Towards Optimal Garbled Circuit Constructions

Mike Rosulek

Samee Zahur, Mike Rosulek, David Evans: Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates. Eurocrypt 2015, ia.cr/2014/756
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
Garbless 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** $≡$ all encrypted gates
- **Garbled encoding** $≡$ one label per 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
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
**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
Applications

secure computation
zero-knowledge proofs
private function evaluation
randomized encodings
secure outsourcing
one-time programs
key-dependent security for encryption
How small can garbled circuits get?
### Garbled circuit size

<table>
<thead>
<tr>
<th>Method</th>
<th>References</th>
<th>Size per gate ($\times \lambda$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>[Yao86, GMW87]</td>
<td>?</td>
</tr>
<tr>
<td>Point/permute</td>
<td>[BeaverMicaliRogaway90]</td>
<td>4</td>
</tr>
</tbody>
</table>

- $\times \lambda$ indicates multiplication by security parameter $\lambda$.
## Garbled circuit size

<table>
<thead>
<tr>
<th></th>
<th>size per gate ($\times \lambda$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR</td>
</tr>
<tr>
<td>Classical</td>
<td>?</td>
</tr>
<tr>
<td>Point/permute</td>
<td>4</td>
</tr>
<tr>
<td>GRR3</td>
<td>3</td>
</tr>
</tbody>
</table>

*References:*
- [Yao86,GMW87]
- [BeaverMicaliRogaway90]
- [NaorPinkasSumner99]
- [PinkasSchneiderSmartWilliams09]
- [KolesnikovSchneider08]
- [KolesnikovMohasselRosulek14]
- [ZahurRosulekEvans15]
<table>
<thead>
<tr>
<th></th>
<th>size per gate (×λ)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XOR</td>
<td>AND</td>
<td></td>
</tr>
<tr>
<td>Classical [Yao86,GMW87]</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Point/permute [BeaverMicaliRogaway90]</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GRR3 [NaorPinkasSumner99]</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Free XOR [KolesnikovSchneider08]</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
# Garbled circuit size

<table>
<thead>
<tr>
<th>Type</th>
<th>Reference</th>
<th>XOR</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>[Yao86,GMW87]</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Point/permute</td>
<td>[BeaverMicaliRogaway90]</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GRR3</td>
<td>[NaorPinkasSumner99]</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Free XOR</td>
<td>[KolesnikovSchneider08]</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>GRR2</td>
<td>[PinkasSchneiderSmartWilliams09]</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
# Garbled circuit size

<table>
<thead>
<tr>
<th>Method</th>
<th>Sources</th>
<th>XOR per gate</th>
<th>AND per gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>[Yao86, GMW87]</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Point/permute</td>
<td>[Beaver, Micali, Rogaway90]</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GRR3</td>
<td>[Naor, Pinkas, Sumner99]</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Free XOR</td>
<td>[Kolesnikov, Schneider08]</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>GRR2</td>
<td>[Pinkas, Schneider, Smart, Williams09]</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>FleXOR</td>
<td>[Kolesnikov, Mohassel, Rosulek14]</td>
<td>{0, 1, 2}</td>
<td>2</td>
</tr>
<tr>
<td>HalfGates</td>
<td>[Zahur, Rosulek, Evans15]</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Is there a lower bound?
Lower bounds for garbled circuits

**Pitfalls:**

Want to resolve *optimal constants*

- e.g., $3\lambda$ vs $2\lambda$ per AND gate
Lower bounds for garbled circuits

**Pitfalls:**

Want to resolve *optimal constants*

- e.g., $3\lambda$ vs $2\lambda$ per AND gate

Asymptotically superior (but impractical) constructions exist

- e.g., size $|C| + O(\text{poly}(\lambda))$ [BonehGGHNSVV14] vs $O(\lambda |C|)$ [Yao]
Pitfalls:

Want to resolve **optimal constants**

- e.g., $3\lambda$ vs $2\lambda$ per AND gate

Asymptotically superior (but impractical) constructions exist

- e.g., size $|C| + O(\text{poly}(\lambda))$ \cite{BonehGGHNSVV14} vs $O(\lambda|C|)$ \cite{Yao}

---

Can we prove a lower bound in a **restricted model** that captures “known, **practical** techniques?”
Lower bounds for garbled circuits

**Pitfalls:**

Want to resolve **optimal constants**

- e.g., $3\lambda$ vs $2\lambda$ per AND gate

Asymptotically superior (but impractical) constructions exist

- e.g., size $|C| + O(poly(\lambda))$ [BonehGGHNSVV14] vs $O(\lambda|C|)$ [Yao]

**Can we prove a lower bound in a restricted model that captures “known, practical techniques?”**

**Theorem ([ZahurEvansRosulek15])**

A **linear gate garbling scheme** requires $2\lambda$ bits to garble a single AND gate. Within this model, “half-gates” construction is optimal.
Rest of talk

1: Restricted model: What is a “linear gate garbling scheme?”

2: The lower bound for AND gates (sketch)

3: Looking forward
“Known, practical techniques”

Symmetric-key cryptography only

- Formalize via computationally unbounded adversaries + random oracle [ImpagliazzoRudich88]
- e.g.: encrypt garbled truth table as $E_{A,B}(C) = H(A||B) \oplus C$

One caveat: state of art schemes are all non-linear in one specific way...
“Known, practical techniques”

**Symmetric-key cryptography only**
- Formalize via computationally unbounded adversaries + random oracle [ImpagliazzoRudich88]
- e.g.: encrypt garbled truth table as $E_{A,B}(C) = H(A||B) \oplus C$

**Linear operations exclusively**
- Wire labels, garbled gates, RO outputs are field/ring elements (e.g.: $GF(2^\lambda)$)
- Garbling/evaluation = linear operations + calls to RO.
- e.g.: xor, polynomial interpolation
“Known, practical techniques”

Symmetric-key cryptography only
- Formalize via computationally unbounded adversaries + random oracle [ImpagliazzoRudich88]
- e.g.: encrypt garbled truth table as $E_{A,B}(C) = H(A\|B) \oplus C$

Linear operations exclusively
- Wire labels, garbled gates, RO outputs are field/ring elements (e.g.: $GF(2^\lambda)$)
- Garbling/evaluation = linear operations + calls to RO.
- e.g.: xor, polynomial interpolation

One caveat: state of art schemes are *all non-linear* in one specific way . . .
Point/permute \cite{BeaverMicaliRogaway90}

\[ A_0, A_1 \quad C_0, C_1 \]

\[ B_0, B_1 \]

\(!\mathbb{E}_{A_0, B_0}(C_0)\]
\(!\mathbb{E}_{A_0, B_1}(C_1)\]
\(!\mathbb{E}_{A_1, B_0}(C_0)\]
\(!\mathbb{E}_{A_1, B_1}(C_0)\]

\(\mathbb{E}_{A_0, B_0}(C_0) \oplus \mathbb{E}_{A_0, B_1}(C_1) \oplus \mathbb{E}_{A_1, B_0}(C_0) \oplus \mathbb{E}_{A_1, B_1}(C_0)\]

Randomly assign \((A_0, B_0)\) or \((A_1, B_1)\) 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

\(!\mathbb{E}_{A_0, B_0}(C_0)\]
\(!\mathbb{E}_{A_0, B_1}(C_1)\]
\(!\mathbb{E}_{A_1, B_0}(C_0)\]
\(!\mathbb{E}_{A_1, B_1}(C_0)\]

\(\mathbb{E}_{A_0, B_0}(C_0) \oplus \mathbb{E}_{A_0, B_1}(C_1) \oplus \mathbb{E}_{A_1, B_0}(C_0) \oplus \mathbb{E}_{A_1, B_1}(C_0)\]

Non-linear: evaluator's choice of linear operation depends on color
Optimized GC schemes crucially take advantage of nonlinearity!
Randomly assign \((\star, \star)\) or \((\star, \star)\) to each pair of wire labels

Include color in the wire label (e.g., as last bit)
Point/permute [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
Point/permute \cite{BeaverMicaliRogaway90}

- Randomly assign $(\bullet, \bullet)$ or $(\circ, \bullet)$ 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
Point/permute [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
- Evaluate by decrypting ciphertext indexed by your colors
Point/permute \[\text{[BeaverMicaliRogaway90]}\]

- 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
- Evaluate by decrypting ciphertext indexed by your colors

\[E_{A_0, B_1}(C_1), E_{A_0, B_0}(C_0), E_{A_1, B_1}(C_0), E_{A_1, B_0}(C_0)\]
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

$\mathbb{E}_{A,B}(C)$ doesn’t need verifiable decryption

$\Rightarrow$ smaller garbled circuit, encryption/decryption “linear”
Point/permute [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

✓ $E_{A,B}(C)$ doesn’t need verifiable decryption
  ⇒ smaller garbled circuit, encryption/decryption “linear”

✗ Non-linear: evaluator’s choice of linear operation depends on color
✗ Optimized GC schemes crucially take advantage of nonlinearity!
Linear (gate) garbling: details

Apart from point-permute, and calls to random oracle, everything is linear.

**Important:** only the constructions, not adversary, must be linear!
Linear (gate) garbling: details

Garbler:
- samples field elements
Linear (gate) garbling: details

Garbler:
- samples field elements
- calls random oracle
Linear (gate) garbling: details

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
Linear (gate) garbling: details

\[ RO \text{ resp.} \langle \begin{array}{c}
A^* = \text{true} \\
A^* = \text{false} \\
B^* = \text{true} \\
B^* = \text{false}
\end{array} \rangle \times RO \text{ resp.} \]

\[
\begin{array}{c}
\text{gb gate} \\
C^\text{false} \\
C^\text{true}
\end{array}
\]

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Linear (gate) garbling: details

**Evaluator:**
- knows subset of garbler's values (and knows colors)
- applies linear combination
- result is output label

**Garbler:**
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Linear (gate) garbling: details

Evaluator:
- knows subset of garbler’s values (and knows colors)

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Linear (gate) garbling: details

$C_f(a,b) = T \times \begin{cases} \text{RO} & \text{resp.} \left\langle A = \text{true}, A = \text{false} \right\rangle \\ \text{gb gate} \end{cases} \times T^{-1} \in \text{kernel of } f(a,b) = \begin{cases} 0 & \text{depends only on colors} \\ \text{fixed, wlog depends only on "color mapping"} \end{cases}$

**Evaluator:**
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

**Garbler:**
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
**Linear (gate) garbling: details**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>RO resp.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>A*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>A*</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>B*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>B*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>gb gate</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>C\text{false}</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>C\text{true}</td>
</tr>
</tbody>
</table>

**Evaluator:**
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

**Garbler:**
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Linear (gate) garbling: details

$F$ \text{resp.} \langle A = \text{true} \rangle \times \langle A = \text{false} \rangle \times \langle B = \text{false} \rangle \times \langle B = \text{true} \rangle \times \langle C = \text{false} \rangle \times \langle C = \text{true} \rangle$

Evaluator:
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Linear (gate) garbling: details

Evaluator:
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination

$\begin{bmatrix} 0 & T \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \end{bmatrix} \in \text{kernel of}$
# Linear (gate) garbling: details

## Evaluator:
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

## Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination

<table>
<thead>
<tr>
<th>gate</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$\in$ kernel of

depends only on output

fixed, wlog

depends only on “color mapping”
Linear (gate) garbling: details

$$ T = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \in \text{kernel of} \quad \begin{bmatrix} \text{fixed, wlog} \\ \text{depends only on} \\ \text{“color mapping”} \end{bmatrix} \{ \text{garbled gate} \} $$

Evaluator:
- knows subset of garbler’s values (and knows colors)
- applies linear combination
- result is output label

Garbler:
- samples field elements
- calls random oracle
- picks secret “color mapping”
- applies linear combination
Lower bound (part 1)

\[ F \times \langle A = \text{false}, A = \text{true}, B = \text{false}, B = \text{true} \rangle = 0 \]
Lower bound (part 1)

\[ \begin{align*}
T \times \begin{cases}
A^* = \text{false} \\
A' = \text{true} \\
B^* = \text{false} \\
B' = \text{true}
\end{cases}
\right) &= 0 \\
T \times \begin{cases}
A^* = \text{false} \\
A' = \text{true} \\
B^* = \text{false} \\
B' = \text{true}
\end{cases}
\right) &= 0
\end{align*} \]
Lower bound (part 1)

\[
\begin{align*}
(T \times F) & = 0 \\
(T \times F) & = 0 \\
(T \times F) & = 0 \\
(T \times F) & = 0
\end{align*}
\]
Lower bound (part 1)

\[
\begin{align*}
&\left(\begin{array}{c}
T \\
F
\end{array}\right) \\
&\times \\
&\left(\begin{array}{c}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right) \\
&= 0
\end{align*}
\]

\[
\begin{align*}
&\left(\begin{array}{c}
T \\
F
\end{array}\right) \\
&\times \\
&\left(\begin{array}{c}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{array}\right) \\
&= 0
\end{align*}
\]

\[
\begin{align*}
&\left(\begin{array}{c}
T \\
F
\end{array}\right) \\
&\times \\
&\left(\begin{array}{c}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{array}\right) \\
&= 0
\end{align*}
\]

\[
\begin{align*}
&\left(\begin{array}{c}
T \\
F
\end{array}\right) \\
&\times \\
&\left(\begin{array}{c}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right) \\
&= 0
\end{align*}
\]
Lower bound (part 1)

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right.
\end{array}
= 0
\end{array}
\]

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right.
\end{array}
= 0
\end{array}
\]

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{array}\right.
\end{array}
= 0
\end{array}
\]

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{array}\right.
\end{array}
= 0
\end{array}
\]

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right.
\end{array}
= 0
\end{array}
\]

\[
\begin{array}{c}
\left(\begin{array}{c}
T \\
F
\end{array}\right) \times
\begin{array}{c}
\left\langle \begin{array}{l}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{array}\right.
\end{array}
= 0
\end{array}
\]
Lower bound (part 1)

\[
\begin{pmatrix}
T \\
F
\end{pmatrix}
\times
\begin{pmatrix}
A = \text{false} \\
A = \text{true} \\
B = \text{false} \\
B = \text{true}
\end{pmatrix}
= 0
\]

\[
\begin{pmatrix}
T \\
F
\end{pmatrix}
\times
\begin{pmatrix}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{pmatrix}
= 0
\]

\[
\begin{pmatrix}
T \\
F
\end{pmatrix}
\times
\begin{pmatrix}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{pmatrix}
= 0
\]

\[
\begin{pmatrix}
T \\
F
\end{pmatrix}
\times
\begin{pmatrix}
A = \text{false} \\
A = \text{true} \\
B = \text{true} \\
B = \text{false}
\end{pmatrix}
= 0
\]
Lower bound (part 1)

\[
\begin{pmatrix}
T & \times & \begin{pmatrix}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{false} \\
B^* = \text{true}
\end{pmatrix} \\
\text{F} & \times & \begin{pmatrix}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{true} \\
B^* = \text{false}
\end{pmatrix}
\end{pmatrix} = 0
\]

\[
\begin{pmatrix}
T & \times & \begin{pmatrix}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{false} \\
B^* = \text{true}
\end{pmatrix} \\
\text{F} & \times & \begin{pmatrix}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{true} \\
B^* = \text{false}
\end{pmatrix}
\end{pmatrix} = 0
\]
Lower bound (part 1)

\[
\begin{pmatrix}
T \\
\end{pmatrix}
\times
\begin{cases}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{false} \\
B^* = \text{true}
\end{cases}
\] = 0

\begin{pmatrix}
T \\
\end{pmatrix}
\times
\begin{cases}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{true} \\
B^* = \text{false}
\end{cases}
\] = 0

\begin{pmatrix}
T \\
\end{pmatrix}
\times
\begin{cases}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{true} \\
B^* = \text{false}
\end{cases}
\] = 0

\begin{pmatrix}
T \\
\end{pmatrix}
\times
\begin{cases}
A^* = \text{false} \\
A^* = \text{true} \\
B^* = \text{false} \\
B^* = \text{true}
\end{cases}
\] = 0

\begin{pmatrix}
T \\
\end{pmatrix}
\times
\begin{cases}
A^* = \text{true} \\
A^* = \text{false} \\
B^* = \text{false} \\
B^* = \text{true}
\end{cases}
\] = 0
Lower bound (part 2)

\[(v - v')(M - M') = 0\]
Lower bound (part 2)

\[(v - v')(M - M') = 0\]

Rest of proof:

- \(v, v'\) distinct (otherwise privacy can be violated)
- \(\text{dim}(\ker(M - M')) \geq 1\)
Lower bound (part 2)

\[(v - v')(M - M') = 0\]

Rest of proof:

- \(v, v'\) distinct (otherwise privacy can be violated)
  \[\Rightarrow \text{dim}(\ker(M - M')) \geq 1\]

- \(M, M'\) distinct (otherwise correctness is violated)
  \[\Rightarrow \text{dim}(\text{rowspace}(M - M')) \geq 1\]
Lower bound (part 2)

\[(v - v')(M - M') = 0\]

Rest of proof:
- \(v, v'\) distinct (otherwise privacy can be violated)
  \(\Rightarrow\) \(\dim(\ker(M - M')) \geq 1\)

- \(M, M'\) distinct (otherwise correctness is violated)
  \(\Rightarrow\) \(\dim(\text{rowspace}(M - M')) \geq 1\)

\(\Rightarrow\) \(M, M'\) have at least 2 rows
\(\Rightarrow\) garbled gate has at least \(2\lambda\) bits
Just the beginning...

**Theorem**

A linear garbling scheme requires $2\lambda$ bits to garble a single AND gate.
Theorem

A linear garbling scheme requires $2\lambda$ bits to garble a single AND gate.

Purpose of lower bounds in a restricted model:

- Give up, you can’t do any better
Just the beginning...

**Theorem**

A linear garbling scheme requires $2\lambda$ bits to garble a single AND gate.

Purpose of lower bounds in a restricted model:
- Give up, you can’t do any better
- Try to do better by avoiding assumptions of the restricted model
Limitations of model

**Limitation:**
point-and-permute is sole source of non-linearity

**Opportunity:**
smaller GC using alternatives to point-permute [BallMalkinRosulek16]
Limitations of model

**Limitation:**
point-and-permute is sole source of non-linearity

**Opportunity:**
smaller GC using alternatives to point-permute [BallMalkinRosulek16]

optimal for gate-by-gate garbling, in \{AND, XOR, NOT\} basis

smaller GC by garbling larger “chunks” of a circuit at a time [MalkinPastroShelat16]
Limitations of model

Limitation: point-and-permute is sole source of non-linearity

Opportunity: smaller GC using alternatives to point-permute [BallMalkinRosulek16]

optimal for gate-by-gate garbling, in \{AND, XOR, NOT\} basis

smaller GC by garbling larger “chunks” of a circuit at a time [MalkinPastroShelat16]

model doesn’t allow nested calls to random oracle

maybe nesting leads to smaller GC? e.g.: \(H(H(A) \oplus B)\)
# Limitations of model

<table>
<thead>
<tr>
<th>Limitation:</th>
<th>Opportunity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>point-and-permute is sole source of non-linearity</td>
<td>smaller GC using alternatives to point-permute [BallMalkinRosulek16]</td>
</tr>
<tr>
<td>optimal for gate-by-gate garbling, in {\texttt{AND}, \texttt{XOR}, \texttt{NOT}} basis</td>
<td>smaller GC by garbling larger “chunks” of a circuit at a time [MalkinPastroShelat16]</td>
</tr>
<tr>
<td>model doesn’t allow nested calls to random oracle</td>
<td>maybe nesting leads to smaller GC? e.g.: $H(H(A) \oplus B)$</td>
</tr>
<tr>
<td>linear!</td>
<td>maybe non-linear (but still practical) techniques lead to smaller GC? e.g., compute $GF(2^λ)$-inverse of a wire label</td>
</tr>
</tbody>
</table>
Beyond garbled circuits

General model of “practical symmetric-key crypto”? 
- Random oracle + linear operations
- Can prove fine-grained lower bounds about concrete constants
Beyond garbled circuits

General model of “practical symmetric-key crypto”?  
- Random oracle + linear operations  
- Can prove fine-grained lower bounds about concrete constants  

Similar models have been fruitful for lower bounds / impossibility results.

Generic group model [Shoup97]:  
- Algorithm only uses prescribed group operations  
- Generic (“structure preserving”) signature schemes require 3 group elements, etc. [AbeGHO11,AbeGOT14]

[Black-box] Arithmetic cryptography  
[IshaiPrabhakaranSahai09,ApplebaumAvronBrzuska15]  
- Algorithm uses arbitrary field as black-box  
- Asymptotic lower bounds & impossibility results
the end!