Row Reduction Techniques for n-Party Garbling

Erik Pohle
erik.pohle@esat.kuleuven.be

CRYPTO August 19th, 2025

COSIC, KU Leuven, Belgium

joint work with Kelong Cong[1], Emmanuela Orsini[2] and Oliver Zajonc[3]

1 Zama, 2 Bocconi University, 3 COSIC, KU Leuven

Overview

Full Row Reduction:

Circuit size:

- $3n\kappa$ (for [HSS17]-style)
- $3(n-1)\kappa$ (for authenticated garbling [WRK17,YZW20])

Improved Preprocessing:

- previous schemes have $4n\kappa$ and $(4n-6)\kappa$ circuit size
- solves open problem from [WRK17]

Overview

Full Row Reduction:

Circuit size:

- $3n\kappa$ (for [HSS17]-style)
- $3(n-1)\kappa$ (for authenticated garbling [WRK17,YZW20])

Improved Preprocessing:

Authenticated field triples with $\mathcal{O}(n^3\sqrt{|C|})$ comm.

Mask preparation with $\mathcal{O}(2\rho|C|)$ comm.

- previous schemes have $4n\kappa$ and $(4n-6)\kappa$ circuit size
- solves open problem from [WRK17]

- generalizes approach by [DILO22] to $n \geq 3$ parties
- improves communication for large circuits

Overview

Full Row Reduction:

Circuit size:

- $3n\kappa$ (for
- $3(n-1)\kappa$ garbling

What will follow

- Two-Party Garbled Circuits
- Authenticated Multi-Party GC
- Our Construction
- Preprocessing
- Results

- previous sc $(4n-6)\kappa$ circuit size
- solves open problem from [WRK17]

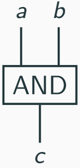
[DILO22] to $n \geq 3$ parties

- improves communication for large circuits

Two-Party Garbled Circuits

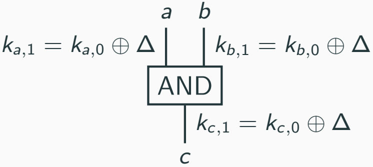
Garbler

Evaluator



Two-Party Garbled Circuits

Garbler

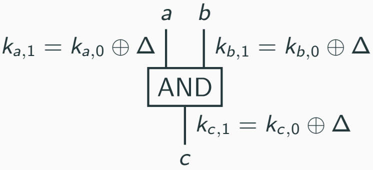


Evaluator

- $\Delta \leftarrow\!\!\$ \{0,1\}^\kappa$ for the whole circuit
- $k_{a,0}, k_{b,0}, k_{c,0} \leftarrow\!\!\$ \{0,1\}^\kappa$ for the gate

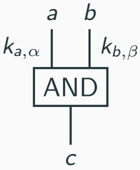
Two-Party Garbled Circuits

Garbler



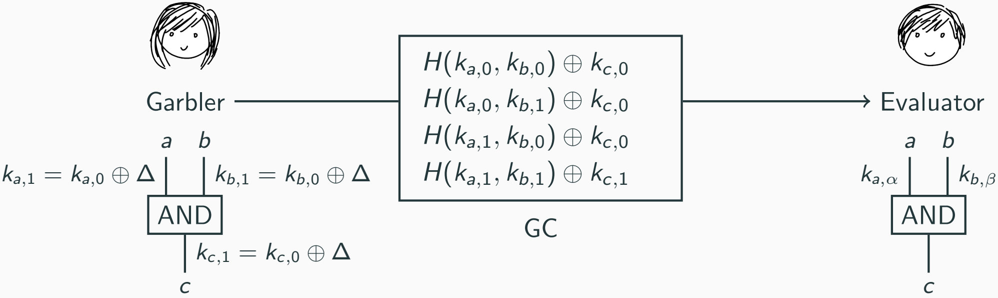
- $\Delta \leftarrow\!\!\$ \{0,1\}^\kappa$ for the whole circuit
- $k_{a,0}, k_{b,0}, k_{c,0} \leftarrow\!\!\$ \{0,1\}^\kappa$ for the gate

Evaluator



- knows $k_{a,\alpha} = k_{a,0} \oplus \alpha \cdot \Delta$ and $k_{b,\beta} = k_{b,0} \oplus \beta \cdot \Delta$

Two-Party Garbled Circuits



- $\Delta \leftarrow\$ \{0,1\}^\kappa$ for the whole circuit
- $k_{a,0}, k_{b,0}, k_{c,0} \leftarrow\$ \{0,1\}^\kappa$ for the gate

- knows $k_{a,\alpha} = k_{a,0} \oplus \alpha \cdot \Delta$ and $k_{b,\beta} = k_{b,0} \oplus \beta \cdot \Delta$

Two-Party Garbled Circuits

Garbler

a b

$k_{a,1} = k_{a,0} \oplus \Delta$ $k_{b,1} = k_{b,0} \oplus \Delta$

AND

$k_{c,1} = k_{c,0} \oplus \Delta$

c

$$
\boxed{
\begin{array}{l}
H(k_{a,0}, k_{b,0}) \oplus k_{c,0} \\
H(k_{a,0}, k_{b,1}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,0}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,1}) \oplus k_{c,1}
\end{array}
}
$$

GC

Evaluator

a b

$k_{a,\alpha}$ $k_{b,\beta}$

AND

$k_{c,\alpha\wedge\beta}$

c

- $\Delta \leftarrow\$ \{0,1\}^\kappa$ for the whole circuit
- $k_{a,0}, k_{b,0}, k_{c,0} \leftarrow\$ \{0,1\}^\kappa$ for the gate

- knows $k_{a,\alpha} = k_{a,0} \oplus \alpha \cdot \Delta$ and $k_{b,\beta} = k_{b,0} \oplus \beta \cdot \Delta$
- can decrypt one row correctly with $H(k_{a,\alpha}, k_{b,\beta})$
- obtains $k_{c,\alpha\wedge\beta}$

Row Reduction in Two-Party Setting [NPR99]

- Garbler sets $k_{c,0} = H(k_{a,0}, k_{b,0})$
- only 3 rows left

Garbler

$$
\boxed{\begin{array}{l}
H(k_{a,0}, k_{b,1}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,0}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,1}) \oplus k_{c,1}
\end{array}}
$$

GC

Evaluator

Row Reduction in Two-Party Setting [NPR99]

- Garbler sets $k_{c,0} = H(k_{a,0}, k_{b,0})$
- only 3 rows left

Garbler

$$
\boxed{\begin{array}{l}
H(k_{a,0}, k_{b,1}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,0}) \oplus k_{c,0} \\
H(k_{a,1}, k_{b,1}) \oplus k_{c,1}
\end{array}}
$$

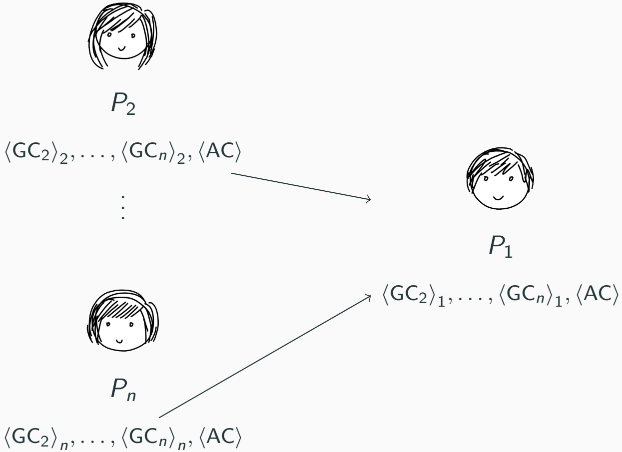
GC

Evaluator

⚠ garbler can no longer choose $k_{c,0}$ freely $\implies$ cannot choose all $k_{c,0}$ at the same time!

Authenticated Garbling for Multi-Party Garbled Circuits [WRK17,YWZ20]

- all parties jointly create *shares* of garbled circuit(s)



- Setting: active security, dishonest majority
- in [WRK17]: $n-1$ garblers, 1 evaluator
- $(n-1)$ GC's to evaluate + information for evaluator to check correctness

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus (\underbrace{\lambda_a \oplus \alpha}_{\hat{a}})\Delta^i$

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus (\underbrace{\lambda_a \oplus \alpha}_{\hat{a}})\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \left(\lambda^i, \{\ \langle \lambda^i \Delta^j \rangle, \langle \lambda^j \Delta^i \rangle\ \}_{j \neq i}\right)$
- same Δ as MAC key and as FreeXOR offset

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus (\underbrace{\lambda_a \oplus \alpha}_{\hat{a}})\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \left(\lambda^i, \{\ \langle\lambda^i\Delta^j\rangle, \langle\lambda^j\Delta^i\rangle\ \}_{j\neq i}\right)$
- same Δ as MAC key and as FreeXOR offset
- parties hold $[\![r_{\hat{a}\hat{b}}]\!] = [\![\overbrace{(\lambda_a \oplus \hat{a})}^{\alpha}\overbrace{(\lambda_b \oplus \hat{b})}^{\beta} \oplus \lambda_c]\!] \qquad \forall(\hat{a}, \hat{b}) \in \{0,1\}^2$

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus \underbrace{(\lambda_a \oplus \alpha)}_{\hat{a}}\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \left(\lambda^i, \{\ \langle\lambda^i\Delta^j\rangle, \langle\lambda^j\Delta^i\rangle\ \}_{j\neq i}\right)$
- same Δ as MAC key and as FreeXOR offset
- parties hold $[\![r_{\hat{a}\hat{b}}]\!] = [\![(\overbrace{\lambda_a \oplus \hat{a}}^{\alpha})(\overbrace{\lambda_b \oplus \hat{b}}^{\beta}) \oplus \lambda_c]\!] \qquad \forall(\hat{a}, \hat{b}) \in \{0,1\}^2$

$$\mathrm{GC}_i[\hat{a}, \hat{b}] = H(k^i_{a,\hat{a}}, k^i_{b,\hat{b}}) \oplus \left(\underset{\text{masked output bit}}{r^i_{\hat{a}\hat{b}}}, \{\ \underset{\langle \mathrm{GC}_j \rangle}{\langle k^j_{w,0} \oplus r_{\hat{a}\hat{b}}\Delta^j\rangle_i}\ \}_{j>1},\ \underset{\langle AC \rangle}{\langle r_{\hat{a}\hat{b}}\Delta^1\rangle} \right) \qquad \forall(\hat{a}, \hat{b}) \in \{0,1\}^2$$

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus \underbrace{(\lambda_a \oplus \alpha)}_{\hat{a}}\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \Big(\lambda^i, \{\ \langle\lambda^i\Delta^j\rangle, \langle\lambda^j\Delta^i\rangle\ \}_{j\neq i}\Big)$
- same Δ as MAC key and as FreeXOR offset
- parties hold $[\![r_{\hat{a}\hat{b}}]\!] = [\![\overbrace{(\lambda_a \oplus \hat{a})}^{\alpha}\overbrace{(\lambda_b \oplus \hat{b})}^{\beta} \oplus \lambda_c]\!] \qquad \forall(\hat{a},\hat{b}) \in \{0,1\}^2$

$$\mathrm{GC}_i[\hat{a},\hat{b}] = H(k^i_{a,\hat{a}}, k^i_{b,\hat{b}}) \oplus \Big(\ \underset{\uparrow}{r^i_{\hat{a}\hat{b}}}\ , \{\ \underset{\substack{\uparrow \\ \langle \mathrm{GC}_j\rangle}}{\langle k^j_{w,0} \oplus r_{\hat{a}\hat{b}}\Delta^j\rangle_i}\ \}_{j>1},\ \underset{\substack{\uparrow \\ \langle AC\rangle}}{\langle r_{\hat{a}\hat{b}}\Delta^1\rangle}\ \Big) \qquad \forall(\hat{a},\hat{b}) \in \{0,1\}^2$$

masked output bit

Evaluator P_1:

- active key $\boldsymbol{k}_{a,\hat{a}} = \big(k^2_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta^2, \ldots, k^n_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta^n\big)$ decrypts $\mathrm{GC}_2[\hat{a},\hat{b}], \ldots, \mathrm{GC}_n[\hat{a},\hat{b}]$

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus \underbrace{(\lambda_a \oplus \alpha)}_{\hat{a}}\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \left(\lambda^i, \{\ \langle \lambda^i \Delta^j \rangle, \langle \lambda^j \Delta^i \rangle\ \}_{j \neq i}\right)$
- same Δ as MAC key and as FreeXOR offset
- parties hold $[\![r_{\hat{a}\hat{b}}]\!] = [\![\overbrace{(\lambda_a \oplus \hat{a})}^{\alpha}\overbrace{(\lambda_b \oplus \hat{b})}^{\beta} \oplus \lambda_c]\!] \qquad \forall (\hat{a}, \hat{b}) \in \{0,1\}^2$

$$\mathsf{GC}_i[\hat{a}, \hat{b}] = H(k^i_{a,\hat{a}}, k^i_{b,\hat{b}}) \oplus \left(\underbrace{r^i_{\hat{a}\hat{b}}}_{\text{masked output bit}}, \{\ \underbrace{\langle k^j_{w,0} \oplus r_{\hat{a}\hat{b}} \Delta^j \rangle_i}_{\langle \mathsf{GC}_j \rangle}\ \}_{j>1},\ \underbrace{\langle r_{\hat{a}\hat{b}} \Delta^1 \rangle}_{\langle AC \rangle} \right) \qquad \forall (\hat{a}, \hat{b}) \in \{0,1\}^2$$

Evaluator P_1:

- active key $\boldsymbol{k}_{a,\hat{a}} = \left(k^2_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta^2, \ldots, k^n_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta^n\right)$ decrypts $\mathsf{GC}_2[\hat{a}, \hat{b}], \ldots, \mathsf{GC}_n[\hat{a}, \hat{b}]$
- obtains $r_{\hat{a}\hat{b}} = \hat{c}$ and active key $\boldsymbol{k}_{c,\hat{c}}$ and checks correctness of $r_{\hat{a}\hat{b}} = \alpha\beta \oplus \lambda_c$

AND Gate

- every wire $k^i_{a,1} = k^i_{a,0} \oplus \underbrace{(\lambda_a \oplus \alpha)}_{\hat{a}}\Delta^i$

sharing between P_i and P_j

- use *authenticated* bit shares $[\![\lambda]\!] = \left(\lambda^i, \{\ \langle\lambda^i\Delta^j\rangle, \langle\lambda^j\Delta^i\rangle\ \}_{j\neq i}\right)$
- same Δ as MAC key and as FreeXOR offset
- parties hold $[\![r_{\hat{a}\hat{b}}]\!] = [\![\overbrace{(\lambda_a \oplus \hat{a})}^{\alpha}\overbrace{(\lambda_b \oplus \hat{b})}^{\beta} \oplus \lambda_c]\!] \qquad \forall(\hat{a},\hat{b}) \in \{0,1\}^2$

$$\mathrm{GC}_i[\hat{a},\hat{b}] = H(k^i_{a,\hat{a}}, k^i_{b,\hat{b}}) \oplus \left(r^i_{\hat{a}\hat{b}}\ , \{\ \langle k^j_{w,0} \oplus r_{\hat{a}\hat{b}}\Delta^j\rangle_i\ \}_{j>1},\ \langle r_{\hat{a}\hat{b}}\Delta^1\rangle \right) \qquad \forall(\hat{a},\hat{b}) \in \{0,1\}^2$$

masked output bit $\qquad \langle \mathrm{GC}_j\rangle \qquad \langle AC\rangle$

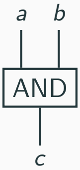
Evalu

Sending $r_{\hat{a}\hat{b}}$ and $r_{\hat{a}\hat{b}}\Delta^1$ can be removed at cost of additional online rounds [YWZ20]

- active key $\boldsymbol{k}_{a,\hat{a}} = (k_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta, \ldots, k_{a,0} \oplus (\lambda_a \oplus \alpha)\Delta)$ decrypts $\mathrm{GC}_2[\hat{a},\hat{b}], \ldots, \mathrm{GC}_n[\hat{a},\hat{b}]$
- obtains $r_{\hat{a}\hat{b}} = \hat{c}$ and active key $\boldsymbol{k}_{c,\hat{c}}$ and checks correctness of $r_{\hat{a}\hat{b}} = \alpha\beta \oplus \lambda_c$

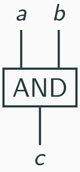
Our Row Reduction

Split AND equation into three parts: $\alpha\beta \oplus \lambda_c$



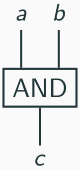
Our Row Reduction

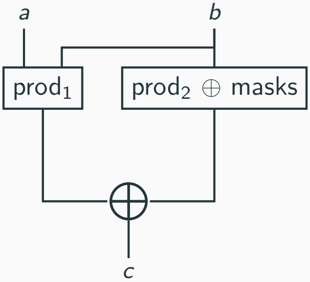
Split AND equation into three parts: $\alpha\beta \oplus \lambda_c = \underbrace{(\alpha \oplus \lambda_a)\beta}_{\text{prod}_1} \oplus \underbrace{(\beta \oplus \lambda_b)\lambda_a}_{\text{prod}_2} \oplus \underbrace{\lambda_a\lambda_b \oplus \lambda_c}_{\text{masks}}$



Our Row Reduction

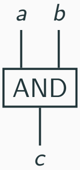
Split AND equation into three parts: $\alpha\beta \oplus \lambda_c = \underbrace{(\alpha \oplus \lambda_a)\beta}_{\text{prod}_1} \oplus \underbrace{(\beta \oplus \lambda_b)\lambda_a}_{\text{prod}_2} \oplus \underbrace{\lambda_a\lambda_b \oplus \lambda_c}_{\text{masks}}$

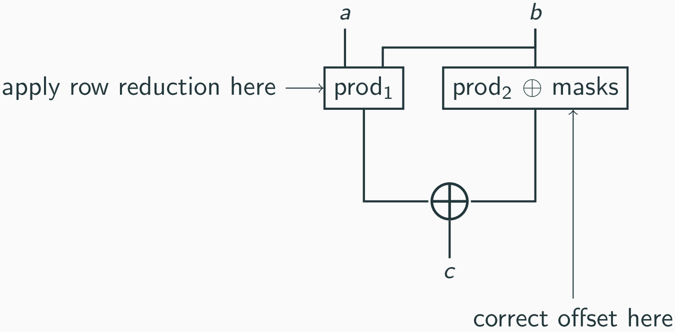




Our Row Reduction

Split AND equation into three parts: $\alpha\beta \oplus \lambda_c = \underbrace{(\alpha \oplus \lambda_a)\beta}_{\text{prod}_1} \oplus \underbrace{(\beta \oplus \lambda_b)\lambda_a}_{\text{prod}_2} \oplus \underbrace{\lambda_a\lambda_b \oplus \lambda_c}_{\text{masks}}$





The prod_1 gadget $(\alpha \oplus \lambda_a)\beta$

(adapted multi-party version of the length-2 one-hot garbling construction [HK21])

- the parties hold $[\![r_1]\!] = [\![\lambda_a]\!]$ and compute

The prod_1 gadget $(\alpha \oplus \lambda_a)\beta$

(adapted multi-party version of the length-2 one-hot garbling construction [HK21])

- the parties hold $[\![r_1]\!] = [\![\lambda_a]\!]$ and compute

$$C_1^i = H(k_{a,0}^i) \oplus H(k_{a,1}^i) \oplus \left(r_1^i,\ \{\langle k_{b,0}^j \oplus r_1 \Delta^j \rangle\}_{j>1},\ \langle r_1 \Delta^1 \rangle_i\right)$$

The prod_1 gadget $(\alpha \oplus \lambda_a)\beta$

(adapted multi-party version of the length-2 one-hot garbling construction [HK21])

- the parties hold $[\![r_1]\!] = [\![\lambda_a]\!]$ and compute

$$C_1^i = H(k_{a,0}^i) \oplus H(k_{a,1}^i) \oplus \left(r_1^i, \{\langle k_{b,0}^j \oplus r_1 \Delta^j \rangle\}_{j>1}, \langle r_1 \Delta^1 \rangle_i \right)$$

- evaluator P_1 computes $H(k_{a,\hat{a}}^i) \oplus \hat{a} C_1^i \oplus \hat{a} k_{b,\hat{b}}^i = \langle \mathsf{prod}_1 \rangle_i, \{\langle k^j \rangle_i\}_{j>1}, \langle \mathsf{prod}_1 \Delta^1 \rangle_i$

The prod_1 gadget $(\alpha \oplus \lambda_a)\beta$

(adapted multi-party version of the length-2 one-hot garbling construction [HK21])

- the parties hold $[\![r_1]\!] = [\![\lambda_a]\!]$ and compute

$$C_1^i = H(k_{a,0}^i) \oplus H(k_{a,1}^i) \oplus \left(r_1^i,\ \{\langle k_{b,0}^j \oplus r_1 \Delta^j \rangle\}_{j>1},\ \langle r_1 \Delta^1 \rangle_i\right)$$

- evaluator P_1 computes $H(k_{a,\hat{a}}^i) \oplus \hat{a} C_1^i \oplus \hat{a} k_{b,\hat{b}}^i = \langle \mathsf{prod}_1 \rangle_i,\ \{\langle k^j \rangle_i\}_{j>1},\ \langle \mathsf{prod}_1 \Delta^1 \rangle_i$
- but: $k^j = \mathsf{prod}_1 \Delta^j \oplus \sum_i H(k_{a,0}^i)$ with undesired offset
- the offset will be corrected in the prod_2 gadget

The prod_2 gadget

is a regular unary gate with added masks and offset terms

- the parties hold $[\![r_{2,\hat{b}}]\!] = [\![\hat{b} \cdot \lambda_a \oplus \lambda_a \lambda_b \oplus \lambda_c]\!]$ for $\hat{b} \in \{0, 1\}$

The prod_2 gadget

is a regular unary gate with added masks and offset terms

- the parties hold $[\![r_{2,\hat{b}}]\!] = [\![\hat{b} \cdot \lambda_a \oplus \lambda_a\lambda_b \oplus \lambda_c]\!]$ for $\hat{b} \in \{0,1\}$

$$C^i_{2,0} = H(k^i_{b,0}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,0}, \{\langle k^j_{c,0} \oplus r_{2,0}\Delta^j\rangle_i\}_{j>1}, \langle r_{2,0}\Delta^1\rangle_i)$$
$$C^i_{2,1} = H(k^i_{b,1}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,1}, \{\langle k^j_{c,1} \oplus r_{2,1}\Delta^j\rangle_i\}_{j>1}, \langle r_{2,1}\Delta^1\rangle_i)$$

The prod_2 gadget

is a regular unary gate with added masks and offset terms

- the parties hold $[\![r_{2,\hat{b}}]\!] = [\![\hat{b} \cdot \lambda_a \oplus \lambda_a \lambda_b \oplus \lambda_c]\!]$ for $\hat{b} \in \{0,1\}$

$$C^i_{2,0} = H(k^i_{b,0}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,0}, \{\langle k^j_{c,0} \oplus r_{2,0}\Delta^j \rangle_i\}_{j>1}, \langle r_{2,0}\Delta^1 \rangle_i)$$
$$C^i_{2,1} = H(k^i_{b,1}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,1}, \{\langle k^j_{c,1} \oplus r_{2,1}\Delta^j \rangle_i\}_{j>1}, \langle r_{2,1}\Delta^1 \rangle_i)$$

- evaluator P_1 computes regular decryption of $C^i_{2,\hat{b}}$ as $H(k^i_{b,\hat{b}}) \oplus C^i_{2,\hat{b}}$

The prod_2 gadget

is a regular unary gate with added masks and offset terms

- the parties hold $[\![r_{2,\hat{b}}]\!] = [\![\hat{b} \cdot \lambda_a \oplus \lambda_a \lambda_b \oplus \lambda_c]\!]$ for $\hat{b} \in \{0,1\}$

$$C^i_{2,0} = H(k^i_{b,0}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,0}, \{\langle k^j_{c,0} \oplus r_{2,0}\Delta^j \rangle_i\}_{j>1}, \langle r_{2,0}\Delta^1 \rangle_i)$$
$$C^i_{2,1} = H(k^i_{b,1}) \oplus H(k^i_{a,0}) \oplus (r^i_{2,1}, \{\langle k^j_{c,1} \oplus r_{2,1}\Delta^j \rangle_i\}_{j>1}, \langle r_{2,1}\Delta^1 \rangle_i)$$

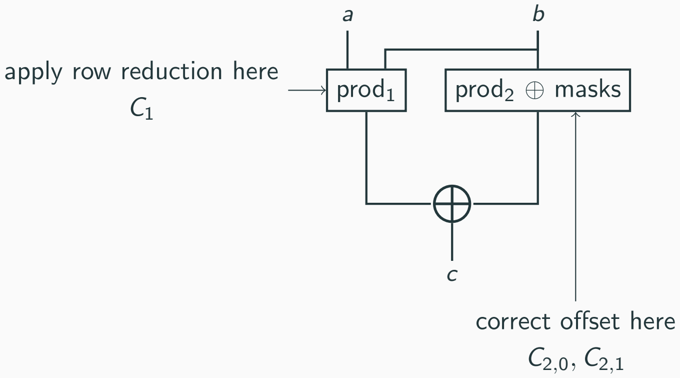
- evaluator P_1 computes regular decryption of $C^i_{2,\hat{b}}$ as $H(k^i_{b,\hat{b}}) \oplus C^i_{2,\hat{b}}$

Finally: XORing the prod_1 and prod_2 gadgets yields: $\langle \hat{c} \rangle_i$, $k^i_{c,\hat{c}}$ and $\langle \hat{c}\Delta^1 \rangle_i$

Summary

The general idea:

Split AND equation into three parts: $\alpha\beta \oplus \lambda_c = \underbrace{(\alpha + \lambda_a)\beta}_{\text{prod}_1} \oplus \underbrace{(\beta \oplus \lambda_b)\lambda_a}_{\text{prod}_2} \oplus \underbrace{\lambda_a\lambda_b \oplus \lambda_c}_{\text{masks}}$



Improved Preprocessing

we extend the approach of [DILO22] from 2 to $n \geq 3$ parties

Func. Indep. Prep. authenticated mult. triples in $\mathbb{F}_2$ $\longrightarrow$ **Func. Dep. Prep.** beaver multiplication of $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$ $\longrightarrow$ **Garble/Eval** use $[\![\lambda_a \lambda_b]\!]$ to construct GC

Improved Preprocessing

we extend the approach of [DILO22] from 2 to $n \geq 3$ parties

Func. Indep. Prep.		**Func. Dep. Prep.**		**Garble/Eval**
authenticated mult. triples in $\mathbb{F}_2$	$\longrightarrow$	beaver multiplication of $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$	$\longrightarrow$	use $[\![\lambda_a \lambda_b]\!]$ to construct GC

[YWZ20] expensive $\mathcal{O}(\kappa|C|)$ cheap $|C|$

Improved Preprocessing

we extend the approach of [DILO22] from 2 to $n \geq 3$ parties

	Func. Indep. Prep. authenticated mult. triples in $\mathbb{F}_2$	$\rightarrow$	**Func. Dep. Prep.** beaver multiplication of $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$	$\rightarrow$	**Garble/Eval** use $[\![\lambda_a \lambda_b]\!]$ to construct GC
[YWZ20]	expensive $\mathcal{O}(\kappa \lvert C \rvert)$		cheap $\lvert C \rvert$		
[DILO22] (and ours)	cheap (sublinear in $\lvert C \rvert$)		ok $\rho \lvert C \rvert$ $\rho < \kappa$		

Improved Preprocessing

we extend the approach of [DILO22] from 2 to $n \geq 3$ parties

Func. Indep. Prep. authenticated mult. triples in $\mathbb{F}_2$ $\longrightarrow$ **Func. Dep. Prep.** beaver multiplication of $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$ $\longrightarrow$ **Garble/Eval** use $[\![\lambda_a \lambda_b]\!]$ to construct GC

[YWZ20]	expensive $\mathcal{O}(\kappa\|C\|)$	cheap $\|C\|$
[DILO22] (and ours)	cheap (sublinear in $\|C\|$)	ok $\rho\|C\|$ $\rho < \kappa$

for this we

1. design a PCG for correlation $\langle x \rangle, \langle y \rangle, \langle xy \rangle$ with authentication values $\{\langle x\alpha^i \rangle, \langle y\alpha^i \rangle, \langle xy\alpha^i \rangle\}_i$ in $\mathbb{F}_{2^\rho}$

Improved Preprocessing

we extend the approach of [DILO22] from 2 to $n \geq 3$ parties

Func. Indep. Prep.	$\rightarrow$	**Func. Dep. Prep.**	$\rightarrow$	**Garble/Eval**
authenticated mult. triples in $\mathbb{F}_2$		beaver multiplication of $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$		use $[\![\lambda_a \lambda_b]\!]$ to construct GC

[YWZ20]	expensive $\mathcal{O}(\kappa\|C\|)$	cheap $\|C\|$
[DILO22] (and ours)	cheap (sublinear in $\|C\|$)	ok $\rho\|C\|$ $\rho < \kappa$

for this we

1. design a PCG for correlation $\langle x \rangle, \langle y \rangle, \langle xy \rangle$ with authentication values $\{\langle x\alpha^i \rangle, \langle y\alpha^i \rangle, \langle xy\alpha^i \rangle\}_i$ in $\mathbb{F}_{2^\rho}$
2. perform a "large field" multiplication for $[\![\lambda_a]\!] \times [\![\lambda_b]\!]$
3. "key switch" to authentication in $\mathbb{F}_{2^\kappa}$ using n-party VOLE

see full paper for details

Results

Full Row Reduction:

Circuit size:

- $3n\kappa$ (for [HSS17]-style)
- $3(n-1)\kappa$ (for authenticated garbling [WRK17,YWZ20])

- 25% to 43% smaller circuit compared to [HSS17], [WRK17] and [YWZ20]

Execution time in ms for AES-128 circuit

Parties	4	8	12
[WRK17]	223	423	629
Ours	168	359	540

Improved Preprocessing:

Authenticated field triples with $\mathcal{O}(n^3\sqrt{|C|})$ comm.

Mask preparation with $\mathcal{O}(2\rho|C|)$ comm.

- $\times 6$ lower comm. cost compared to [HSS17]
- $\times 2.2$ lower comm. cost compared to [YWZ20]

Thank you!

eprint 2025/829

github.com/zama-ai/copz25-code

erik.pohle@esat.kuleuven.be

References

[NPS99] Naor, Pinkas and Sumner. Privacy Preserving Auctions and Mechanism Design. 1st ACM Conference on Electronic Commerce.

[HSS17] Hazay, Scholl and Soria-Vazquez. Low Cost Constant Round MPC Combining BMR and Oblivious Transfer. Asiacrypt'17.

[WRK17] Wang, Ranellucci and Katz. Global-scale secure multiparty computation. CCS'17.

[YWZ20] Yang, Wang and Zhang. More Efficient MPC from Improved Triple Generation and Authenticated Garbling. CCS'20.

[HK21] Heath and Kolesnikov. One Hot Garbling. CCS'21.

[DILO22] Dittmer, Ishai, Lu and Ostrovsky. Authenticated Garbling from Simple Correlations. CRYPTO'22.