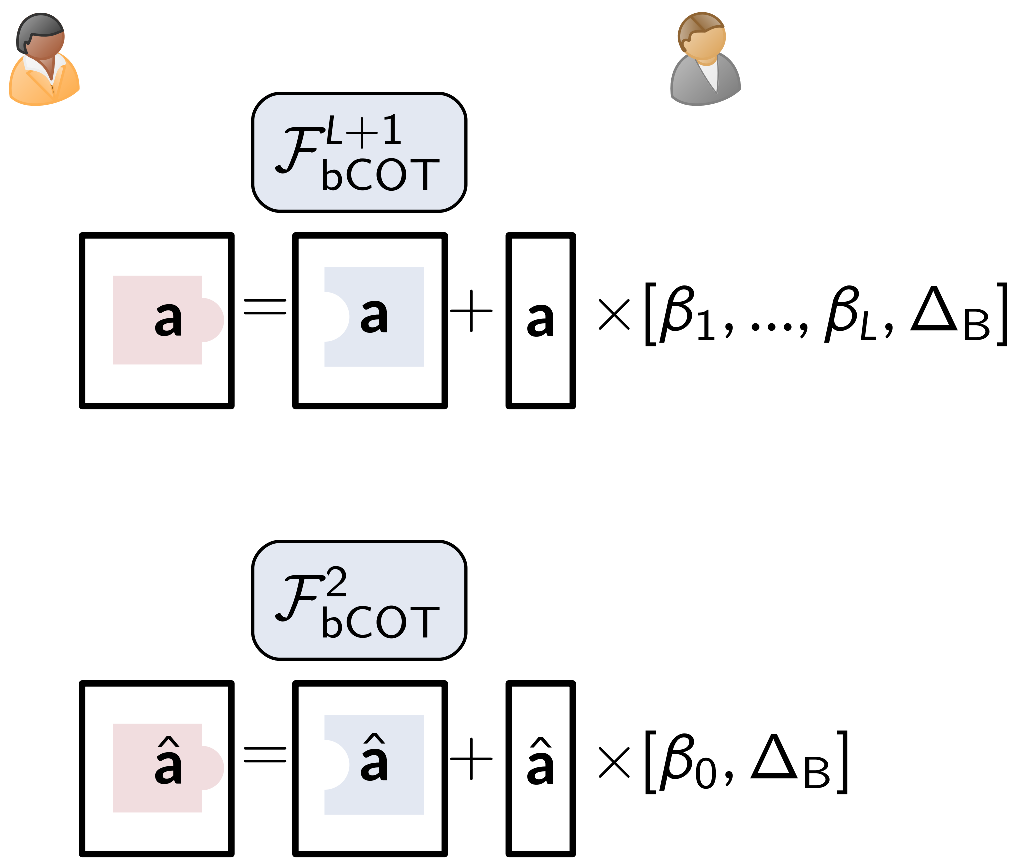


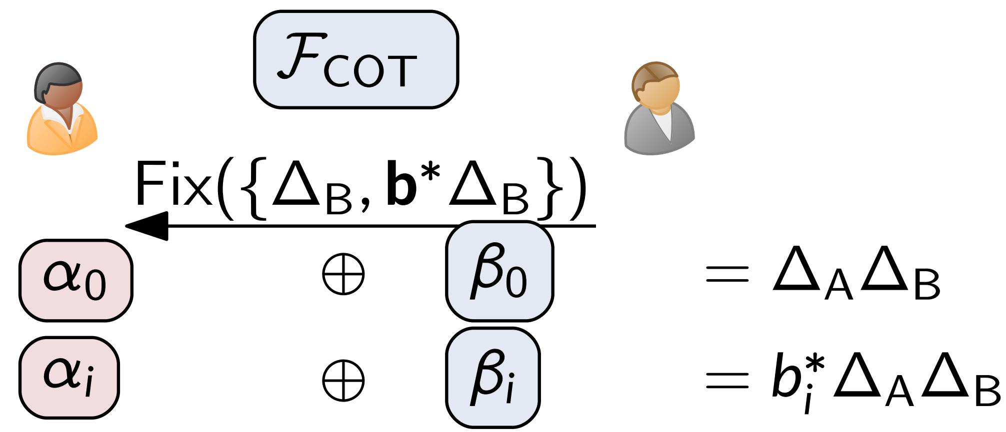
Optimizing the Compressed Preprocessing Protocol, Continued

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

- $D_A[\hat{a}_k] \oplus D_B[\hat{a}_k] = \hat{a}_k \Delta_A \Delta_B$
- $D_A[a_i b_j] \oplus D_B[a_i b_j] = a_i b_j \Delta_A \Delta_B$

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





Optimizing the Compressed Preprocessing Protocol, Continued

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

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

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

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

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

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

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

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

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

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

$\Leftrightarrow$

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

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

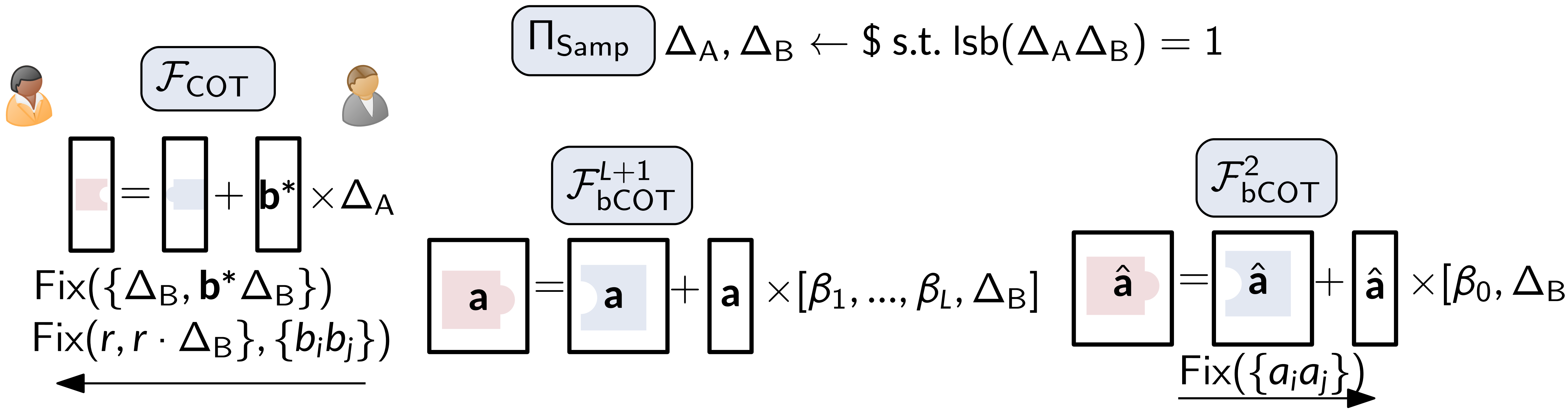
$$\mathsf{K}[\beta_0] \oplus \beta_0 \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_0]$$
$$\mathsf{K}[\beta_i] \oplus \beta_i \cdot \Delta'_\mathsf{A} = \mathsf{M}[\beta_i]$$

[DIO21] gives a modular way of proving equality under independent keys

Optimizing the Compressed Preprocessing Protocol, Completed

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

Optimizing the Compressed Preprocessing Protocol, Completed



Optimizing the Compressed Preprocessing Protocol, Completed

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

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

$\square = \square + \mathbf{b}^* \times \Delta_A$

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

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

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

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

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

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

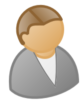
$\mathrm{Fix}(\{a_i a_j\})$

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

Optimizing the Compressed Preprocessing Protocol, Completed

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

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



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

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

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

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

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

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

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

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

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

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

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

Optimizing the Compressed Preprocessing Protocol, Completed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Optimizing the Compressed Preprocessing Protocol, Completed

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

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

$[\,] = [\,] + [\mathbf{b}^*] \times \Delta_{\mathsf{A}}$

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

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

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

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

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

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

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

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

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

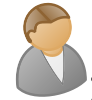
$\tilde{b}_k = m_{k,1} \oplus \mathsf{lsb}(\mathsf{D}_{\mathsf{A}}[\tilde{b}_k])$

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

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

- Check $\{\beta_i\}$ consistency by $\mathsf{Fix}(\Delta'_{\mathsf{A}})$
- Check $\mathbf{b}^*\Delta_{\mathsf{B}}, \{a_i a_j\}, \{b_i b_j\}$ 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 $\mathsf{CheckZero}([y] - y)$

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 $\mathcal{I}_{\mathsf{A}}$ $\quad$ Π_{GC} $\quad$ $\mathbf{y}$ 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 $w \in \mathcal{O}$

$\mathbf{y}$ for $\mathcal{I}_{\mathsf{B}}$ $\quad$ Π_{GC} $\quad$ $\mathbf{x}$ 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 $w \in \mathcal{O}$

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?

x for $\mathcal{I}_{\text{A}}$ Π_{GC} **y** for $\mathcal{I}_{\text{B}}$

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

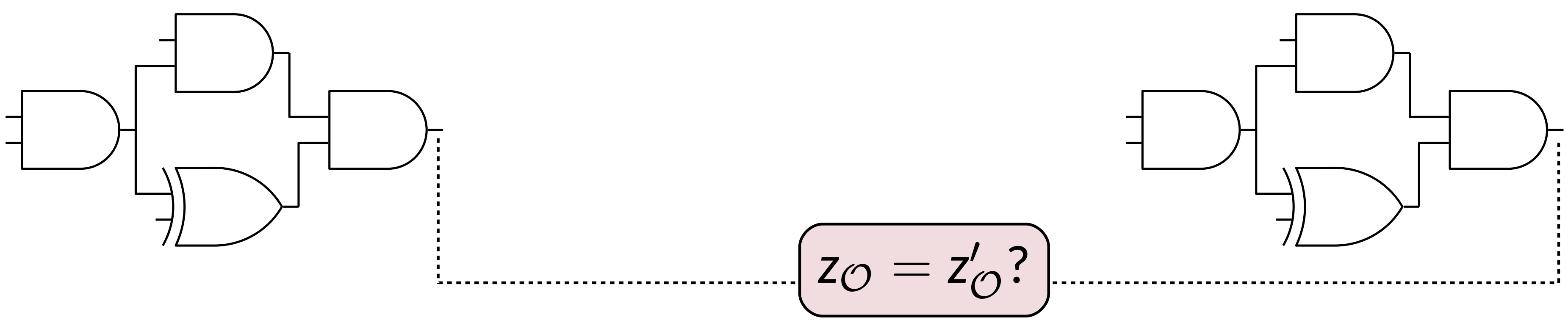
$\{z_w\}$ for $w \in \mathcal{O}$

y for $\mathcal{I}_{\text{B}}$ Π_{GC} **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 $w \in \mathcal{O}$

- [HEK12, HsV20]: Check for equality in circuit outputs



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?

x for $\mathcal{I}_A$ Π_{GC} **y** for $\mathcal{I}_B$

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

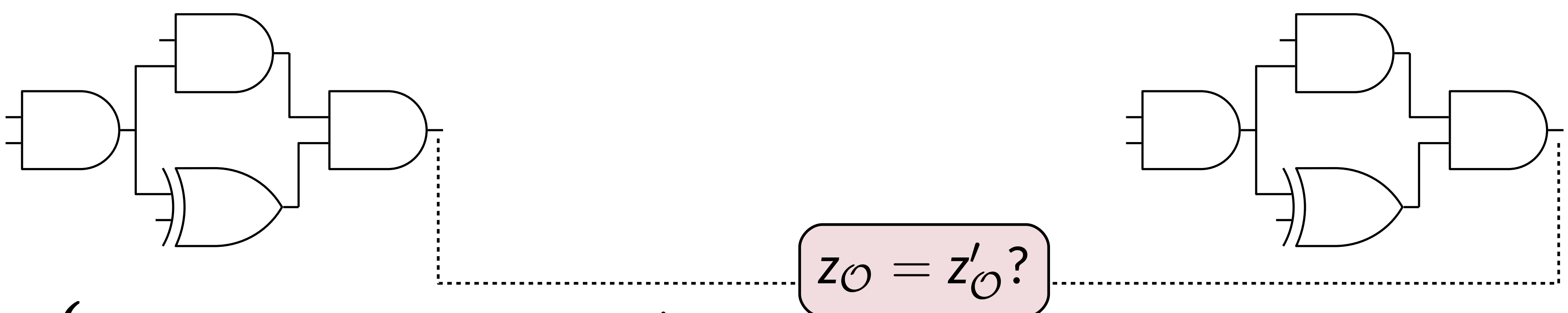
$\{\Lambda_k, \mathsf{L}_{k,\Lambda_k}\}$ for $k \in \mathcal{W}$
$\{z_w\}$ for $w \in \mathcal{O}$

y for $\mathcal{I}_B$ Π_{GC} **x** for $\mathcal{I}_A$

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

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

- [HEK12, HsV20]: Check for equality in circuit outputs



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

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

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 $k \in \mathcal{W}$

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

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

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

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

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

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}

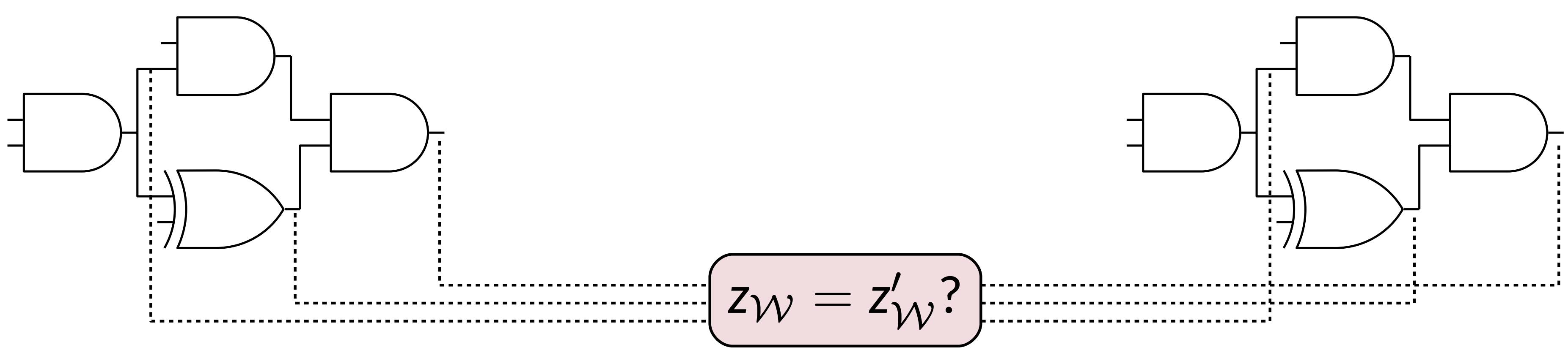
$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$

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

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

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

[HK21] Garbled Sharing



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 $2\kappa + 3\rho + 4$ 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×)	$16\kappa + 22$ (8.09×)
KRRW-PCG	4	4	$5\kappa + 7$ (2.53×)	$8\kappa + 14$ (4.05×)
DILO	7	2	$2\kappa + 8\rho + 1$ (2.25×)	$2\kappa + 8\rho + 5$ (2.27×)
This work	8	3	$2\kappa + 5$ ($\approx$ **1**×)	$4\kappa + 10$ (2.04×)
This work+DILO	8	2	$2\kappa + 3\rho + 2$ (1.48×)	$2\kappa + 3\rho + 4$ ($\approx$ **1.48**×)

Thanks for your listening

Merci beaucoup