Practical Garbled Circuit Optimizations

Collaborators: David Evans / Vlad Kolesnikov / Payman Mohassel / Samee Zahur
Garbled circuit framework [Yao86]
Garbled circuit framework [Yao86]

Garbling a circuit:
▶ Pick random labels $W_0, W_1$ on each wire
▶ "Encrypt" truth table of each gate
▶ Garbled circuit $\equiv$ all encrypted gates
▶ Garbled encoding $\equiv$ one label per wire

Garbled evaluation:
▶ Only one ciphertext per gate is decryptable
▶ Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random **labels** $W_0, W_1$ on each wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random **labels** \(W_0, W_1\) on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** \(\equiv\) all encrypted gates
- **Garbled encoding** \(\equiv\) one label per wire
Garbled circuit framework

\[ A_0, A_1, E_0, E_1, B_0, B_1, F_0, F_1, C_0, C_1, G_0, G_1, D_0, D_1, H_0, H_1, I_0, I_1 \]

Garbling a circuit:
- Pick random labels \( W_0, W_1 \) on each wire
- “Encrypt” truth table of each gate
- Garbled circuit \( \equiv \) all encrypted gates
- Garbled encoding \( \equiv \) one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable

\[ E_{A_0,B_0}(E_0), E_{A_0,B_1}(E_1), E_{A_1,B_0}(E_0), E_{A_1,B_1}(E_0), E_{A_0,B_0}(F_0), E_{A_0,B_1}(F_1), E_{A_1,B_0}(F_0), E_{A_1,B_1}(F_1), E_{C_0,D_0}(G_0), E_{C_0,D_1}(G_1), E_{C_1,D_0}(G_0), E_{C_1,D_1}(G_0), E_{F_0,G_0}(H_0), E_{F_0,G_1}(H_1), E_{F_1,G_0}(H_0), E_{F_1,G_1}(H_0), E_{E_0,H_0}(I_0), E_{E_0,H_1}(I_1), E_{E_1,H_0}(I_1), E_{E_1,H_1}(I_1) \]
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
- **Garbled encoding** $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random **labels** $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
- **Garbled encoding** $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- Garbled circuit $\equiv$ all encrypted gates
- Garbled encoding $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- Garbled circuit $\equiv$ all encrypted gates
- Garbled encoding $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Applications: 2PC and more

\[ f(x, y) = g(x, y) \]

Private function evaluation, zero-knowledge proofs, encryption with key-dependent message security, randomized encodings, secure outsourcing, one-time programs, ...
Applications: 2PC and more

Garbling is a fundamental primitive \[\text{[BellareHoangRogaway12]}\]

Garbled circuit \( f \)
garbled input \( x \),
output wire labels

\( x \)

\( y \)
Applications: 2PC and more

Garbling is a fundamental primitive \cite{BellareHoangRogaway12}
Applications: 2PC and more

Garbling is a fundamental primitive \[\text{BellareHoangRogaway12}\]

$$f(x, y)$$

$$\text{garbled circuit } f$$

$$\text{garbled input } x, \text{ output wire labels}$$

$$\text{input wire labels}$$

$$\text{OT}$$

$$\text{garbled } y$$

$$y$$

$$x$$
Applications: 2PC and more

Garbling is a fundamental primitive \[\text{BellareHoangRogaway12}\]

Private function evaluation, zero-knowledge proofs, encryption with key-dependent message security, randomized encodings, secure outsourcing, one-time programs, . . .
Applications: 2PC and more

Garbling is a fundamental primitive [BellareHoangRogaway12]
Syntax [BellareHoangRogaway12]

Security properties:

Privacy: \((F, X, d)\) reveals nothing beyond \(f(x)\)

Obliviousness: \((F, X)\) reveals nothing

Authenticity: given \((F, X)\), hard to find \(\tilde{Y}\) that decodes \(\neq \{f(x), \bot\}\)
Syntax [BellareHoangRogaway12]

Security properties:
- **Privacy:** \((F, X, d)\) reveals nothing beyond \(f(x)\)
- **Obliviousness:** \((F, X)\) reveals nothing
- **Authenticity:** given \((F, X)\), hard to find \(\tilde{Y}\) that decodes \(\prec\{f(x), \perp}\)
Security properties:

Privacy: \((F, X, d)\) reveals nothing beyond \(f(x)\)

Obliviousness: \((F, X)\) reveals nothing

Authenticity: given \((F, X)\), hard to find \(\tilde{Y}\) that decodes \(\notin \{f(x), \bot\}\)
Parameters to optimize

hardness assumption

computation

size
Parameters to optimize

- size
- computation
- hardness assumption
Average bits per garbled gate

Graph showing the average bits per garbled gate over time for different years: 1986, 1990, 1999, 2008, 2009, 2014, 2015. The lines represent different cryptographic algorithms: DES, SHA256, SHA1, and AES. The graph indicates a decrease in the average bits per garbled gate over time, with a prediction that by 2026, all garbled circuits will have zero size.
Average bits per garbled gate

[Yao, Goldreich, Micali, Wigderson] [Beaver, Micali, Rogaway] [Naor, Pinkas, Sumner] [Kolesnikov, Mohassel, Rosulek] [Pinkas, Schneider, Smart, Williams] [Kolesnikov, Schneider] [Zahur, Rosulek, Evans]


DES SHA256 SHA1 AES

Prediction: by 2026, all garbled circuits will have zero size.
Average bits per garbled gate

Prediction: by 2026, all garbled circuits will have zero size.
Murky beginnings [Yao86]

- Position in this list leaks semantic value
Murky beginnings [Yao86]

- Position in this list leaks semantic value
Position in this list leaks semantic value $\Rightarrow$ permute ciphertexts
Murky beginnings \cite{Yao86}

\[
\begin{aligned}
A_0, A_1 & \quad C_0, C_1 \\
B_0, B_1 & \\
\end{aligned}
\]

- Position in this list leaks semantic value $\implies$ permute ciphertexts
- Need to detect [in]correct decryption

\[\text{\textbf{E}}_{A_0, B_0}(C_0)\]
\[\text{\textbf{E}}_{A_0, B_1}(C_1)\]
\[\text{\textbf{E}}_{A_1, B_0}(C_0)\]
\[\text{\textbf{E}}_{A_1, B_1}(C_0)\]
Murky beginnings [Yao86]

Position in this list leaks semantic value \( \implies \) permute ciphertexts

Need to detect [in]correct decryption

(Apparently) no one knows exactly what Yao had in mind:

- \( E_{K_0,K_1}(M) = \langle E(K_0,S_0), E(K_1,S_1) \rangle \) where \( S_0 \oplus S_1 = M \)

- \( E_{K_0,K_1}(M) = E(K_1,E(K_0,M)) \) [GoldreichMicaliWigderson87]

[GoldreichMicaliWigderson87] [LindellPinkas09]
Permute-and-Point [BeaverMicaliRogaway90]

Randomly assign \((A_0, A_1)\) or \((B_0, B_1)\) to each pair of wire labels.

Include color in the wire label (e.g., as last bit).

Order the 4 ciphertexts canonically, by color of keys.

Evaluate by decrypting ciphertext indexed by your colors.

Can use one-time-secure symmetric encryption!
Permute-and-Point [BeaverMicaliRogaway90]

- Randomly assign (⃗, ⃗) or (⃗, ⃗) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
Permuted-and-Point [BeaverMicaliRogaway90]

- Randomly assign \((\bullet, \bullet)\) or \((\bullet, \circ)\) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys

\[ \text{Randomly assign } (\bullet, \bullet) \text{ or } (\bullet, \circ) \text{ to each pair of wire labels} \]
\[ \text{Include color in the wire label (e.g., as last bit)} \]
\[ \text{Order the 4 ciphertexts canonically, by color of keys} \]

\[ \mathcal{E}_{A_0, B_0}(C_0) \]
\[ \mathcal{E}_{A_0, B_1}(C_1) \]
\[ \mathcal{E}_{A_1, B_0}(C_0) \]
\[ \mathcal{E}_{A_1, B_1}(C_0) \]
Permute-and-Point \[\text{[BeaverMicaliRogaway90]}\]

- Randomly assign \((\bullet, \bullet)\) or \((\bullet, \bullet)\) to each pair of wire labels.
- Include color in the wire label (e.g., as last bit).
- Order the 4 ciphertexts canonically, by color of keys.

\[\begin{align*}
\text{\textcolor{red}{A}_0, A_1} & \quad \text{\textcolor{red}{C}_0, C_1} \\
\textcolor{red}{B}_0, B_1 \\
\end{align*}\]
Permute-and-Point [BeaverMicaliRogaway90]

- Randomly assign (●, ●) or (●, ○) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys
- Evaluate by decrypting ciphertext indexed by your colors
Permuted Point-Point [BeaverMicaliRogaway90]

- Randomly assign `(●,●)` or `(●,○)` to each pair of wire labels.
- Include color in the wire label (e.g., as last bit).
- Order the 4 ciphertexts canonically, by color of keys.
- Evaluate by decrypting ciphertext indexed by your colors.

Can use one-time-secure symmetric encryption!
Permute-and-Point [BeaverMicaliRogaway90]

Randomly assign \((\bullet, \bullet)\) or \((\bullet, \circ)\) to each pair of wire labels

Include color in the wire label (e.g., as last bit)

Order the 4 ciphertexts canonically, by color of keys

Evaluate by decrypting ciphertext indexed by your colors

Can use **one-time-secure** symmetric encryption!
Computational cost of garbling

\[ E_{A,B}(C): \]

PRF(\(A\), gateID) \(\oplus\) PRF(\(B\), gateID) \(\oplus\) C

\[ \sim 6s \text{ [extrapolated]} \]

time from Fairplay [MNPS04]: PRF = SHA256

[NaorPinkasSumner99]
Computational cost of garbling

\[ 2 \text{ hash} \gg 1 \text{ hash} \]

\[ E_{A,B}(C): \]

PRF(\(A, \text{gateID}\)) \oplus PRF(\(B, \text{gateID}\)) \oplus C

\[ \sim6s \text{ [extrapolated]} \]

time from Fairplay [MNPS04]: PRF = SHA256

H(\(A||B||\text{gateID}\)) \oplus C

\[ 0.15s \]

time from [sS12]; H = SHA256
Computational cost of garbling

2 hash $\gg$ 1 hash $\gg$ 1 block cipher

$E_{A,B}(C)$:

PRF($A$, gateID) $\oplus$ PRF($B$, gateID) $\oplus$ C
[NaorPinkasSumner99] $\sim$ 6s [extrapolated]

time from Fairplay [MNPS04]: PRF = SHA256

H($A||B||$gateID) $\oplus$ C
[LindellPinkasSmart08] 0.15s

time from [sS12]; H = SHA256

AES256($A||B$, gateID) $\oplus$ C
[shelatShen12] 0.12s
Computational cost of garbling

\[ 2 \text{ hash} \gg 1 \text{ hash} \gg 1 \text{ block cipher} \gg 1 \text{ block cipher w/o key schedule} \]

\[ E_{A,B}(C): \]

\[ \Prf(A, \text{gateID}) \oplus \Prf(B, \text{gateID}) \oplus C \]

\[ \Prf = \text{SHA256} \]

\[ \text{time from Fairplay } [MNPS04]: \Prf = \text{SHA256} \]

\[ \sim 6s \text{ [extrapolated]} \]

\[ H(A \parallel B \parallel \text{gateID}) \oplus C \]

\[ H = \text{SHA256} \]

\[ \text{time from } [sS12]; \]

\[ 0.15s \]

\[ \text{AES256} (A \parallel B, \text{gateID}) \oplus C \]

\[ \text{time from } [sS12]; \]

\[ 0.12s \]

\[ \text{AES} (\text{const}, K) \oplus K \oplus C \]

\[ \text{where } K = 2A \oplus 4B \oplus \text{gateID} \]

\[ 0.0003s \]

\[ \text{[BellareHoangKeelvedhiRogaway13]} \]
Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size ($\times \lambda$)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4</td>
<td>4/8</td>
<td>1/2</td>
<td>hash/PRF</td>
</tr>
</tbody>
</table>
Garbled Row Reduction [NaorPinkasSumner99]

What wire label will be payload of 1st (••) ciphertext? Choose that label so that 1st ciphertext is 0. No need to include 1st ciphertext in garbled gate. Evaluate as before, but imagine ciphertext 0 if you got ••.
Garbled Row Reduction

\[ C_0 \leftarrow \{0, 1\}^n \]
\[ C_1 \leftarrow \{0, 1\}^n \]

What wire label will be payload of 1st (••) ciphertext?

Choose that label so that 1st ciphertext is 0

No need to include 1st ciphertext in garbled gate

Evaluate as before, but imagine ciphertext 0 if you got ••.
Garbled Row Reduction [NaorPinkasSumner99]

- What wire label will be payload of 1st (●●) ciphertext?

- No need to include 1st ciphertext in garbled gate
- Evaluate as before, but imagine ciphertext 0 if you got ●●.
Garbled Row Reduction

\[ C_0 \leftarrow \{0, 1\}^n \]
\[ C_1 = \mathbb{E}_{A_0, B_1}^{-1}(0^n) \]

What wire label will be payload of 1st (●●) ciphertext?
Choose that label so that 1st ciphertext is 0^n
Garbled Row Reduction

\[ C_0 \leftarrow \{0, 1\}^n \]
\[ C_1 = E_{A_0, B_1}^{-1}(0^n) \]

What wire label will be payload of 1st (●●) ciphertext?

Choose that label so that 1st ciphertext is 0^n
Garbled Row Reduction

\[ C_0 \leftarrow \{0, 1\}^n \]
\[ C_1 = E_{A_0, B_1}^{-1}(0^n) \]

- What wire label will be payload of 1st (● ●) ciphertext?
- Choose that label so that 1st ciphertext is \( 0^n \)
- No need to include 1st ciphertext in garbled gate
Garbled Row Reduction

[NaorPinkasSumner99]

\[ C_0 \leftarrow \{0, 1\}^n \]
\[ C_1 = \mathbb{E}_{A_0, B_1}^{-1}(0^n) \]

What wire label will be payload of 1st (●●) ciphertext?
Choose that label so that 1st ciphertext is \(0^n\)
No need to include 1st ciphertext in garbled gate
Evaluate as before, but imagine ciphertext \(0^n\) if you got ●●.
## Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size ($\times \lambda$)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4</td>
<td>4/8</td>
<td>1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3</td>
<td>4/8</td>
<td>1/2</td>
<td>hash/PRF</td>
</tr>
</tbody>
</table>
Free XOR

\[ A_0, A_1 \]

\[ B_0, B_1 \]

\[ C_0, C_1 \]

\[ \begin{align*}
C \leftarrow \{0, 1\} \\
A \oplus B = A \oplus \overline{B}
\end{align*} \]

Wire's offset \( \equiv \) XOR of its two labels

\[ \Delta \]

Choose all wires to have same (secret) offset

Choose \( \overline{f.sc/a.sc/l.sc/s.sc/e.sc} \) output = \( \overline{f.sc/a.sc/l.sc/s.sc/e.sc} \) input

Evaluated by \( \times.sc/o.sc/r.sc \)ing input wire labels (no crypto)
Wire’s offset $\equiv$ XOR of its two labels
Free XOR [KolesnikovSchneider08]

- Wire’s offset $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
Free XOR \cite{KolesnikovSchneider08}

- Wire’s offset $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
Free XOR [KolesnikovSchneider08]

- Wire’s offset $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose FALSE output $=\text{FALSE}$ input $\oplus \text{FALSE}$ input
Free XOR  

$C := A \oplus B$

- Wire’s offset $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose $\text{FALSE}$ output $= \text{FALSE}$ input $\oplus \text{FALSE}$ input
- Evaluate by XORing input wire labels (no crypto)
Free XOR

\[ C := A \oplus B \]

- Wire’s **offset** $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose FALSE output $= FALSE$ input $\oplus$ FALSE input
- Evaluate by xorring input wire labels (no crypto)
Free XOR  

$C := A \oplus B$

$A, \ A \oplus \Delta$

$B, \ B \oplus \Delta$

$C, \ C \oplus \Delta$

- Wire’s offset $\equiv$ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose FALSE output $= \text{FALSE input} \oplus \text{FALSE input}$
- Evaluate by xoring input wire labels (no crypto)
Freedom at a cost...

\[ A, A \oplus \Delta \rightarrow C, C \oplus \Delta \]

\[
\begin{align*}
E_A, B (C) \\
E_A, B \oplus \Delta (C \oplus \Delta) \\
E_{A \oplus \Delta, B} (C) \\
E_{A \oplus \Delta, B \oplus \Delta} (C)
\end{align*}
\]

- Still need to garble AND gates

\[ \text{ChoiKatzKumaresanZhou12} \]
Freedom at a cost . . .

\[ C \leftarrow \{0, 1\}^n \]

\[ A, A \oplus \Delta \]
\[ B, B \oplus \Delta \]
\[ C, C \oplus \Delta \]

- Still need to garble AND gates
- Compatible with garbled row-reduction
Freedom at a cost...

\[ C := \mathbb{E}_{A,B}^{-1}(0^n) \]

- Still need to garble AND gates
- Compatible with garbled row-reduction
Freedom at a cost...

- Still need to garble AND gates
- Compatible with garbled row-reduction
- Secret $\Delta$ used in key and payload of ciphertexts!
Freedom at a cost...

\[ A, A \oplus \Delta,\quad B, B \oplus \Delta \quad \xrightarrow{\text{AND gate}} \quad C, C \oplus \Delta \]

- Still need to garble AND gates
- Compatible with garbled row-reduction
- Secret $\Delta$ used in key and payload of ciphertexts!
- Requires related-key + circularity assumption [ChoiKatzKumaresanZhou12]
## Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size (×λ)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR AND</td>
<td>XOR AND</td>
<td>XOR AND</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4 4</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>GRR3</td>
<td>3 3</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0 3</td>
<td>0 4</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
</tbody>
</table>
Garbled gates with only 2 ciphertexts!
Row reduction ++  [PinkasSchneiderSmartWilliams09]

Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

\[
K_1 = \mathbb{B}^{-1}_{A_0, B_0}(0^n)
\]
\[
K_2 = \mathbb{B}^{-1}_{A_0, B_1}(0^n)
\]
\[
K_3 = \mathbb{B}^{-1}_{A_1, B_0}(0^n)
\]
\[
K_4 = \mathbb{B}^{-1}_{A_1, B_1}(0^n)
\]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

\[
\begin{align*}
K_1 &= E_{A_0, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
K_2 &= E_{A_0, B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \\
K_3 &= E_{A_1, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
K_4 &= E_{A_1, B_1}^{-1}(0^n) \leadsto \text{learn } C_0
\end{align*}
\]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

  \[ K_1 = \mathbb{E}_{A_0, B_0}^{-1} (0^n) \sim \text{learn } C_0 \]
  \[ K_2 = \mathbb{E}_{A_0, B_1}^{-1} (0^n) \sim \text{learn } C_1 \]
  \[ K_3 = \mathbb{E}_{A_1, B_0}^{-1} (0^n) \sim \text{learn } C_0 \]
  \[ K_4 = \mathbb{E}_{A_1, B_1}^{-1} (0^n) \sim \text{learn } C_0 \]
Row reduction ++  [PinkasSchneiderSmartWilliams09]

Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

  \[ K_1 = \mathbb{E}_{A_0,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \]
  \[ K_2 = \mathbb{E}_{A_0,B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \]
  \[ K_3 = \mathbb{E}_{A_1,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \]
  \[ K_4 = \mathbb{E}_{A_1,B_1}^{-1}(0^n) \leadsto \text{learn } C_0 \]

\[ P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4) \]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  \[ K_1 = \mathbb{E}_A^{-1}(0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_2 = \mathbb{E}_A^{-1}(0^n) \leadsto \text{learn } C_1 \]
  
  \[ K_3 = \mathbb{E}_A^{-1}(0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_4 = \mathbb{E}_A^{-1}(0^n) \leadsto \text{learn } C_0 \]

\[ P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4) \]

\[ (2, K_2), (5, P(5)), (6, P(6)) \]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

\[
K_1 = \mathcal{E}_{A_0, B_0}^{-1} (0^n) \leadsto \text{learn } C_0
\]
\[
K_2 = \mathcal{E}_{A_0, B_1}^{-1} (0^n) \leadsto \text{learn } C_1
\]
\[
K_3 = \mathcal{E}_{A_1, B_0}^{-1} (0^n) \leadsto \text{learn } C_0
\]
\[
K_4 = \mathcal{E}_{A_1, B_1}^{-1} (0^n) \leadsto \text{learn } C_0
\]

\[P = \text{uniq deg-2 poly thru (1, } K_1), (3, K_3), (4, K_4)\]
\[Q = \text{uniq deg-2 poly thru (2, } K_2), (5, P(5)), (6, P(6))\]
Garbled gates with only 2 ciphertexts!

Evaluator can know exactly one of:

- \( K_1 = \mathbb{E}_{A_0,B_0}^{-1} (0^n) \) to learn \( C_0 \)
- \( K_2 = \mathbb{E}_{A_0,B_1}^{-1} (0^n) \) to learn \( C_1 \)
- \( K_3 = \mathbb{E}_{A_1,B_0}^{-1} (0^n) \) to learn \( C_0 \)
- \( K_4 = \mathbb{E}_{A_1,B_1}^{-1} (0^n) \) to learn \( C_0 \)

\( C_0 = P(0); \ C_1 = Q(0) \)

\( P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4) \)

\( Q = \text{uniq deg-2 poly thru } (2, K_2), (5, P(5)), (6, P(6)) \)
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

\[
\begin{align*}
K_1 &= E_{A_0, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
K_2 &= E_{A_0, B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \\
K_3 &= E_{A_1, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
K_4 &= E_{A_1, B_1}^{-1}(0^n) \leadsto \text{learn } C_0
\end{align*}
\]

\[C_0 = P(0); \quad C_1 = Q(0)\]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  \begin{align*}
  K_1 &= \mathbb{E}_{A_0,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_2 &= \mathbb{E}_{A_0,B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \\
  K_3 &= \mathbb{E}_{A_1,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_4 &= \mathbb{E}_{A_1,B_1}^{-1}(0^n) \leadsto \text{learn } C_0
  \end{align*}

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

\[ C_0 = P(0); C_1 = Q(0) \]

\[ P = \text{uniq deg-2 poly thru} \]

\[ (1, K_1), (3, K_3), (4, K_4) \]

\[ Q = \text{uniq deg-2 poly thru} \]

\[ (2, K_2), (5, P(5)), (6, P(6)) \]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  \[
  K_1 = \mathbb{E}_{A_0, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_2 = \mathbb{E}_{A_0, B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \\
  K_3 = \mathbb{E}_{A_1, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_4 = \mathbb{E}_{A_1, B_1}^{-1}(0^n) \leadsto \text{learn } C_0
  \]

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

\[C_0 = P(0); \ C_1 = Q(0)\]

\[P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4)\]

\[Q = \text{uniq deg-2 poly thru } (2, K_2), (5, P(5)), (6, P(6))\]
Row reduction ++  [PinkasSchneiderSmartWilliams09]

Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:

  \[
  K_1 = \mathbb{E}_{A_0,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_2 = \mathbb{E}_{A_0,B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \\
  K_3 = \mathbb{E}_{A_1,B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \\
  K_4 = \mathbb{E}_{A_1,B_1}^{-1}(0^n) \leadsto \text{learn } C_0
  \]

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

\[
C_0 = P(0); \quad C_1 = Q(0)
\]

\[
P = \text{uniq deg-2 poly thru} \\
(1, K_1), (3, K_3), (4, K_4)
\]

\[
Q = \text{uniq deg-2 poly thru} \\
(2, K_2), (5, P(5)), (6, P(6))
\]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  $K_1 = \mathbb{E}^{-1}_{A_0, B_0} (0^n) \leadsto \text{learn } C_0$

  $K_2 = \mathbb{E}^{-1}_{A_0, B_1} (0^n) \leadsto \text{learn } C_1$

  $K_3 = \mathbb{E}^{-1}_{A_1, B_0} (0^n) \leadsto \text{learn } C_0$

  $K_4 = \mathbb{E}^{-1}_{A_1, B_1} (0^n) \leadsto \text{learn } C_0$

- Evaluate by interpolating poly thru $K_i$, $P(5)$ and $P(6)$

$C_0 = P(0); C_1 = Q(0)$

$P = \text{ uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4)$

$Q = \text{ uniq deg-2 poly thru } (2, K_2), (5, P(5)), (6, P(6))$
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  \[ K_1 = \mathbb{E}_{A_0, B_0}^{-1} (0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_2 = \mathbb{E}_{A_0, B_1}^{-1} (0^n) \leadsto \text{learn } C_1 \]
  
  \[ K_3 = \mathbb{E}_{A_1, B_0}^{-1} (0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_4 = \mathbb{E}_{A_1, B_1}^{-1} (0^n) \leadsto \text{learn } C_0 \]

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

\[ C_0 = P(0); C_1 = Q(0) \]

\[ P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4) \]

\[ Q = \text{uniq deg-2 poly thru } (2, K_2), (5, P(5)), (6, P(6)) \]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  \[ K_1 = E_{A_0, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \]
  \[ K_2 = E_{A_0, B_1}^{-1}(0^n) \leadsto \text{learn } C_1 \]
  \[ K_3 = E_{A_1, B_0}^{-1}(0^n) \leadsto \text{learn } C_0 \]
  \[ K_4 = E_{A_1, B_1}^{-1}(0^n) \leadsto \text{learn } C_0 \]

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

\[ C_0 = P(0); \ C_1 = Q(0) \]

\[ P = \text{uniq deg-2 poly thru } (1, K_1), (3, K_3), (4, K_4) \]

\[ Q = \text{uniq deg-2 poly thru } (2, K_2), (5, P(5)), (6, P(6)) \]
Garbled gates with only 2 ciphertexts!

- Evaluator can know exactly one of:
  
  \[ K_1 = E^{-1}_{A_0, B_0} (0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_2 = E^{-1}_{A_0, B_1} (0^n) \leadsto \text{learn } C_1 \]
  
  \[ K_3 = E^{-1}_{A_1, B_0} (0^n) \leadsto \text{learn } C_0 \]
  
  \[ K_4 = E^{-1}_{A_1, B_1} (0^n) \leadsto \text{learn } C_0 \]

- Evaluate by interpolating poly thru \( K_i, P(5) \) and \( P(6) \)

- **Incompatible** with Free-XOR: can’t ensure \( C_0 \oplus C_1 = \Delta \)
## Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size (×λ)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR AND</td>
<td>XOR AND</td>
<td>XOR AND</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4 4</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3 3</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0 3</td>
<td>0 4</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
<tr>
<td><strong>GRR2</strong></td>
<td><strong>2 2</strong></td>
<td><strong>4/8 4/8</strong></td>
<td><strong>1/2 1/2</strong></td>
<td><strong>PRF/hash</strong></td>
</tr>
</tbody>
</table>
FleXOR [KolesnikovMohassielRosulek14]

\[ A, A \oplus \Delta_1 \]
FleXOR [KolesnikovMohasselRosulek14]

- Translate to a new wire offset
FleXOR [KolesnikovMohasselRosulek14]

A, A ⊕ ∆₁ → A⁺, A⁺ ⊕ ∆₂

- Translate to a new wire offset (unary a ↦ a gate)
FleXOR [KolesnikovMohasselRosulek14]

- Translate to a new wire offset (unary $a \mapsto a$ gate)
FleXOR [KolesnikovMohasselRosulek14]

\[ A, A \oplus \Delta_1 \rightarrow A^*, A^* \oplus \Delta_2 \]

\[ E_A \quad (A^*) \]
\[ E_{A \oplus \Delta_1} \quad (A^* \oplus \Delta_2) \]

- Translate to a new wire offset (unary \( a \mapsto \textit{a gate} \))
FleXOR [KolesnikovMohasselRosulek14]

$A, A \oplus \Delta_1 \rightarrow A^*, A^* \oplus \Delta_2$

$A^* \leftarrow \{0, 1\}^n$

$E_A(A^*)$

$E_{A \oplus \Delta_1}(A^* \oplus \Delta_2)$

- Translate to a new wire offset (unary $a \mapsto a$ gate)
FleXOR \[\text{[KolesnikovMohasselRosulek14]}\]

\[A^* := E_A^{-1}(0^n)\]

\[
\begin{align*}
A, A \oplus \Delta_1 &\quad A^*, A^* \oplus \Delta_2 \\
E_A(A^*) &\quad E_A(A^* \oplus \Delta_2)
\end{align*}
\]

- Translate to a new wire offset (unary \(a \mapsto a\) gate)
A, A ⊕ Δ₁ → A*, A* ⊕ Δ₂

A* := \( E_A^{-1}(0^n) \)

\[ \begin{align*}
\mathbb{E}_{A \oplus \Delta_1}(A^* \oplus \Delta_2) & \quad 0^n \\
\end{align*} \]

- Translate to a new wire offset (unary \( a \mapsto a \) gate)
A, A ⊕ Δ₁ → A*, A* ⊕ Δ₂

\[ A^* := \mathcal{E}^{-1}_A(0^n) \]

\[ \mathcal{E}_{A \oplus \Delta_1}(A^* \oplus \Delta_2) \]

- Translate to a new wire offset (unary \( a \mapsto a \) gate) using 1 ciphertext
FleXOR [KolesnikovMohasselRosulek14]

\[ A, A \oplus \Delta_1 \xrightarrow{\Delta_1 \rightarrow \Delta_2} A^*, A^* \oplus \Delta_2 \]

\[ A^* := E^{-1}_A(0^n) \]

\[ E_{A \oplus \Delta_1}(A^* \oplus \Delta_2) \]

- Translate to a new wire offset (unary \( a \mapsto a \) gate) using 1 ciphertext
FleXOR \[\text{[KolesnikovMohasselRosulek14]}\]

\[
\begin{align*}
A, A \oplus \Delta_A \\
B, B \oplus \Delta_B \\
\rightarrow C, C \oplus \Delta_C
\end{align*}
\]

- Adjust inputs to target offset \(\Delta_C\) (1 ciphertext each), then XOR is free.
- If input wire already suitable, no need to adjust.
- Total cost: 0, 1 or 2 depending on how many \(\{\Delta_A, \Delta_B, \Delta_C\}\) distinct.

Combinatorial optimization problem:
- Choose an offset for each wire, minimizing total cost of XOR gates
- Subject to compatibility with 2-ciphertext row-reduction of AND gates
- (or) Subject to removing circularity property of free-XOR
Adjust inputs to target offset $\Delta_C$ (1 ciphertext each)
Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free
Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free

If input wire already suitable, no need to adjust
Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free
- If input wire already suitable, no need to adjust
- Total cost: 0, 1 or 2 depending on how many $\{\Delta_A, \Delta_B, \Delta_C\}$ distinct.
FleXOR \cite{KolesnikovMohasselRosulek14}

\[ A, A \oplus \Delta_A \xrightarrow{\Delta_A \rightarrow \Delta_C} C, C \oplus \Delta_C \]

\[ B, B \oplus \Delta_C \]

- Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free.
- If input wire already suitable, no need to adjust.
- Total cost: 0, 1 or 2 depending on how many $\{\Delta_A, \Delta_B, \Delta_C\}$ distinct.

**Combinatorial optimization problem**: Choose an offset for each wire, minimizing total cost of XOR gates.

- Subj. to compatibility with 2-ciphertext row-reduction of AND gates.
- (or) Subj. to removing circularity property of free-XOR.
# Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size ($\times \lambda$)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR AND</td>
<td>XOR AND</td>
<td>XOR AND</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4 4</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3 3</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0 3</td>
<td>0 4</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
<tr>
<td>GRR2</td>
<td>2 2</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>FleXOR</td>
<td>{0, 1, 2} 2</td>
<td>{0, 1, 2} 4</td>
<td>{0, 1, 2} 1</td>
<td>circ. hash</td>
</tr>
</tbody>
</table>
What if garbler knows in advance the truth value on one input wire?
What if garbler knows in advance the truth value on one input wire?

\[
A, A \oplus \Delta \quad C, C \oplus \Delta \\
B, B \oplus \Delta
\]
Half Gates [ZahurRosulekEvans15]

What if garbler knows in advance the truth value on one input wire?

\[
\begin{align*}
A, A \oplus \Delta & \quad \quad \quad \quad \quad \quad \quad \quad \quad C, C \oplus \Delta \\
B, B \oplus \Delta
\end{align*}
\]

if \( a = 0 \):

\[
\begin{array}{c|c}
0 & 0 \\
1 & 0 \\
\end{array}
\]

unary gate \( b \mapsto 0 \)
What if garbler knows in advance the truth value on one input wire?

if $a = 0$:

unary gate $b \mapsto 0$
What if garbler knows in advance the truth value on one input wire?

if $a = 0$:

- $\mathbb{E}_B(C)$
- $\mathbb{E}_{B\oplus \Delta}(C)$

unary gate $b \mapsto 0$
What if garbler knows in advance the truth value on one input wire?

Half Gates \[\text{[ZahurRosulekEvans15]}\]

Diagram:

\[
\begin{align*}
A, A \oplus \Delta & \quad & \text{Unary gate} \quad & \quad B, B \oplus \Delta \\
& \quad \quad \\quad \rightarrow \quad & \quad \quad \quad \quad \rightarrow \\
C, C \oplus \Delta & \quad & \text{Unary gate} \\
\end{align*}
\]
What if garbler knows in advance the truth value on one input wire?

If \( a = 0 \):
\[
\text{unary gate } b \mapsto \overline{0}
\]

If \( a = 1 \):
\[
\begin{array}{c|c}
0 & 0 \\
1 & 1 \\
\end{array}
\]

unary gate \( b \mapsto b \)
What if garbler knows in advance the truth value on one input wire?

If $a = 0$: unary gate $b \mapsto 0$

If $a = 1$:

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B \oplus \Delta$</td>
<td>$C \oplus \Delta$</td>
</tr>
</tbody>
</table>

Unary gate $b \mapsto b$
What if garbler knows in advance the truth value on one input wire?

If $a = 0$:

- unary gate $b \mapsto \overline{0}$

If $a = 1$:

- $E_B(C)$
- $E_{B \oplus \Delta}(C \oplus \Delta)$

Unary gate $b \leftrightarrow b$
What if garbler knows in advance the truth value on one input wire?

- If $a = 0$: 
  - $A \oplus \Delta$ 
  - $B, B \oplus \Delta$ 
  - $C, C \oplus \Delta$
  - $\mathcal{E}_B(C)$
  - $\mathcal{E}_{B \oplus \Delta}(C)$
  - Unary gate $b \mapsto 0$

- If $a = 1$: 
  - $A \oplus \Delta$ 
  - $B, B \oplus \Delta$ 
  - $C, C \oplus \Delta$
  - $\mathcal{E}_B(C \oplus \Delta)$
  - $\mathcal{E}_{B \oplus \Delta}(C \oplus \Delta)$
  - Unary gate $b \mapsto b$
What if garbler knows in advance the truth value on one input wire?

$A, A \oplus \Delta$  $C, C \oplus \Delta$

$B, B \oplus \Delta$

Fine print: permute ciphertexts with permute-and-point.
What if garbler knows in advance the truth value on one input wire?

\[ C \leftarrow \{0, 1\}^n \]

Diagram:

Input wires: \( A, A \oplus \Delta \) and \( B, B \oplus \Delta \)

Output wires: \( C, C \oplus \Delta \)

\[ E_B(C) \]
\[ E_{B \oplus \Delta}(C \oplus a\Delta) \]
What if garbler knows in advance the truth value on one input wire?

$C := \mathbb{E}_B^{-1}(0^n)$

$E_B(C)$

$E_{B \oplus \Delta}(C \oplus a\Delta)$
What if garbler knows in advance the truth value on one input wire?

\[ C := \mathbb{E}_B^{-1}(0^n) \]

\[ \begin{align*}
A, A \oplus \Delta & \quad B, B \oplus \Delta \\
C, C \oplus \Delta & \quad 0^n \quad \mathbb{E}_{B \oplus \Delta}(C \oplus a\Delta)
\end{align*} \]
What if garbler knows in advance the truth value on one input wire?

\[ C := \mathbb{E}_B^{-1}(0^n) \]

\[ \mathbb{E}_{B \oplus \Delta}(C \oplus a\Delta) \]

Fine print: permute ciphertexts with permute-and-point.
Half Gates [ZahurRosulekEvans15]

What if evaluator knows the truth value on one input wire?

$A, A \oplus \Delta \rightarrow C, C \oplus \Delta$

$B, B \oplus \Delta$

Fine print: no need for permute-and-point here
What if evaluator knows the truth value on one input wire?

A, A ⊕ Δ

B, B ⊕ Δ

C, C ⊕ Δ

Evaluator has B (knows f.sc/a.sc/l.sc/s.sc/e.sc):
⇒ should obtain C (f.sc/a.sc/l.sc/s.sc/e.sc)

Evaluator has B ⊕ Δ (knows t.sc/r.sc/u.sc/e.sc):
⇒ should be able to transfer truth value from "a" wire to "c" wire
▶ Suffices to learn A ⊕ C

Fine print: no need for permute-and-point here
Half Gates [ZahurRosulekEvans15]

What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows \text{FALSE}):  
\[ \Rightarrow \text{ should obtain } C \text{ (FALSE) } \]
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows FALSE):

$\Rightarrow$ should obtain $C$ (FALSE)
What if **evaluator** knows the truth value on one input wire?

Evaluator has $B$ (knows **false**):  
⇒ should obtain $C$ (**false**)

Evaluator has $B \oplus \Delta$ (knows **true**):
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows false):
$\Rightarrow$ should obtain $C$ (false)

Evaluator has $B \oplus \Delta$ (knows true):
$\Rightarrow$ should be able to transfer truth value from “a” wire to “c” wire
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows FALSE):
⇒ should obtain $C$ (FALSE)

Evaluator has $B \oplus \Delta$ (knows TRUE):
⇒ should be able to transfer truth value from “a” wire to “c” wire
▶ Suffices to learn $A \oplus C$
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows FALSE):

⇒ should obtain $C$ (FALSE)

Evaluator has $B \oplus \Delta$ (knows TRUE):

⇒ should be able to transfer truth value from “a” wire to “c” wire

▶ Suffices to learn $A \oplus C$
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows FALSE):
⇒ should obtain $C$ (FALSE)

Evaluator has $B \oplus \Delta$ (knows TRUE):
⇒ should be able to transfer truth value from “a” wire to “c” wire
▶ Suffices to learn $A \oplus C$
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows \textsc{false}):

\Rightarrow \text{ should obtain } C (\textsc{false})

Evaluator has $B \oplus \Delta$ (knows \textsc{true}):

\Rightarrow \text{ should be able to \textit{transfer} truth value from “a” wire to “c” wire}

\hspace{1em} ▶ Suffices to learn $A \oplus C$
What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows $\text{FALSE}$):
⇒ should obtain $C$ ($\text{FALSE}$)

Evaluator has $B \oplus \Delta$ (knows $\text{TRUE}$):
⇒ should be able to transfer truth value from “a” wire to “c” wire
  ▶ Suffices to learn $A \oplus C$
**What if evaluator knows the truth value on one input wire?**

Evaluator has $B$ (knows **false**):

$\implies$ should obtain $C$ (**false**)

Evaluator has $B \oplus \Delta$ (knows **true**):

$\implies$ should be able to **transfer** truth value from “$a$” wire to “$c$” wire

- Suffices to learn $A \oplus C$
**Half Gates** [ZahurRosulekEvans15]

*What if evaluator knows the truth value on one input wire?*

Evaluator has $B$ (knows **false**):

⇒ should obtain $C$ (**false**)

Evaluator has $B \oplus \Delta$ (knows **true**):

⇒ should be able to *transfer* truth value from “a” wire to “c” wire

- Suffices to learn $A \oplus C$
Half Gates [ZahurRosulekEvans15]

What if evaluator knows the truth value on one input wire?

Evaluator has $B$ (knows FALSE):
⇒ should obtain $C$ (FALSE)

Evaluator has $B \oplus \Delta$ (knows TRUE):
⇒ should be able to transfer truth value from “$a$” wire to “$c$” wire
▶ Suffices to learn $A \oplus C$
**What if evaluator knows the truth value on one input wire?**

Evaluator has $B$ (knows false):

$\implies$ should obtain $C$ (false)

Evaluator has $B \oplus \Delta$ (knows true):

$\implies$ should be able to transfer truth value from “a” wire to “c” wire

- Suffices to learn $A \oplus C$

Fine print: no need for permute-and-point here
Two halves make a whole!

\[ a \land b \]
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]

- Garbler chooses random bit \( r \)
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]
\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

- Garbler chooses random bit \( r \)
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]
\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

- Garbler chooses random bit \( r \)
- Arrange for evaluator to learn \( a \oplus r \) in the clear
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]

\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

one input known to evaluator

▶ Garbler chooses random bit \( r \)

▶ Arrange for evaluator to learn \( a \oplus r \) in the clear
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]
\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

one input known to garbler

- Garbler chooses random bit \( r \)
- Arrange for evaluator to learn \( a \oplus r \) in the clear
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]

\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

one input known to garbler

- Garbler chooses random bit \( r \)
- Arrange for evaluator to learn \( a \oplus r \) in the clear
- Total cost = 2 “half gates” + 1 XOR gate = 2 ciphertexts
Two halves make a whole!

\[ a \land b = (a \oplus r \oplus r) \land b \]
\[ = [(a \oplus r) \land b] \oplus [r \land b] \]

one input known to garbler

- Garbler chooses random bit \( r \)
  - \( r = \) color bit of \texttt{FALSE} wire label \( A \)
- Arrange for evaluator to learn \( a \oplus r \) in the clear
  - \( a \oplus r = \) color bit of wire label evaluator gets (\( A \) or \( A \oplus \Delta \))
- Total cost = 2 “half gates” + 1 XOR gate = 2 ciphertexts
## Scoreboard

<table>
<thead>
<tr>
<th></th>
<th>size ($\times \lambda$)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR</td>
<td>AND</td>
<td>XOR</td>
<td>AND</td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4</td>
<td>4/8</td>
<td>1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3</td>
<td>4/8</td>
<td>1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>circ. hash</td>
</tr>
<tr>
<td>GRR2</td>
<td>2</td>
<td>4/8</td>
<td>1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>FleXOR</td>
<td>{0, 1, 2}</td>
<td>{0, 1, 2}</td>
<td>{0, 1, 2}</td>
<td>circ. symm</td>
</tr>
<tr>
<td>HalfGates</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>circ. hash</td>
</tr>
<tr>
<td></td>
<td>size ($\times \lambda$)</td>
<td>garble cost</td>
<td>eval cost</td>
<td>assumption</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>XOR AND</td>
<td>XOR AND</td>
<td>XOR AND</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4 4</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3 3</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0 3</td>
<td>0 4</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
<tr>
<td>GRR2</td>
<td>2 2</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>FleXOR</td>
<td>{0, 1, 2} 2</td>
<td>{0, 1, 2} 4</td>
<td>{0, 1, 2} 1</td>
<td>circ. symm</td>
</tr>
<tr>
<td>HalfGates</td>
<td>0 2</td>
<td>0 4</td>
<td>0 2</td>
<td>circ. hash</td>
</tr>
<tr>
<td>[XYZ26]?</td>
<td>0 &lt; 2?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Every practical garbling scheme is combination of:

- Calls to symmetric primitive (can be modeled as random oracle)
- $GF(2^\lambda)$-linear operations (XOR, polynomial interpolation)
Every practical garbling scheme is combination of:

- Calls to symmetric primitive (can be modeled as random oracle)
- \textit{GF}(2^\lambda)\textit{-linear operations (XOR, polynomial interpolation)}

\textbf{Theorem ([ZahurRosulekEvans15])}

Garbling a single AND gate requires 2 ciphertexts (2\lambda bits), if garbling scheme is “linear” in this sense.
Every practical garbling scheme is combination of:

- Calls to symmetric primitive (can be modeled as random oracle)
- \( GF(2^\lambda) \)-linear operations (\( \text{xor} \), polynomial interpolation)

**Theorem ([ZahurRosulekEvans15])**

Garbling a single \text{AND} gate requires 2 ciphertexts (\( 2\lambda \) bits), if garbling scheme is “linear” in this sense.

Half-gates construction is \textit{size-optimal} among schemes that:

- ... use “known techniques”
- ... work gate-by-gate in \{\text{xor, AND, NOT}\} basis
Ways forward?

1: Consider larger “chunks” of circuit, beyond {XOR, AND, NOT} basis?
Ways forward?

1: Consider larger “chunks” of circuit, beyond \{\text{XOR, AND, NOT}\} basis?

2: Discover some clever non-linear approach to garbling?
Ways forward?

1: Consider larger “chunks” of circuit, beyond \{\textsc{xor, and, not}\} basis?

2: Discover some clever non-linear approach to garbling?

3: Wait for break-even point for asymptotically superior methods?
Ways forward?

1: Consider larger “chunks” of circuit, beyond \{\text{XOR, AND, NOT}\} basis?

2: Discover some clever non-linear approach to garbling?

3: Wait for break-even point for asymptotically superior methods?

4: Use weaker security when situation calls for it.
ZK via garbled circuits

“$\exists w : R(x, w) = 1$ ”
ZK via garbled circuits [JawurekKerschbaumOrlandi13]

“\( \exists w : R(x, w) = 1 \)”

garbled \( R(x, \cdot) \)

\( x, w \)

\( x \)
ZK via garbled circuits [JawurekKerschbaumOrlandi13]

“∃w : R(x, w) = 1”

garbled R(x, ·)

x, w

w

garbled w

OT

input

wire labels

prover knows valid wire label

⇒

open garbled circuit

correct GC

⇒

garbled output leaks nothing

open garbled output

Prover knows entire input to garbled circuit!
ZK via garbled circuits

\[ \exists w : R(x, w) = 1 \]

- garbled \( R(x, \cdot) \)
- wire labels
- OT
- input
- commit(garbled output)
- \( x \)
- \( x, w \)

"Prover knows entire input to garbled circuit!"
ZK via garbled circuits \cite{JawurekKerschbaumOrlandi13}

\[ \exists w : R(x, w) = 1 \]

\( x, w \)

```
garbled \( R(x, \cdot) \)
```

```
w
```

```
garbled w
```

```
input wire labels
```

```
OT
```

```
commit(garbled output)
```

```
contains TRUE wire label \( \Rightarrow \) prover knows valid \( w \)
ZK via garbled circuits

"∃w : R(x, w) = 1"

garbled R(x, ·)

x, w

garbled w

OT

input wire labels

commit(garbled output)

open garbled circuit

contains TRUE wire label
⇒ prover knows valid w

x
ZK via garbled circuits [JawurekKerschbaumOrlandi13]

\[ \exists w : R(x, w) = 1 \]

garbled \( R(x, \cdot) \)

\( w \rightarrow \) garbled \( w \)

input

\( x \rightarrow \) wire labels

commit(garbled output)

contains \textsc{true} wire label \( \Rightarrow \) prover knows valid \( w \)

open garbled circuit

correct GC \( \Rightarrow \) garbled output leaks nothing about \( w \)
ZK via garbled circuits \cite{JawurekKerschbaumOrlandi13}

\[ \exists w : R(x, w) = 1 \]

- garbled \( R(x, \cdot) \)
- \( w \) input wire labels
- \( x \)
- commit(garbled output)
- open garbled circuit
- open garbled output
- contains TRUE wire label \( \Rightarrow \) prover knows valid \( w \)

correct GC \( \Rightarrow \) garbled output leaks nothing about \( w \)
ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"∃w : R(x, w) = 1"

garbled R(x, ·)

x, w

w

OT

input

wire labels

garbled w

commit(garbled output)

open garbled circuit

open garbled output

contains true wire label ⇒ prover knows valid w

correct GC ⇒ garbled output leaks nothing about w

Prover knows entire input to garbled circuit!
Privacy-free garbling

For this ZK protocol, garbled circuit does not require **privacy** property

- Only **authenticity** is needed
- Garbled circuits can be significantly smaller in this case
Privacy-free garbling [FrederiksenNielsenOrlandi15]

For this ZK protocol, garbled circuit does not require **privacy** property

- Only **authenticity** is needed
- Garbled circuits can be significantly smaller in this case

<table>
<thead>
<tr>
<th></th>
<th>size ($\times \lambda$)</th>
<th>garble cost</th>
<th>eval cost</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR AND</td>
<td></td>
<td>XOR AND</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>large?</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P</td>
<td>4/8</td>
<td>4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3</td>
<td>3/3</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>Free XOR</td>
<td>0/3</td>
<td>0 4</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
<tr>
<td>GRR2</td>
<td>2/2</td>
<td>4/8 4/8</td>
<td>1/2 1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>FleXOR</td>
<td>{0, 1, 2}</td>
<td>{0, 1, 2}</td>
<td>{0, 1, 2} 1</td>
<td>circ. hash</td>
</tr>
<tr>
<td>HalfGates</td>
<td>0/2</td>
<td>0 4</td>
<td>0 2</td>
<td>circ. hash</td>
</tr>
<tr>
<td>PrivFree *</td>
<td>0/1</td>
<td>0 2</td>
<td>0 1</td>
<td>circ. hash</td>
</tr>
</tbody>
</table>
A success story!

- Reduction in size by 10x
- Reduction in computation by 10000x
the end!