Actively Secure Half-Gates with Minimum Overhead under Duplex Networks

Hongrui Cui
Shanghai Jiao Tong University

Xiao Wang
Northwestern University

Kang Yang
State Key Laboratory of Cryptology

Yu Yu
Shanghai Jiao Tong University
Shanghai Qi Zhi Institute

Apr. 24, 2023 · Eurocrypt 2023

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}: $2\kappa + 1$		$\mathcal{F}_{\text{VOLE}}$-hyb. $2\kappa + 8\rho$ $\mathcal{F}_{\text{DAMT}}$-hyb. $2\kappa + 4\rho$

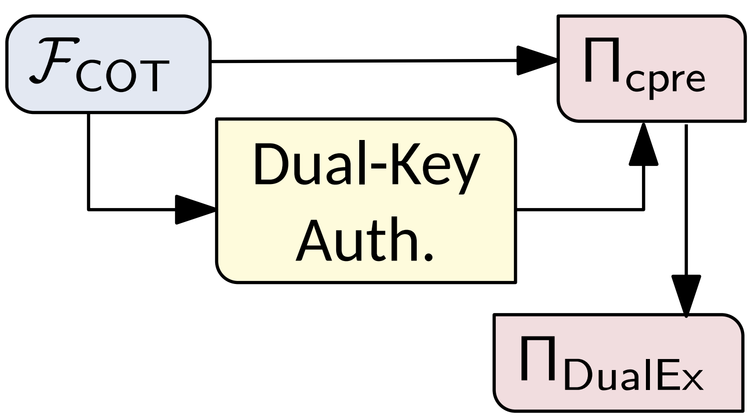
Π_{pre} TinyOT* $+$ Π_{online} Checking $=$ Actively-secure constant-round 2PC

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

a Δ_B $\oplus$ **a** Δ_B $=$ **a** $\cdot \Delta_B$

- 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

a, **â**, Δ_A

a **â** **b** $\hat{\mathbf{b}}$

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

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

Δ_A, Δ_B

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

b, $\hat{\mathbf{b}}$, Δ_B

a **â** **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 $= \mathbf{a} \cdot \Delta_B$

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

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

$\mathcal{F}_{\mathrm{pre}}$ 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	$L_{k,0} \oplus (\lambda_i \cdot \lambda_j \oplus \lambda_k)\Delta_A$
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

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


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

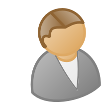
a â b $\hat{\mathbf{b}}$

$\mathcal{F}_{pre}$ 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	$L_{k,0} \oplus K[\Lambda_{00}]$	$M[\Lambda_{00}]$
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	$L_{k,0} \oplus K[\Lambda_{11}]$	$M[\Lambda_{11}]$

Free-XOR GC $\Rightarrow$

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

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

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

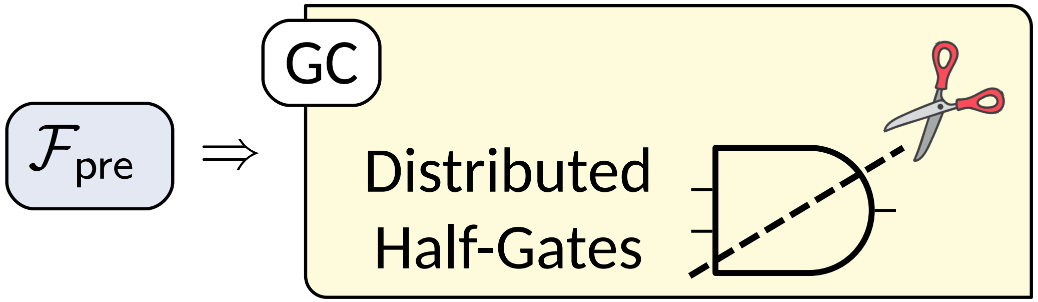
Λ_i	Λ_j	Alice's AuthGC	Bob's AuthGC
0	0	$M[\Lambda_{00}]$	$K[\Lambda_{00}]$
0	1	$M[\Lambda_{01}]$	$K[\Lambda_{01}]$
1	0	$M[\Lambda_{10}]$	$K[\Lambda_{10}]$
1	1	$M[\Lambda_{11}]$	$K[\Lambda_{11}]$

IT-MAC Soundness $\Rightarrow$

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

KRRW18: Distributed Half-Gates Garbling + Equality Checking

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



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

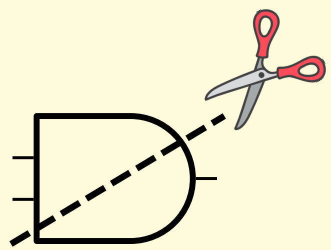
4κ bits/AND WRK17	⇒	$2\kappa + 1$ bits/AND KRRW18

KRRW18: Distributed Half-Gates Garbling + Equality Checking

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

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

$\mathcal{F}_{\text{pre}}$ $\Rightarrow$ GC: Distributed Half-Gates



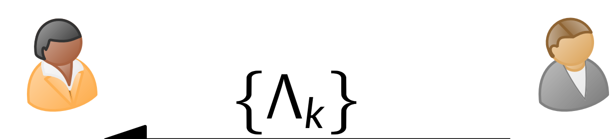
4κ bits/AND WRK17	$\Rightarrow$	$2\kappa + 1$ bits/AND KRRW18

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

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

Check:

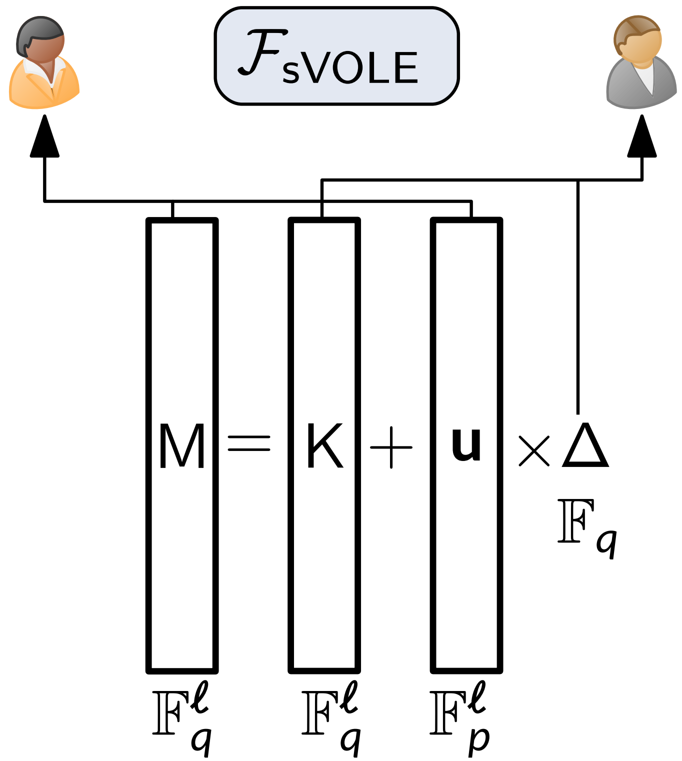
- Evaluator sends $\{\Lambda_w\}$ for all AND gates
- The checking equation reduces to equality check
- Use random linear combination to reduce comm.



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

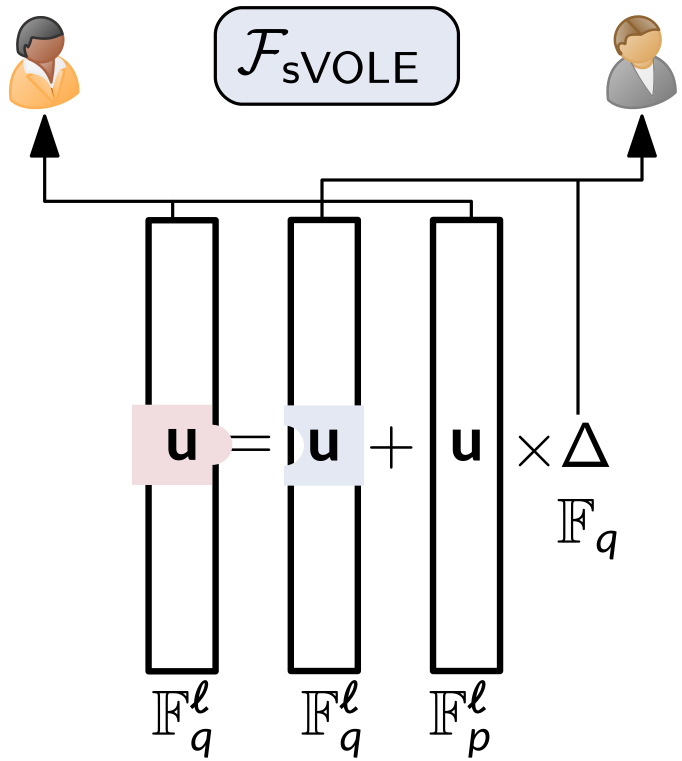
4ρ bits/AND WRK17	$\Rightarrow$	0 bits/AND KRRW18

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



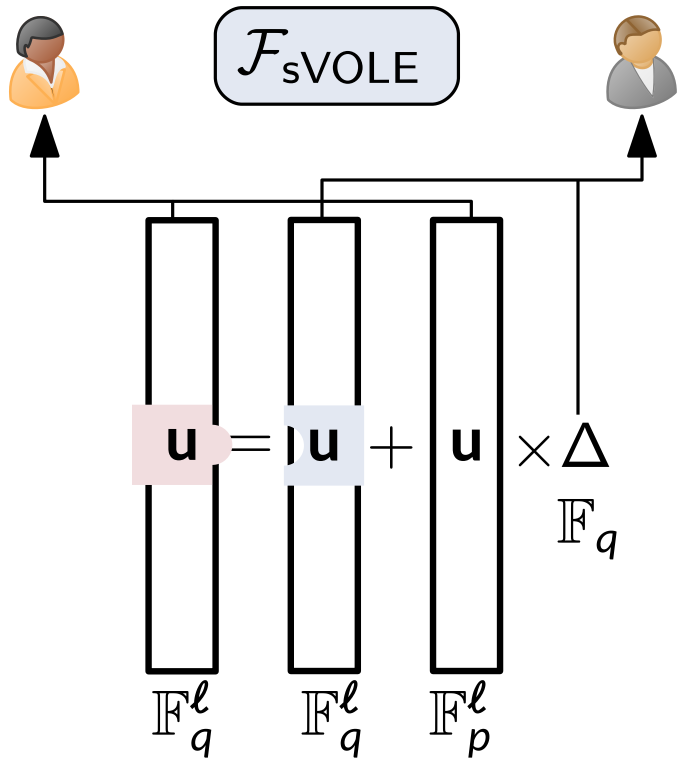
- Efficient protocol for $\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\text{sVOLE}}$ as $\mathcal{F}_{\text{COT}}$

Derandomization operation: Fix

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

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

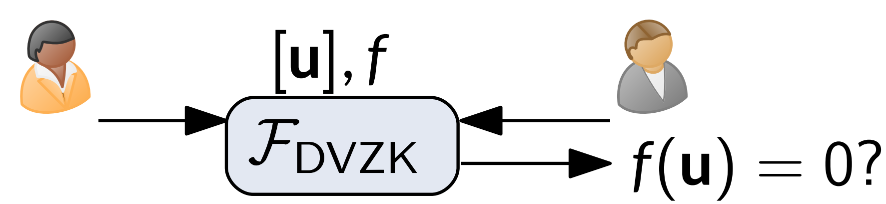
Efficient COT/sVOLE and Designated Verifier Zero Knowledge



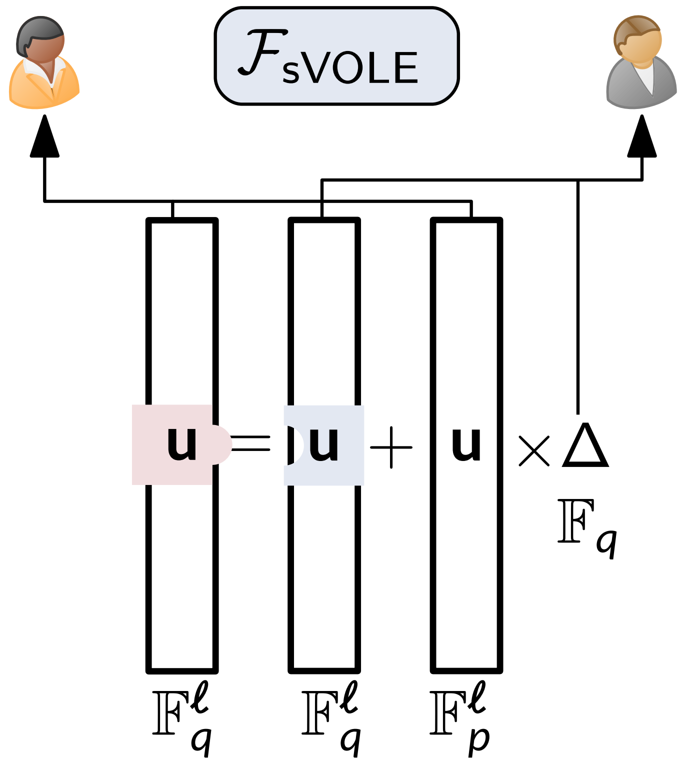
- Efficient protocol for $\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\text{sVOLE}}$ as $\mathcal{F}_{\text{COT}}$

> Derandomization operation: Fix
>
> $\delta := \mathbf{x} \oplus \mathbf{u}$
>
> $\mathbf{x} := \mathbf{u}$ $\qquad$ $\mathbf{x} := \mathbf{u} \oplus \delta \cdot \Delta$

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



Efficient COT/sVOLE and Designated Verifier Zero Knowledge



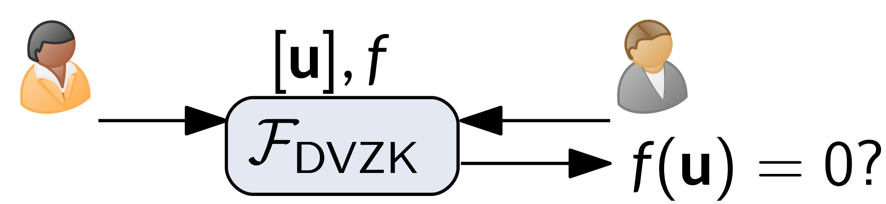
- Efficient protocol for $\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$ with sublinear comm. and linear comp. from LPN [YWL+20,CRR21,...]
- We refer the $\mathbb{F}_p = \mathbb{F}_2$ variant of $\mathcal{F}_{\text{sVOLE}}$ as $\mathcal{F}_{\text{COT}}$

Derandomization operation: Fix

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

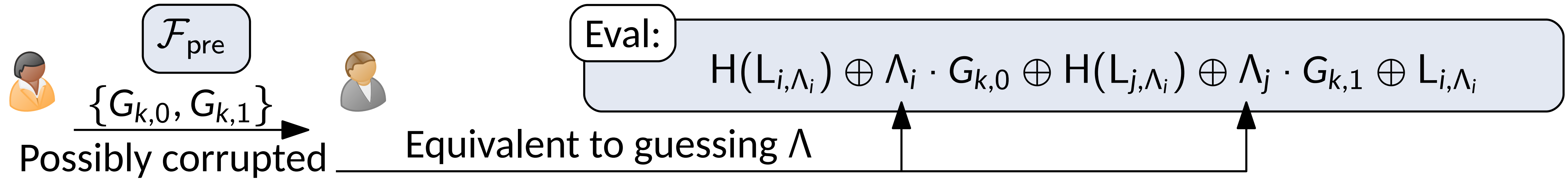
$\mathbf{x} := \mathbf{u}$ $\quad$ $\mathbf{x} := \mathbf{u} \oplus \delta \cdot \Delta$

- 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 an efficient direct $\mathcal{F}_{\text{pre}}$ PCG construction
- The closest is the $\mathcal{F}_{\text{DAMT}}$ 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

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

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

Possibly corrupted

Equivalent to guessing Λ

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

- 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

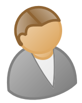
It suffices for **b** to be ρ-wise independent

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

Prior Art: DILO

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

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



Possibly corrupted

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

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

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

- 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 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 $\mathcal{F}_{\mathrm{cpre}}$: Encoding $\mathbf{b}^*$ as Global Keys

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

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

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

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

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

$$\underset{\mathbb{F}_{2^{L\kappa}}^N}{M} = \underset{\mathbb{F}_{2^{L\kappa}}^N}{K} + \underset{\mathbb{F}_2^N}{\mathbf{a}} \times \underset{\mathbb{F}_{2^{L\kappa}}}{\Delta}$$

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

$\Leftrightarrow$

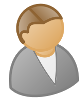
$\mathcal{F}_{\mathrm{bCOT}}$

$$\underset{\mathbb{F}_{2^\kappa}^{N \times L}}{M} = \underset{\mathbb{F}_{2^\kappa}^{N \times L}}{K} + \underset{\mathbb{F}_2^N}{\mathbf{a}} \times \underset{\mathbb{F}_{2^\kappa}^{L \times L}}{\left[\Delta_1 \ldots \Delta_L\right]}$$

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

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

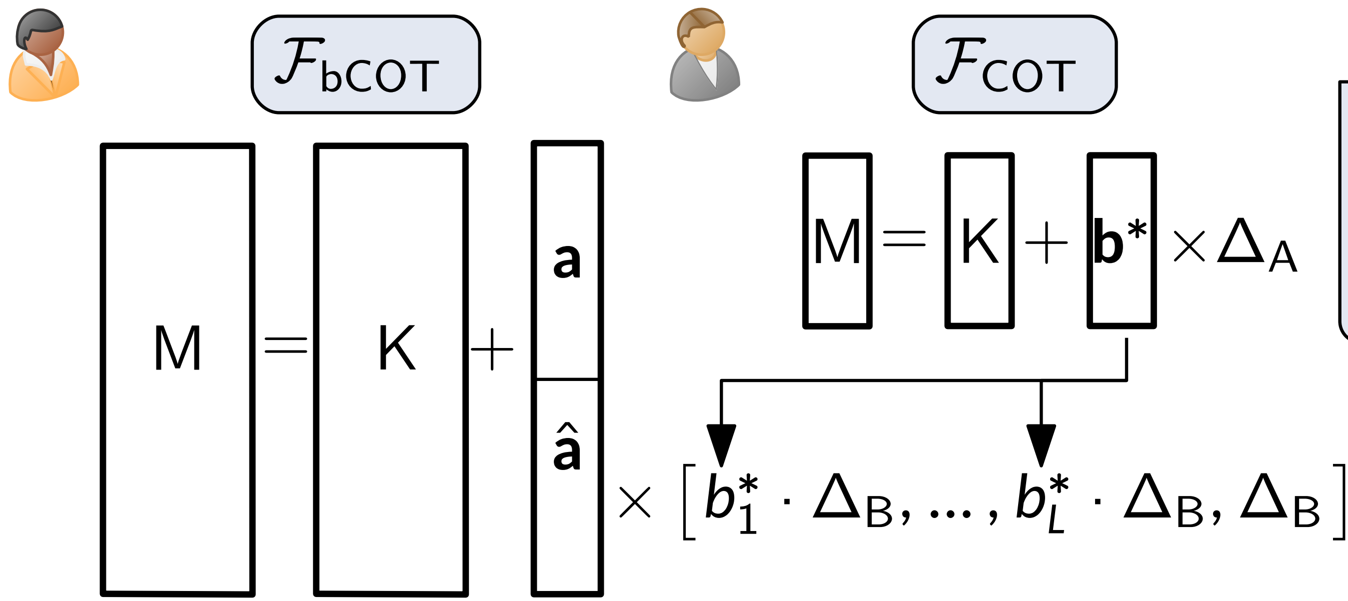
$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \begin{bmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{bmatrix} \times \left[b_1^* \cdot \Delta_{\mathsf{B}}, \ldots, b_L^* \cdot \Delta_{\mathsf{B}}, \Delta_{\mathsf{B}} \right]$$



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

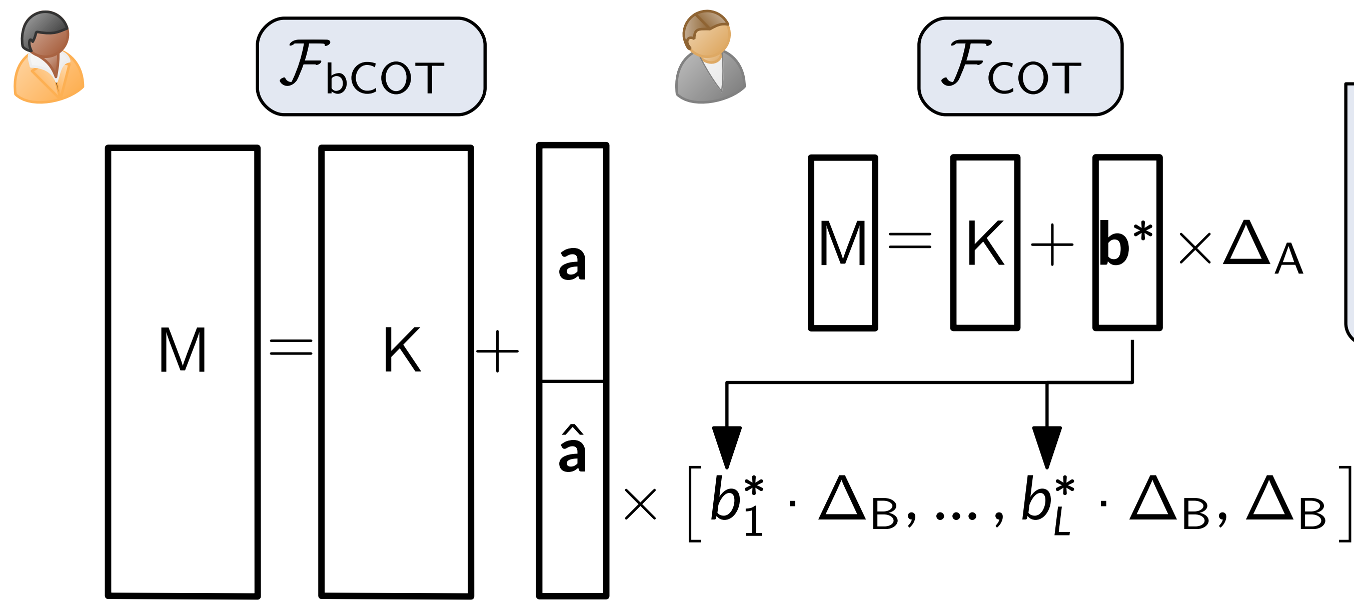
$$\boxed{\mathsf{M}} = \boxed{\mathsf{K}} + \boxed{\mathbf{b}^*} \times \Delta_{\mathsf{A}}$$

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

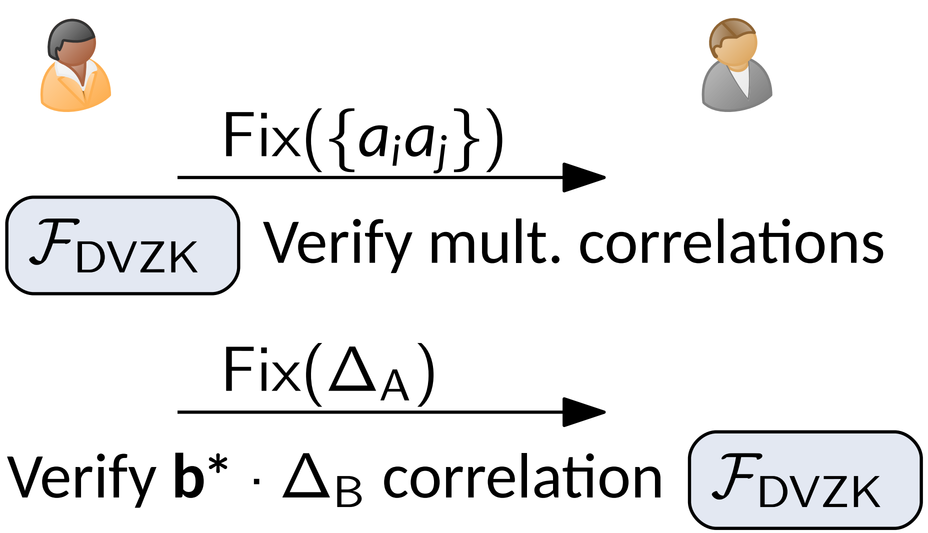


- $[\mathbf{a}]_{b_j^* \Delta_{\mathrm{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\mathrm{B}}}$
- By linearity on IT-MAC, we can get $[a_i b_j]_{\Delta_{\mathrm{B}}}$ for any i, j

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



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



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

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

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

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

- $[\mathbf{a}]_{b_j^* \Delta_{\mathsf{B}}} \equiv [\mathbf{a} b_j^*]_{\Delta_{\mathsf{B}}}$
- 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

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

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

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

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

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

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

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

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

---

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

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

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

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

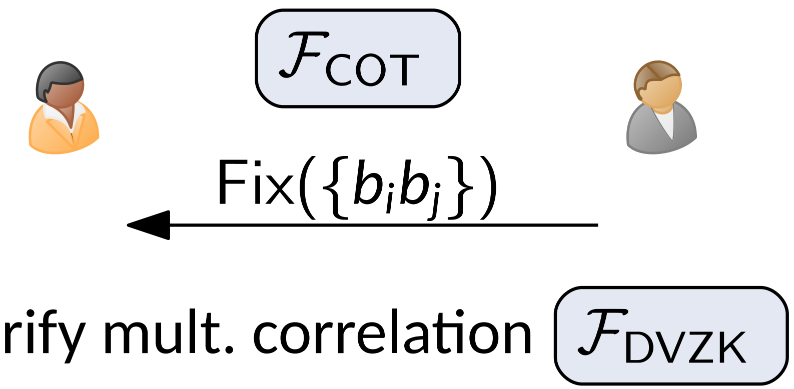
Define $[\tilde{b}_k]_{\Delta_{\mathrm{B}}} := [\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i]_{\Delta_{\mathrm{B}}}$

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

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

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

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

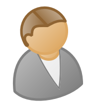


- $\tilde{b}_k = \hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i$
- $\tilde{b}_k \oplus \tilde{b}_k = (\hat{a}_k \oplus a_i a_j \oplus a_i b_j \oplus a_j b_i) \cdot \Delta_\mathrm{A}$

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

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

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

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.

$$\boxed{\Lambda} = \boxed{a} \oplus \boxed{b} \oplus \boxed{z}$$

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.

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

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.

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

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

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

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

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

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.

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

DILO-WRK Check

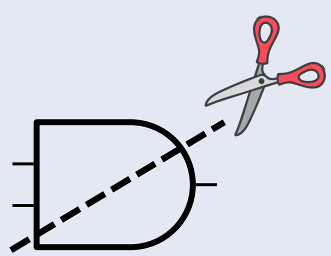
$$\Lambda_k \cdot \Delta_B := \lambda_k \cdot \Delta_B \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_B$$

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

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

GC

Distributed Half-Gates



2κ bits/AND

$+$

AuthGC

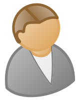
Λ_i	Λ_j	Alice's AuthGC	Bob's AuthGC
0	0	$\mathsf{M}[\Lambda_{00}]$	$\mathsf{K}[\Lambda_{00}]$
0	1	$\mathsf{M}[\Lambda_{01}]$	$\mathsf{K}[\Lambda_{01}]$
1	0	$\mathsf{M}[\Lambda_{10}]$	$\mathsf{K}[\Lambda_{10}]$
1	1	$\mathsf{M}[\Lambda_{11}]$	$\mathsf{K}[\Lambda_{11}]$

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

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

$\rho + 1$ bits

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

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

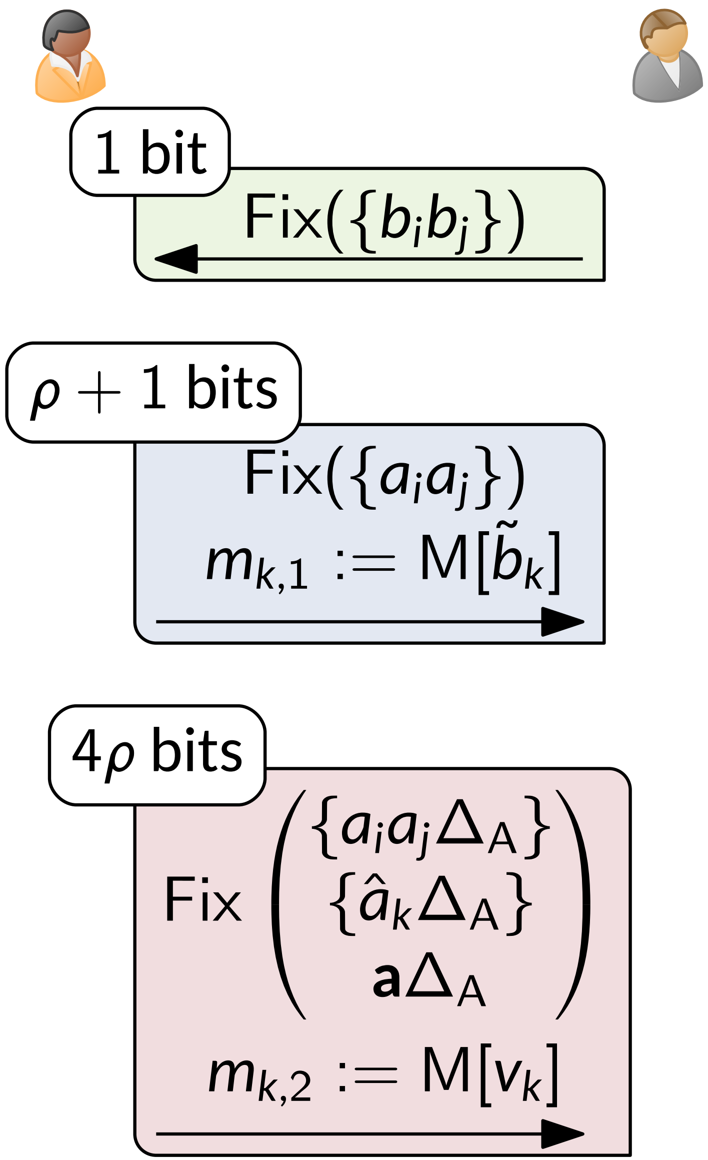
4ρ bits

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

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

Optimizing the Compressed Preprocessing Protocol

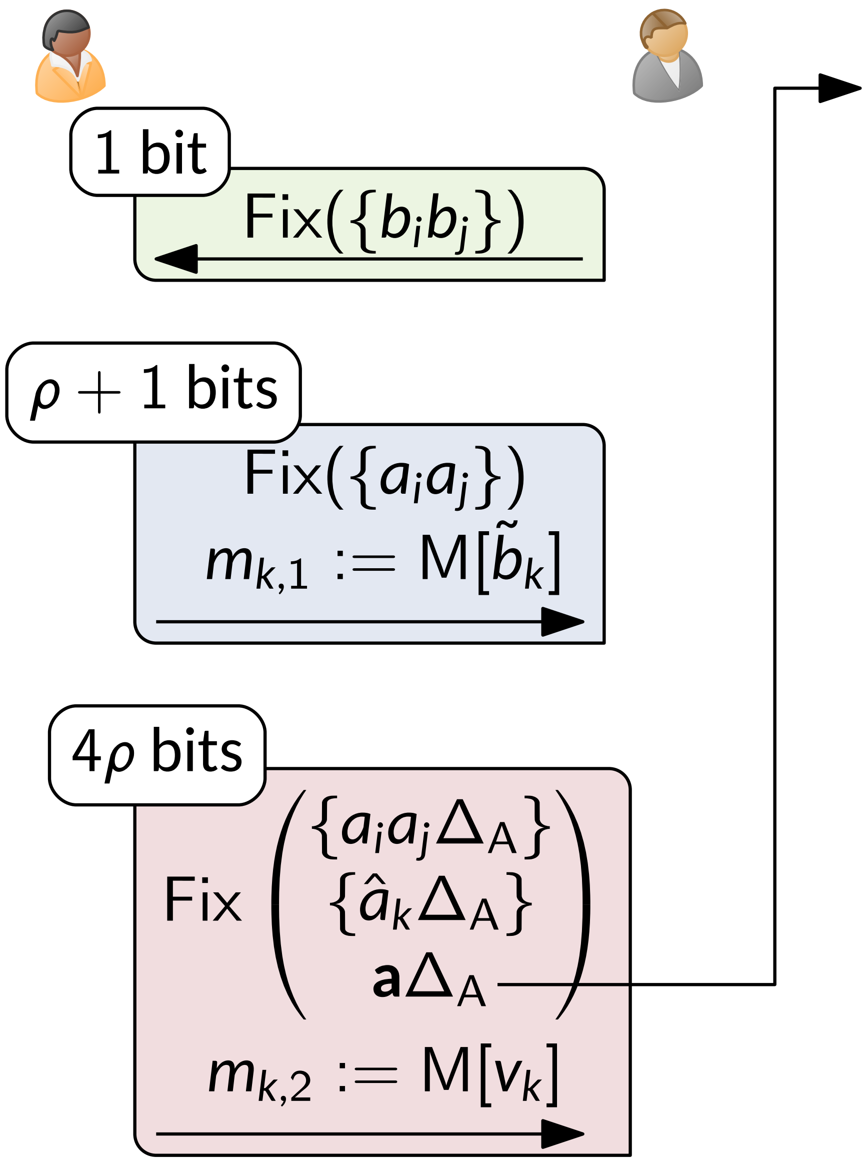
The overhead of DILO is
$5\rho + 2$ bits per AND gate



- 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



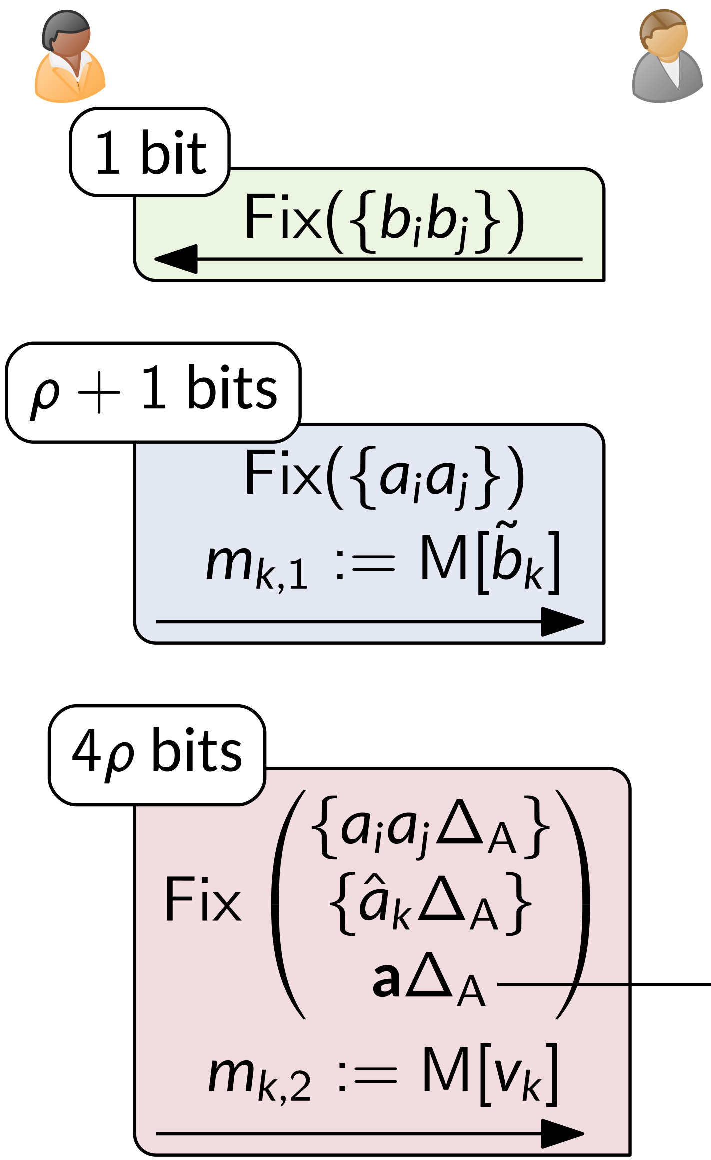
- Why not call $\mathsf{Fix}(\tilde{b}_k)$ directly?
- We need to detect against dishonest $\mathsf{Fix}()$ input

Dual Key Authentication

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

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

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

Dual Key Authentication

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

- 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

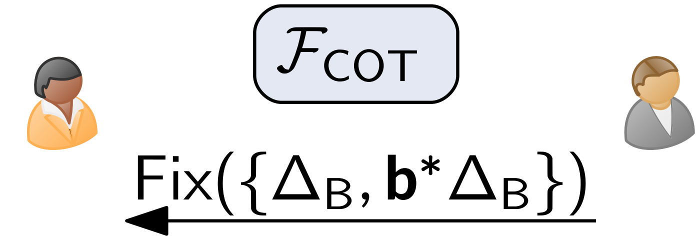
Our goal is to generate $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

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



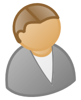
Optimizing the Compressed Preprocessing Protocol, Continued

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

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

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

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



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

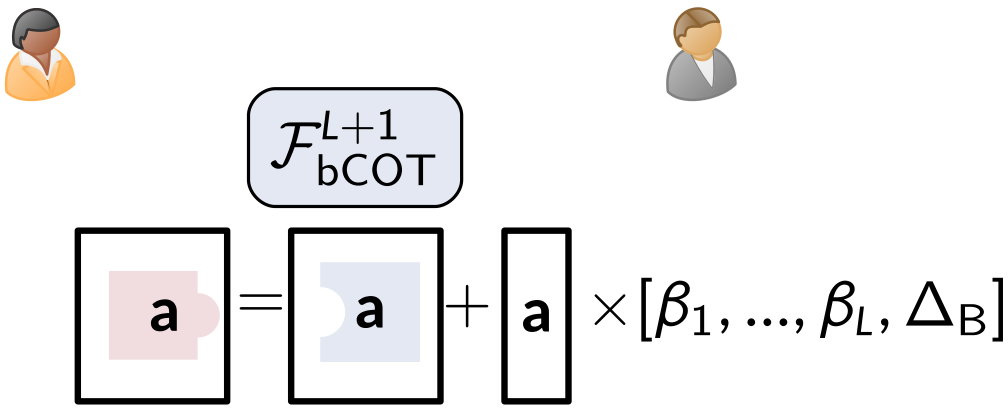
$\Delta_\mathsf{B} \oplus \Delta_\mathsf{B} = \Delta_\mathsf{A} \Delta_\mathsf{B}$

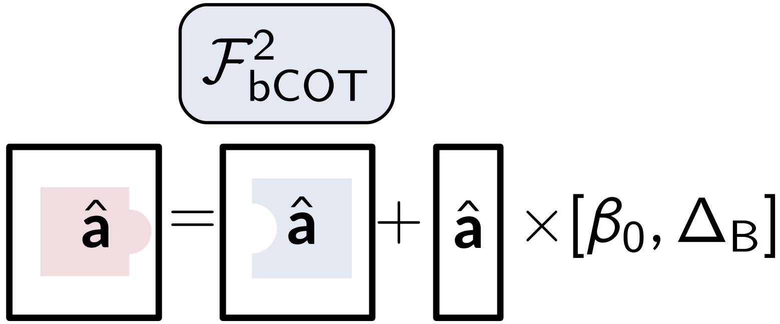
$b_i^* \Delta_\mathsf{B} \oplus b_i^* \Delta_\mathsf{B} = b_i^* \Delta_\mathsf{A} \Delta_\mathsf{B}$

Optimizing the Compressed Preprocessing Protocol, Continued

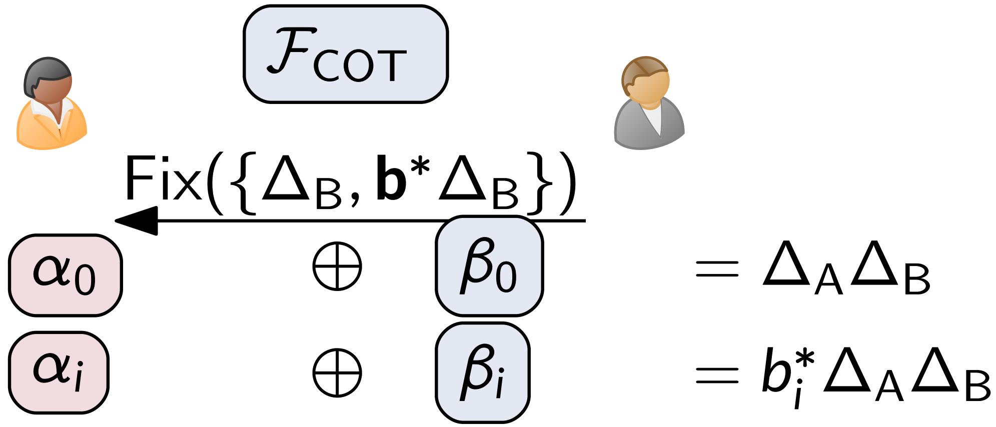
- $\langle \tilde{b}_k \rangle := \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle \oplus \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle$

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





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



Optimizing the Compressed Preprocessing Protocol, Continued

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

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

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

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

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

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

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

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

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

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

$\Leftrightarrow$

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

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

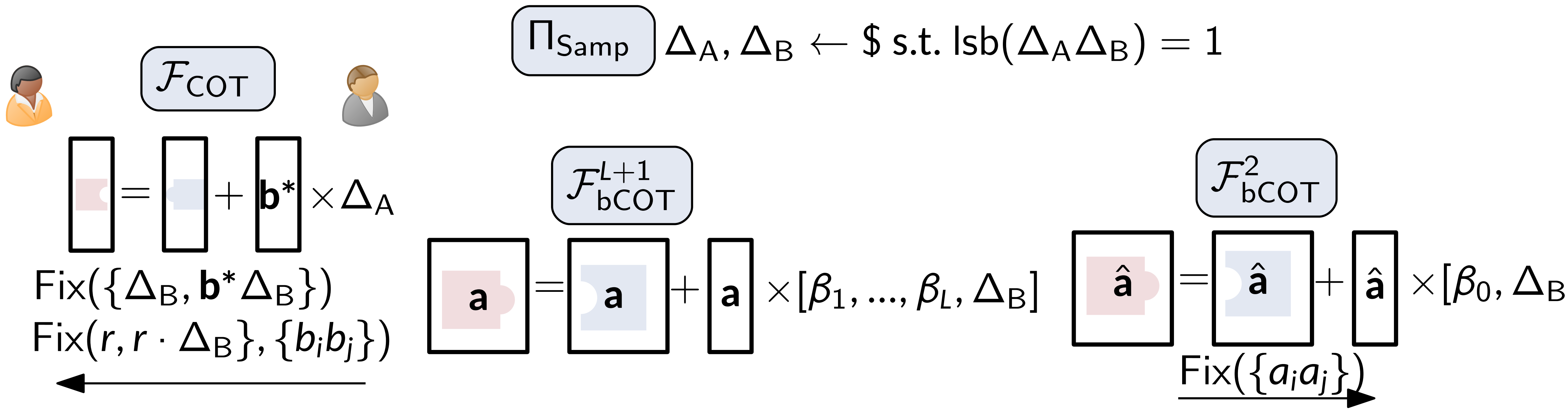
$$\mathrm{K}[\beta_0] \oplus \beta_0 \cdot \Delta'_{\mathrm{A}} = \mathrm{M}[\beta_0]$$
$$\mathrm{K}[\beta_i] \oplus \beta_i \cdot \Delta'_{\mathrm{A}} = \mathrm{M}[\beta_i]$$

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

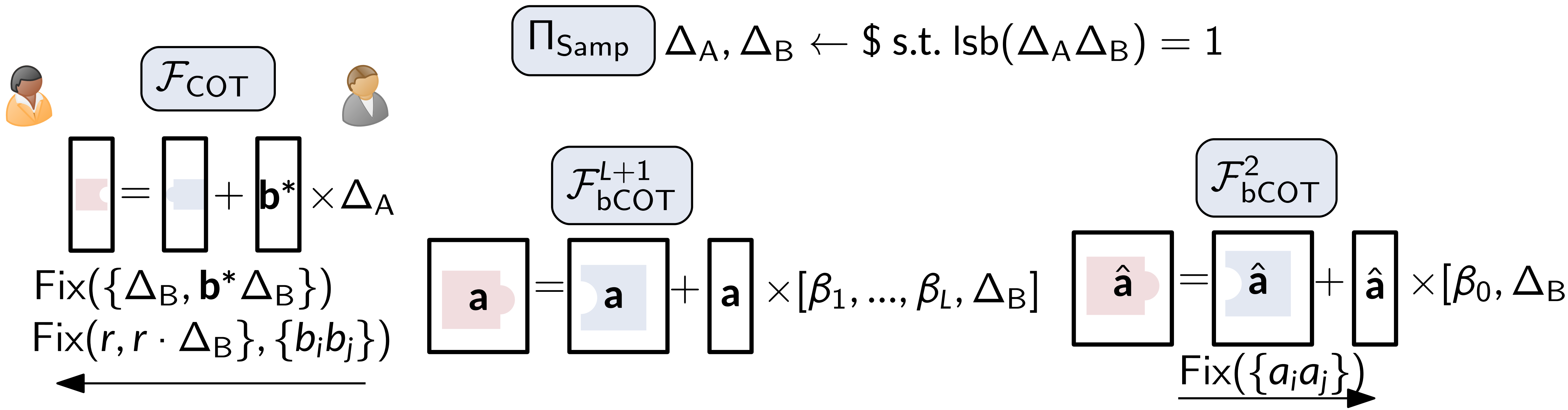
Optimizing the Compressed Preprocessing Protocol, Completed

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

Optimizing the Compressed Preprocessing Protocol, Completed



Optimizing the Compressed Preprocessing Protocol, Completed

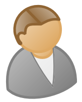


Define $\langle \tilde{b}_k \rangle := \langle a_i b_j \rangle \oplus \langle a_j b_i \rangle \oplus \langle \hat{a}_k \rangle \oplus \langle a_i a_j \rangle$

Optimizing the Compressed Preprocessing Protocol, Completed

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

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



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

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

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

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

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

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

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

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

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

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

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

Optimizing the Compressed Preprocessing Protocol, Completed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Optimizing the Compressed Preprocessing Protocol, Completed

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

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

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

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

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

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

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

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

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

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

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

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

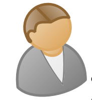
$\tilde{b}_k = m_{k,1} \oplus \mathsf{lsb}(\mathsf{D}_{\mathsf{A}}[\tilde{b}_k])$

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

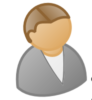
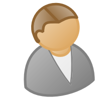
$y := r + \sum_k \chi^k \cdot \tilde{b}_k$ $\longleftarrow$

- 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 $\mathsf{CheckZero}([y] - y)$

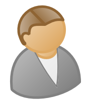
Optimizing the One-way Communication Via Dual Execution

- 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?


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


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

Π_{GC}

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

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


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

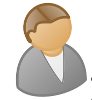
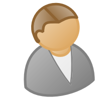

x for $\mathcal{I}_{\text{A}}$

Π_{GC}

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

$\{\Lambda'_k, \text{L}'_{k,\Lambda'_k}\}$ 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 =  $\to$ : $2\kappa + 3\rho + 2$ bits,  $\to$ : 2 bits
- How about optimizing one-way communication? Maybe dual execution?

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

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

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

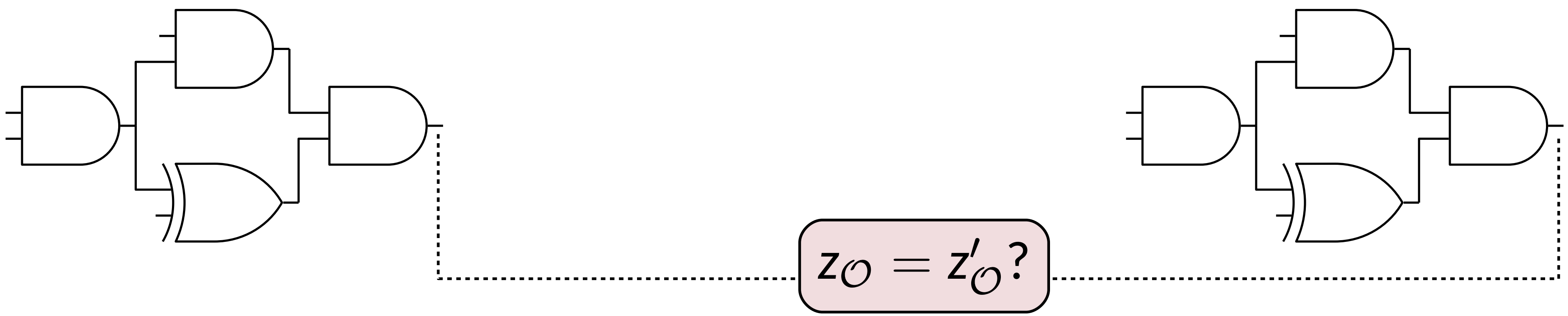
$\{z_w\}$ for $w \in \mathcal{O}$

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

$\{\Lambda'_k, \mathsf{L}'_{k,\Lambda'_k}\}$ 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 = $\to$: $2\kappa + 3\rho + 2$ bits, $\to$: 2 bits
- How about optimizing one-way communication? Maybe dual execution?

x for $\mathcal{I}_A$ $\quad \Pi_{\text{GC}} \quad$ **y** for $\mathcal{I}_B$

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

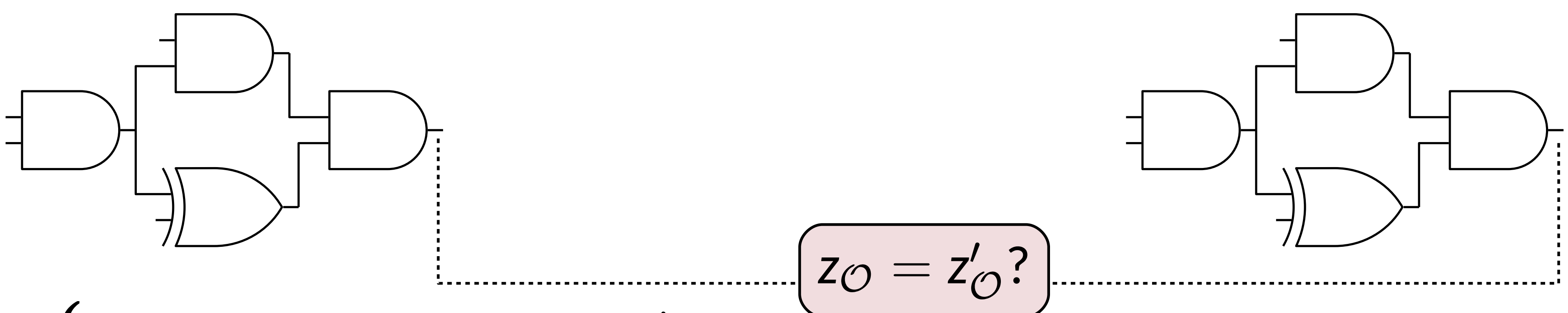
$\{z_w\}$ for $w \in \mathcal{O}$

y for $\mathcal{I}_B$ $\quad \Pi_{\text{GC}} \quad$ **x** for $\mathcal{I}_A$

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

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

- [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

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

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

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

Π_{DG}

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

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

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

$$\text{s.t. } \hat{a}'_k \oplus \hat{b}'_k = (a'_i \oplus b'_i) \cdot (a'_j \oplus b'_j)$$

Π_{DG}

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

Optimizing the One-way Communication Via Dual Execution

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

$[\mathbf{a}], [\hat{\mathbf{a}}], [\mathbf{b}], [\hat{\mathbf{b}}], \Delta_A, \Delta_B \leftarrow \$$

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 $k \in \mathcal{W}$

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

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

$[\mathbf{a}'], [\hat{\mathbf{a}}'], [\mathbf{b}'], [\hat{\mathbf{b}}'], \Delta_A, \Delta_B \leftarrow \$$

s.t. $\hat{a}'_k \oplus \hat{b}'_k = (a'_i \oplus b'_i) \cdot (a'_j \oplus b'_j)$

Π_{DG}

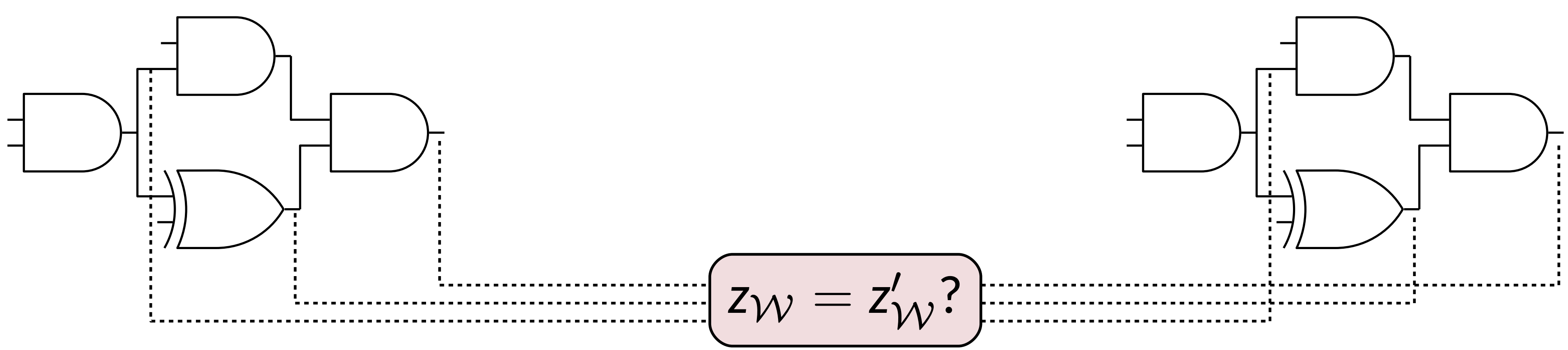
$\{\lambda'_k, \mathsf{L}'_{k,0}, \mathsf{L}'_{k,1}\}$

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

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

[HK21] Garbled Sharing

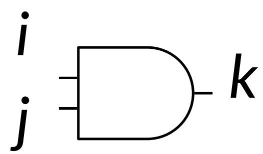
- Color bits and wire masks are authenticated for every wire
- This enables checking equality for every wire across two executions



Optimizing the Two-way Communication

- Based on the WRK consistency checking

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



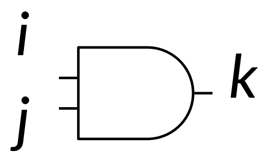
Λ_i	Λ_j	Alice's AuthGC	Bob's AuthGC
0	0	$\mathsf{M}[\Lambda_{00}]$	$\mathsf{K}[\Lambda_{00}]$
0	1	$\mathsf{M}[\Lambda_{01}]$	$\mathsf{K}[\Lambda_{01}]$
1	0	$\mathsf{M}[\Lambda_{10}]$	$\mathsf{K}[\Lambda_{10}]$
1	1	$\mathsf{M}[\Lambda_{11}]$	$\mathsf{K}[\Lambda_{11}]$

IT-MAC Soundness $\Rightarrow$
$$|\Delta_{\mathsf{B}}| = \rho \approx 40$$

Optimizing the Two-way Communication

- Based on the WRK consistency checking

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



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

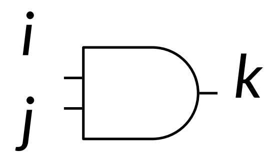
IT-MAC Soundness $\Rightarrow$
$|\Delta_\mathsf{B}| = \rho \approx 40$

- Equivalent to checking $\mathsf{K}[\Lambda_k] = \mathsf{M}[\Lambda_k] \oplus \Lambda_k \cdot \Delta_\mathsf{B}$
- Reduces to linear test if $\Lambda_i, \Lambda_j, \Lambda_k$ are public
- Due to compression, this method does not apply

Optimizing the Two-way Communication

- Based on the WRK consistency checking

$$\begin{aligned}\color{red}{\Lambda_k} &:= \lambda_k \oplus (\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \\ &= \lambda_k \oplus \Lambda_i\Lambda_j \oplus \Lambda_i\lambda_j \oplus \lambda_i\Lambda_j \oplus \lambda_i\lambda_j\end{aligned}$$



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

IT-MAC Soundness $\Rightarrow$
$|\Delta_\mathsf{B}| = \rho \approx 40$

- Equivalent to checking $\color{red}{\mathsf{K}[\Lambda_k] = \mathsf{M}[\Lambda_k]} \oplus \Lambda_k \cdot \Delta_\mathsf{B}$
- Reduces to linear test if $\Lambda_i, \Lambda_j, \Lambda_k$ are public
- Due to compression, this method does not apply

- Our task is to securely check $\color{red}{\Lambda_k} \cdot \Delta_\mathsf{B} = \Lambda_k \cdot \Delta_\mathsf{B}$
- Equivalent to $(\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \cdot \Delta_\mathsf{B} = (\Lambda_k \oplus \lambda_k) \cdot \Delta_\mathsf{B}$

Expanding the Terms

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

$$(\underbrace{\Lambda_i \cdot \Lambda_j \oplus \Lambda_k \oplus b_k \oplus \hat{b}_k \oplus \Lambda_i \cdot b_j \oplus \Lambda_j \cdot b_i}_{B'_k} \oplus a_k \oplus \hat{a}_k \oplus \Lambda_i \cdot a_j \oplus \Lambda_j \cdot a_i) \cdot \Delta_{\mathsf{B}} = 0 \ .$$

$$B'_k \cdot \Delta_{\mathsf{B}} \oplus \mathsf{M}[a_k] \oplus \mathsf{K}[a_k] \oplus \mathsf{M}[\hat{a}_k] \oplus \mathsf{K}[\hat{a}_k] \oplus \Lambda_i \cdot (\mathsf{M}[a_j] \oplus \mathsf{K}[a_j]) \oplus \Lambda_j \cdot (\mathsf{M}[a_i] \oplus \mathsf{K}[a_i]) = 0 \ .$$

$$\underbrace{B'_k \cdot \Delta_{\mathsf{B}} \oplus \mathsf{K}[a_k] \oplus \mathsf{K}[\hat{a}_k] \oplus \Lambda_i \cdot \mathsf{K}[a_j] \oplus \Lambda_j \cdot \mathsf{K}[a_i]}_{B_k} \oplus \underbrace{\mathsf{M}[a_k] \oplus \mathsf{M}[\hat{a}_k]}_{A_{k,0}} \oplus \Lambda_i \cdot \mathsf{M}[a_j] \oplus \Lambda_j \cdot \mathsf{M}[a_i] = 0 \ .$$

Expanding the Terms

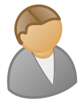
- $(\Lambda_i \Lambda_j \oplus \Lambda_i \lambda_j \oplus \lambda_i \Lambda_j \oplus \lambda_i \lambda_j) \cdot \Delta_{\mathrm{B}} = (\Lambda_k \oplus \lambda_k) \cdot \Delta_{\mathrm{B}}$

$$(\underbrace{\Lambda_i \cdot \Lambda_j \oplus \Lambda_k \oplus b_k \oplus \hat{b}_k \oplus \Lambda_i \cdot b_j \oplus \Lambda_j \cdot b_i}_{B'_k} \oplus a_k \oplus \hat{a}_k \oplus \Lambda_i \cdot a_j \oplus \Lambda_j \cdot a_i) \cdot \Delta_{\mathrm{B}} = 0 \ .$$

$$B'_k \cdot \Delta_{\mathrm{B}} \oplus \mathrm{M}[a_k] \oplus \mathrm{K}[a_k] \oplus \mathrm{M}[\hat{a}_k] \oplus \mathrm{K}[\hat{a}_k] \oplus \Lambda_i \cdot (\mathrm{M}[a_j] \oplus \mathrm{K}[a_j]) \oplus \Lambda_j \cdot (\mathrm{M}[a_i] \oplus \mathrm{K}[a_i]) = 0 \ .$$

$$\underbrace{B'_k \cdot \Delta_{\mathrm{B}} \oplus \mathrm{K}[a_k] \oplus \mathrm{K}[\hat{a}_k] \oplus \Lambda_i \cdot \mathrm{K}[a_j] \oplus \Lambda_j \cdot \mathrm{K}[a_i]}_{B_k} \oplus \underbrace{\mathrm{M}[a_k] \oplus \mathrm{M}[\hat{a}_k]}_{A_{k,0}} \oplus \Lambda_i \cdot \mathrm{M}[a_j] \oplus \Lambda_j \cdot \mathrm{M}[a_i] = 0 \ .$$

- Half-Gates



$\mathrm{L}_{i,0}, \mathrm{L}_{i,1}$ $\qquad$ L_{i,Λ_i}

$$\mathrm{G}_0 = \mathrm{H}(\mathrm{L}_{i,0}) \oplus \mathrm{H}(\mathrm{L}_{i,1}) \oplus M \longrightarrow$$

$$\mathrm{H}(\mathrm{L}_{i,\Lambda_i}) = \mathrm{H}(\mathrm{L}_{i,0}) \oplus \Lambda_i \cdot (\mathrm{H}(\mathrm{L}_{i,0}) \oplus \mathrm{H}(\mathrm{L}_{i,1}))$$

$$H(\mathrm{L}_{i,\Lambda_i}) \oplus \Lambda_i \cdot G_0 = H(\mathrm{L}_{i,0}) \oplus \Lambda_i \cdot M$$

Merging Two Multiplications

- Observation: In Free-XOR, each AND gate input value Λ_i, Λ_j is a *public* linear combination of previous AND outputs
- Denoted as $\Lambda_i = \sum_k c_k^i \cdot \Lambda_k$

Merging Two Multiplications

- Observation: In Free-XOR, each AND gate input value Λ_i, Λ_j is a *public* linear combination of previous AND outputs
- Denoted as $\Lambda_i = \sum_k c_k^i \cdot \Lambda_k$
- To check the entire circuit, we need to evaluate

$$\sum_{(i,j,k,\wedge)\in\mathcal{C}} \chi^k \cdot \big((\Lambda_i \oplus \lambda_i) \cdot (\Lambda_j \oplus \lambda_j) \oplus (\Lambda_k \oplus \lambda_k)\big) \cdot \Delta_{\mathsf{B}} = 0$$

- Equivalent to evaluating the secret sharing of $\sum_{i,j,k} \Lambda_i \cdot \mathsf{M}[a_j] \oplus \sum_{i,j,k} \Lambda_j \cdot \mathsf{M}[a_i]$

$$\sum_k \chi^k \cdot B_k \oplus \sum_k \chi^k \cdot A_{k,0} \oplus \sum_{(i',j',k',\wedge)} \chi^{k'} \cdot \Big(\big(\sum_k c_k^{i'} \cdot \Lambda_k\big) \cdot \mathsf{M}[a_{j'}] \oplus \big(\sum_k c_k^{j'} \cdot \Lambda_k\big) \cdot \mathsf{M}[a_{i'}]\Big) = 0 \ .$$

$$\sum_k \chi^k \cdot B_k \oplus \sum_k \chi^k \cdot A_{k,0} \oplus \sum_k \Lambda_k \cdot \underbrace{\sum_{(i',j',k',\wedge)} \chi^{k'} \cdot \big(c_k^{i'} \cdot \mathsf{M}[a_{j'}] \oplus c_k^{j'} \cdot \mathsf{M}[a_{i'}]\big)}_{A_{k,1}} = 0 \ .$$

Removing the Random Oracle

- CRHF cannot offer pseudorandomness, so we change to "sum of TCCR hash"

(b) $\mathsf{P_A}$ computes $h := \mathsf{H}'(V_1^{\mathsf{A}}, \ldots, V_t^{\mathsf{A}})$, and then sends it to $\mathsf{P_B}$ who checks that $h = \mathsf{H}'(V_1^{\mathsf{B}}, \ldots, V_t^{\mathsf{B}})$. If the check fails, $\mathsf{P_B}$ aborts.

(b) $\mathsf{P_A}$ computes $h_{\mathsf{A}} := \sum_{i=1}^{t} \mathsf{H_{tccr}}(V_i^{\mathsf{A}})$, and then sends it to $\mathsf{P_B}$ who computes $h_{\mathsf{B}} := \sum_{i=1}^{t} \mathsf{H_{tccr}}(V_i^{\mathsf{B}})$ and checks that $h_{\mathsf{A}} = h_{\mathsf{B}}$. If the check fails then $\mathsf{P_B}$ aborts.

Removing the Random Oracle

- CRHF cannot offer pseudorandomness, so we change to "sum of TCCR hash"

(b) $\mathsf{P_A}$ computes $h := \mathsf{H}'(V_1^{\mathsf{A}}, \ldots, V_t^{\mathsf{A}})$, and then sends it to $\mathsf{P_B}$ who checks that $h = \mathsf{H}'(V_1^{\mathsf{B}}, \ldots, V_t^{\mathsf{B}})$. If the check fails, $\mathsf{P_B}$ aborts.

(b) $\mathsf{P_A}$ computes $h_{\mathsf{A}} := \sum_{i=1}^{t} \mathsf{H_{tccr}}(V_i^{\mathsf{A}})$, and then sends it to $\mathsf{P_B}$ who computes $h_{\mathsf{B}} := \sum_{i=1}^{t} \mathsf{H_{tccr}}(V_i^{\mathsf{B}})$ and checks that $h_{\mathsf{A}} = h_{\mathsf{B}}$. If the check fails then $\mathsf{P_B}$ aborts.

- In some cases, we need the following to be pseudorandom

$$\mathsf{H}(x \oplus j \cdot \Delta, i) \ \ s.t. \ \ i \in \mathbb{F}_{2^\kappa}, j \in \mathbb{F}_{2^\kappa}^*$$

- This is referred to as "extended TCR" in the current draft
- Luckily, the TMMO construction in GKWY20 still works in the RPM

$$\mathsf{TMMO}_\Delta^\pi(x, i, j) = \pi(\pi(x \oplus \Delta \cdot j) \oplus i) \oplus \pi(x \oplus \Delta \cdot j)$$

Multi-use of Hash Functions in the Standard Model

- String-OT needs TCR
- Half-Gate needs CCRND
- No multi-use security definition
- Must satisfy them simultaneously $\rightarrow$ TCCR

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	$8\kappa + 11$ ($4.04\times$)	$16\kappa + 22$ ($8.09\times$)
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