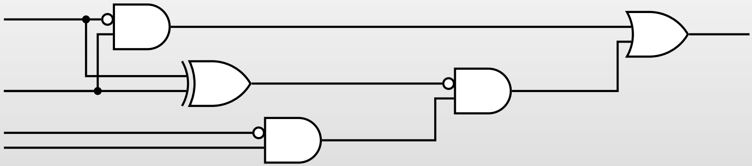
Towards Optimal Garbled Circuit Constructions

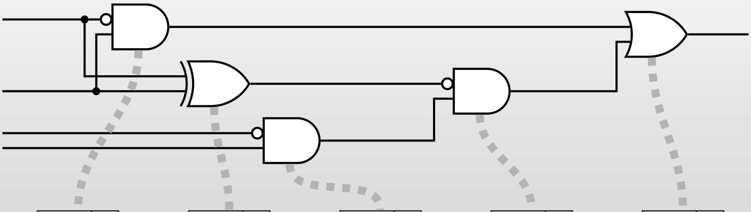
Mike Rosulek
Oregon State University OSU

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]



Garbled circuit framework [Yao86]



0	0	0
0	1	1
1	0	0
1	1	0

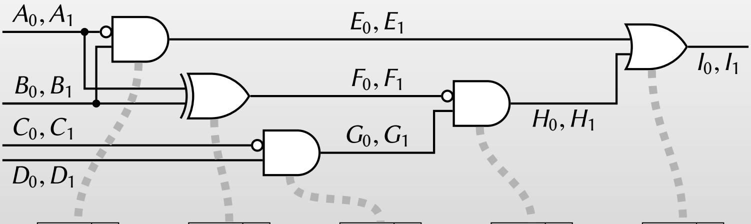
0	0	0
0	1	1
1	0	1
1	1	0

0	0	0
0	1	1
1	0	0
1	1	0

0	0	0
0	1	1
1	0	0
1	1	0

0	0	0
0	1	1
1	0	1
1	1	1

Garbled circuit framework [Yao86]



0	0	0
0	1	1
1	0	0
1	1	0

0	0	0
0	1	1
1	0	1
1	1	0

0	0	0
0	1	1
1	0	0
1	1	0

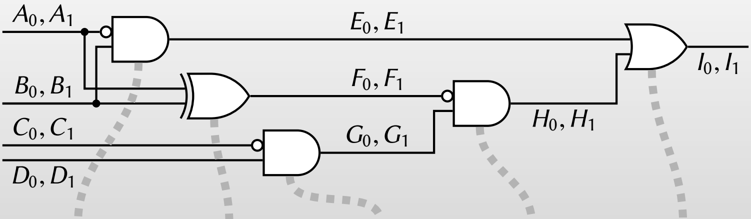
0	0	0
0	1	1
1	0	0
1	1	0

0	0	0
0	1	1
1	0	1
1	1	1

Garbling a circuit:

- Pick random **labels** W_0, W_1 on each wire

Garbled circuit framework [Yao86]



A_0	B_0	E_0
A_0	B_1	E_1
A_1	B_0	E_0
A_1	B_1	E_0

A_0	B_0	F_0
A_0	B_1	F_1
A_1	B_0	F_1
A_1	B_1	F_0

C_0	D_0	G_0
C_0	D_1	G_1
C_1	D_0	G_0
C_1	D_1	G_0

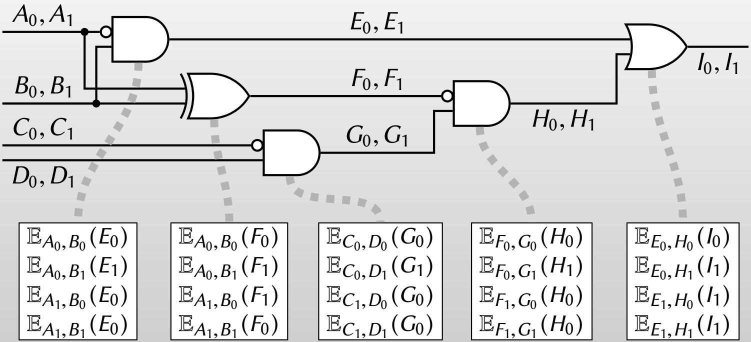
F_0	G_0	H_0
F_0	G_1	H_1
F_1	G_0	H_0
F_1	G_1	H_0

E_0	H_0	I_0
E_0	H_1	I_1
E_1	H_0	I_1
E_1	H_1	I_1

Garbling a circuit:

- Pick random **labels** W_0, W_1 on each wire

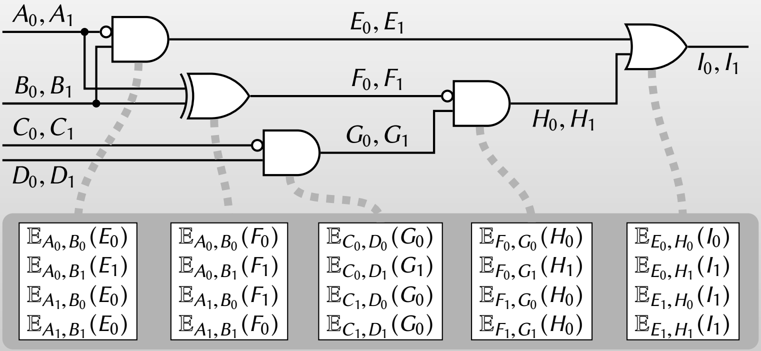
Garbled circuit framework [Yao86]



Garbling a circuit:

- Pick random **labels** W_0, W_1 on each wire
- “Encrypt” truth table of each gate

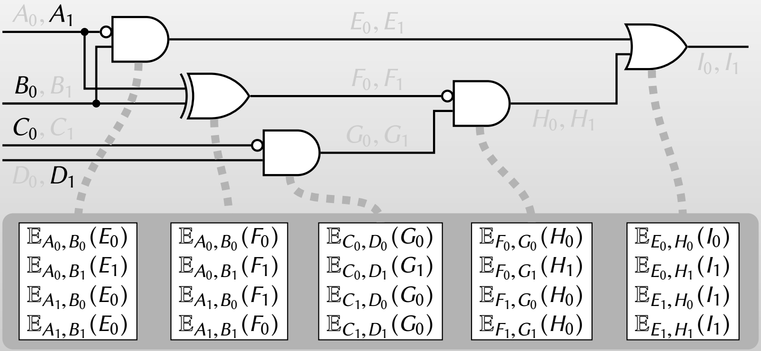
Garbled circuit framework [Yao86]



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

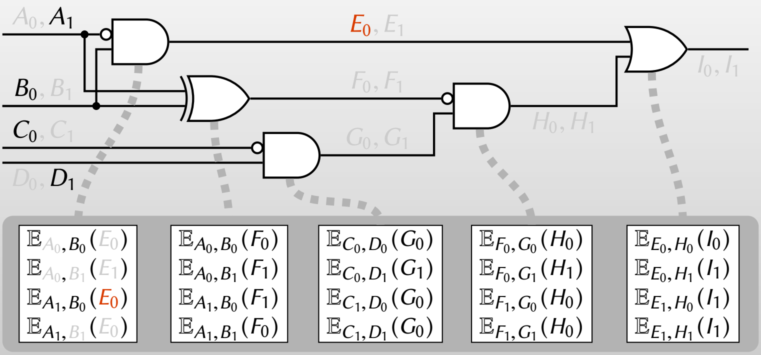
Garbled circuit framework [Yao86]



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

Garbled circuit framework [Yao86]



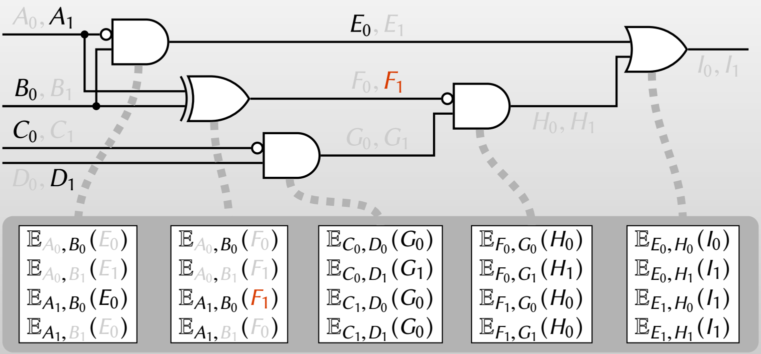
Garbling a circuit:

- 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

Garbled evaluation:

- Only one ciphertext per gate is decryptable

Garbled circuit framework [Yao86]



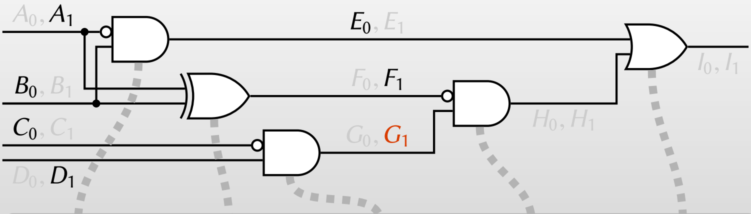
Garbling a circuit:

- Pick random **labels** W_0, W_1 on each wire
- "Encrypt" truth table of each gate
- **Garbled circuit** ≡ all encrypted gates
- **Garbled encoding** ≡ one label per wire

Garbled evaluation:

- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire

Garbled circuit framework [Yao86]



$\mathbb{E}_{A_0,B_0}(E_0)$	$\mathbb{E}_{A_0,B_0}(F_0)$	$\mathbb{E}_{C_0,D_0}(G_0)$	$\mathbb{E}_{F_0,G_0}(H_0)$	$\mathbb{E}_{E_0,H_0}(I_0)$
$\mathbb{E}_{A_0,B_1}(E_1)$	$\mathbb{E}_{A_0,B_1}(F_1)$	$\mathbb{E}_{C_0,D_1}(G_1)$	$\mathbb{E}_{F_0,G_1}(H_1)$	$\mathbb{E}_{E_0,H_1}(I_1)$
$\mathbb{E}_{A_1,B_0}(E_0)$	$\mathbb{E}_{A_1,B_0}(F_1)$	$\mathbb{E}_{C_1,D_0}(G_0)$	$\mathbb{E}_{F_1,G_0}(H_0)$	$\mathbb{E}_{E_1,H_0}(I_1)$
$\mathbb{E}_{A_1,B_1}(E_0)$	$\mathbb{E}_{A_1,B_1}(F_0)$	$\mathbb{E}_{C_1,D_1}(G_0)$	$\mathbb{E}_{F_1,G_1}(H_0)$	$\mathbb{E}_{E_1,H_1}(I_1)$

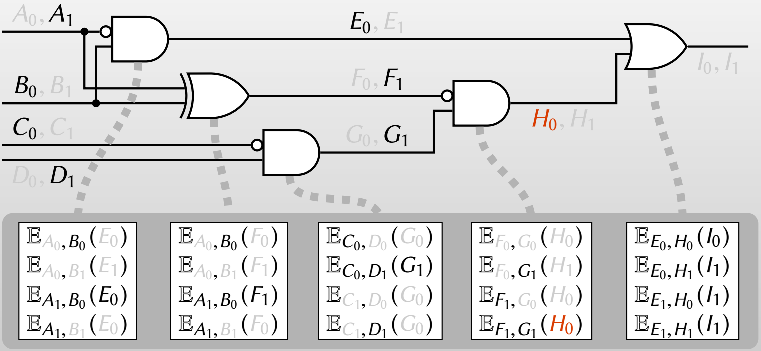
Garbling a circuit:

- 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

Garbled evaluation:

- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire

Garbled circuit framework [Yao86]



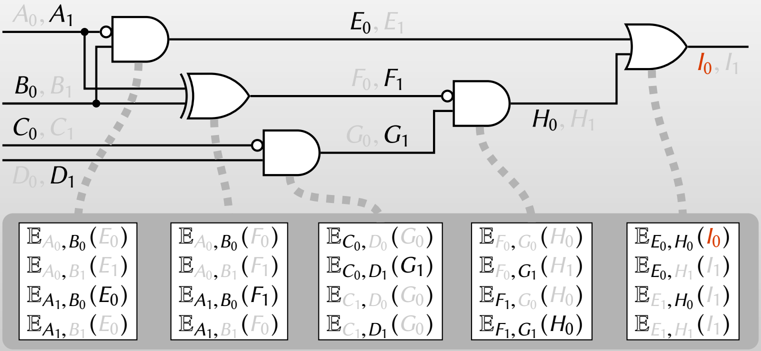
Garbling a circuit:

- Pick random **labels** W_0, W_1 on each wire
- "Encrypt" truth table of each gate
- **Garbled circuit** ≡ all encrypted gates
- **Garbled encoding** ≡ 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** ≡ all encrypted gates
- **Garbled encoding** ≡ 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

$\vdots$

How small can garbled circuits get?

Garbled circuit size

		size per gate ($\times\lambda$)	
		XOR	AND
Classical	[Yao86,GMW87]	?	?
Point/permute	[BeaverMicaliRogaway90]	4	4

Garbled circuit size

		size per gate ($\times\lambda$) XOR	AND
Classical	[Yao86,GMW87]	?	?
Point/permute	[BeaverMicaliRogaway90]	4	4
GRR3	[NaorPinkasSumner99]	3	3

Garbled circuit size

		size per gate ($\times\lambda$) XOR	AND
Classical	[Yao86,GMW87]	?	?
Point/permute	[BeaverMicaliRogaway90]	4	4
GRR3	[NaorPinkasSumner99]	3	3
Free XOR	[KolesnikovSchneider08]	0	3

Garbled circuit size

		size per gate ($\times\lambda$)	
		XOR	AND
Classical	[Yao86,GMW87]	?	?
Point/permute	[BeaverMicaliRogaway90]	4	4
GRR3	[NaorPinkasSumner99]	3	3
Free XOR	[KolesnikovSchneider08]	0	3
GRR2	[PinkasSchneiderSmartWilliams09]	2	2

Garbled circuit size

		size per gate ($\times\lambda$)	
		XOR	AND
Classical	[Yao86,GMW87]	?	?
Point/permute	[BeaverMicaliRogaway90]	4	4
GRR3	[NaorPinkasSumner99]	3	3
Free XOR	[KolesnikovSchneider08]	0	3
GRR2	[PinkasSchneiderSmartWilliams09]	2	2
FleXOR	[KolesnikovMohasselRosulek14]	$\{0, 1, 2\}$	2
HalfGates	[ZahurRosulekEvans15]	**0**	**2**

Is there a lower bound?

Lower bounds for garbled circuits

Pitfalls:

Want to resolve **optimal constants**

- e.g., 3λ vs 2λ per AND gate

Lower bounds for garbled circuits

Pitfalls:

Want to resolve **optimal constants**

- e.g., 3λ vs 2λ per AND gate

Asymptotically superior (but impractical) constructions exist

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

Lower bounds for garbled circuits

Pitfalls:

Want to resolve **optimal constants**

- e.g., 3λ vs 2λ per AND gate

Asymptotically superior (but impractical) constructions exist

- e.g., size $|C| + O(\text{poly}(\lambda))$ [BonehGGHNSVV14] vs $O(\lambda|C|)$ [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., 3λ vs 2λ per AND gate

Asymptotically superior (but impractical) constructions exist

- e.g., size $|C| + O(\text{poly}(\lambda))$ [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 2λ 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 $\mathbb{E}_{A,B}(C) = H(A \| B) \oplus C$

“Known, practical techniques”

Symmetric-key cryptography only

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

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

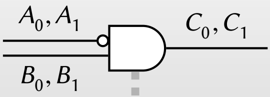
- Formalize via computationally unbounded adversaries + random oracle [ImpagliazzoRudich88]
- e.g.: encrypt garbled truth table as $\mathbb{E}_{A,B}(C) = H(A \| B) \oplus C$

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

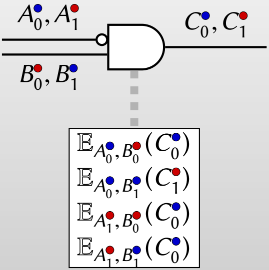
One caveat: state of art schemes are *all non-linear* in one specific way . . .

Point/permute [BeaverMicaliRogaway90]



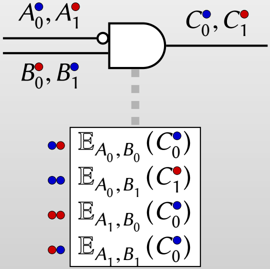
$\mathbb{E}_{A_0,B_0}(C_0)$
$\mathbb{E}_{A_0,B_1}(C_1)$
$\mathbb{E}_{A_1,B_0}(C_0)$
$\mathbb{E}_{A_1,B_1}(C_0)$

Point/permute [BeaverMicaliRogaway90]



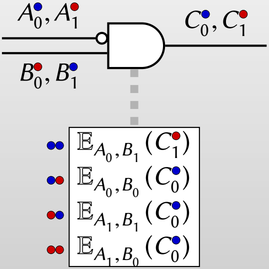
- Randomly assign (•,•) or (•,•) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)

Point/permute [BeaverMicaliRogaway90]



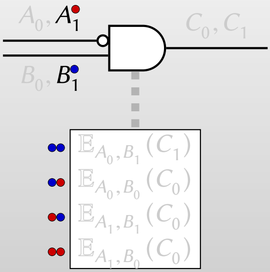
- Randomly assign (•, •) or (•, •) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys

Point/permute [BeaverMicaliRogaway90]



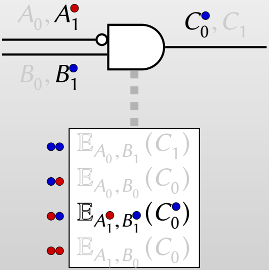
- Randomly assign (•, •) or (•, •) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys

Point/permute [BeaverMicaliRogaway90]



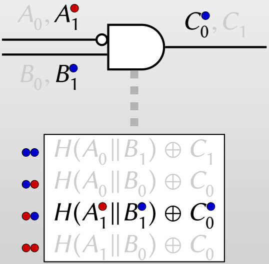
- Randomly assign (•,•) or (•,•) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys
- Evaluate by decrypting ciphertext indexed by your colors

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

Point/permute [BeaverMicaliRogaway90]

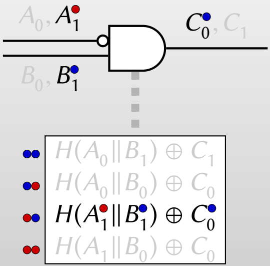


- Randomly assign (•,•) or (•,•) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys
- Evaluate by decrypting ciphertext indexed by your colors

✔ $\mathbb{E}_{A,B}(C)$ 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

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

$\Rightarrow$ 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

$\$_{\mathbb{F}}$

Garbler:

- samples field elements

Linear (gate) garbling: details

Garbler:

- samples field elements
- calls random oracle

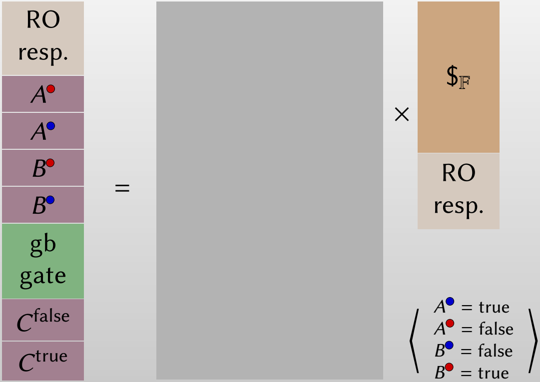
Linear (gate) garbling: details

$$\left\langle \begin{array}{l} A^{\bullet} = \text{true} \\ A^{\bullet} = \text{false} \\ B^{\bullet} = \text{false} \\ B^{\bullet} = \text{true} \end{array} \right\rangle$$

Garbler:

- samples field elements
- calls random oracle
- picks secret "color mapping"

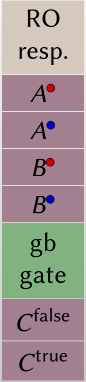
Linear (gate) garbling: details



Garbler:

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

Linear (gate) garbling: details

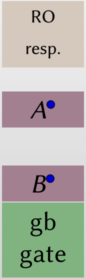


Evaluator:

Garbler:

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

Linear (gate) garbling: details



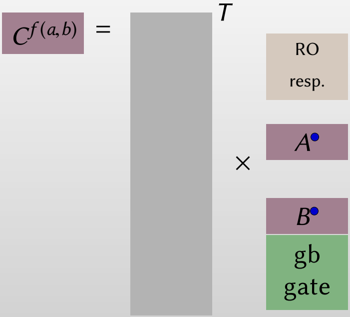
Evaluator:

- 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



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

$$0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \\ \mathbf{-1} \\ \mathbf{0} \end{bmatrix}^T \times \begin{bmatrix} \text{RO resp.} \\ A^{\color{red}\bullet} \\ A^{\color{blue}\bullet} \\ B^{\color{red}\bullet} \\ B^{\color{blue}\bullet} \\ \text{gb gate} \\ C^{\text{false}} \\ C^{\text{true}} \end{bmatrix}$$

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

Linear (gate) garbling: details

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \\ \\ \mathbf{-1} \\ \mathbf{0} \end{bmatrix}^T \in \text{kernel of}$$

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

$$\begin{array}{r}\text{depends only on colors}\left\{\begin{array}{c}0\\0\\0\\ \\ \end{array}\right.\\ \text{depends only on output}\left\{\begin{array}{c}\mathbf{-1}\\ \mathbf{0}\end{array}\right.\end{array}{}^{T} \in \text{kernel of} \begin{array}{|c|}\hline \text{fixed, wlog}\\ \hline \text{depends only on “color mapping”}\\ \hline\end{array}$$

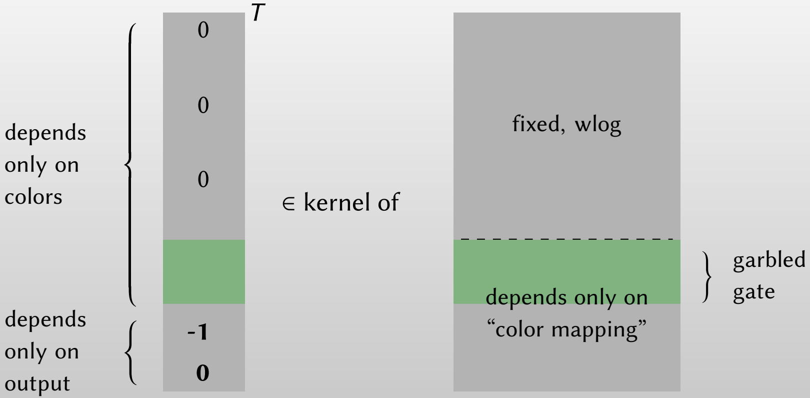
Evaluator:

- 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



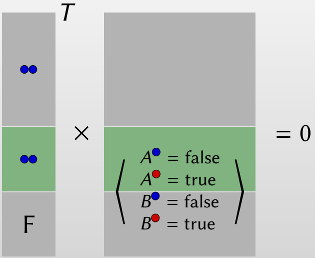
Evaluator:

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

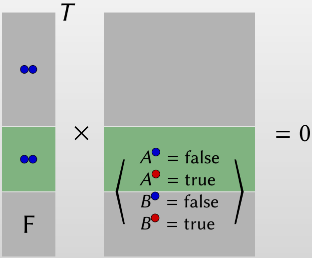
Garbler:

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

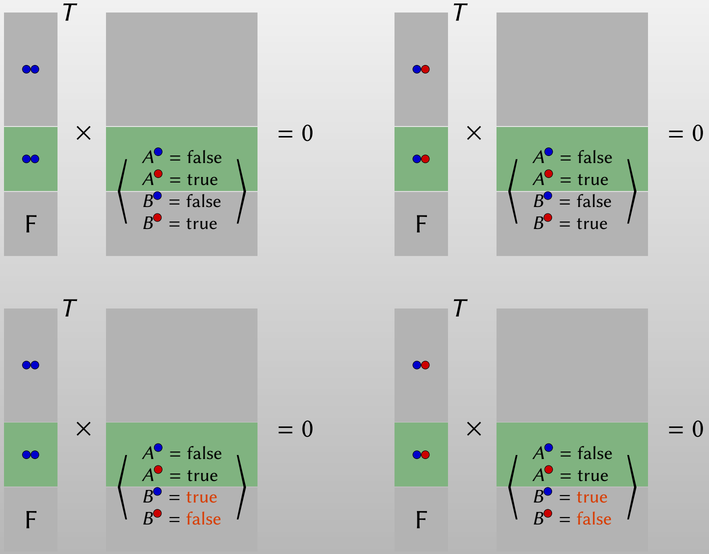
Lower bound (part 1)



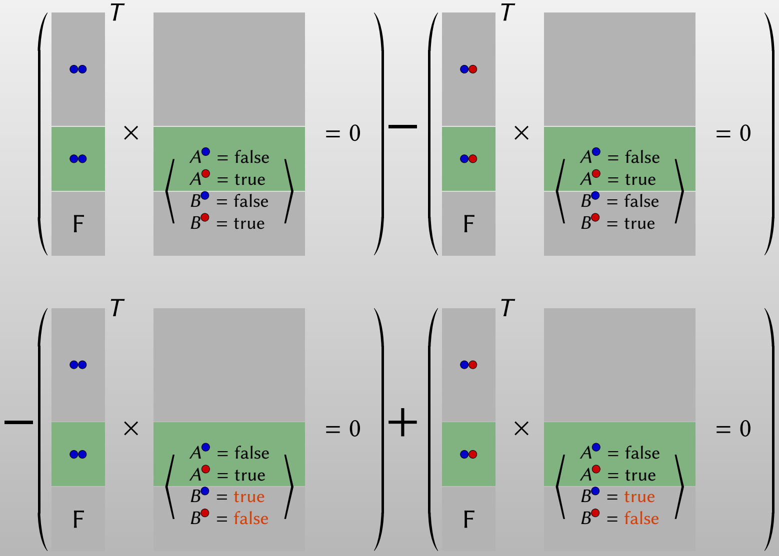
Lower bound (part 1)



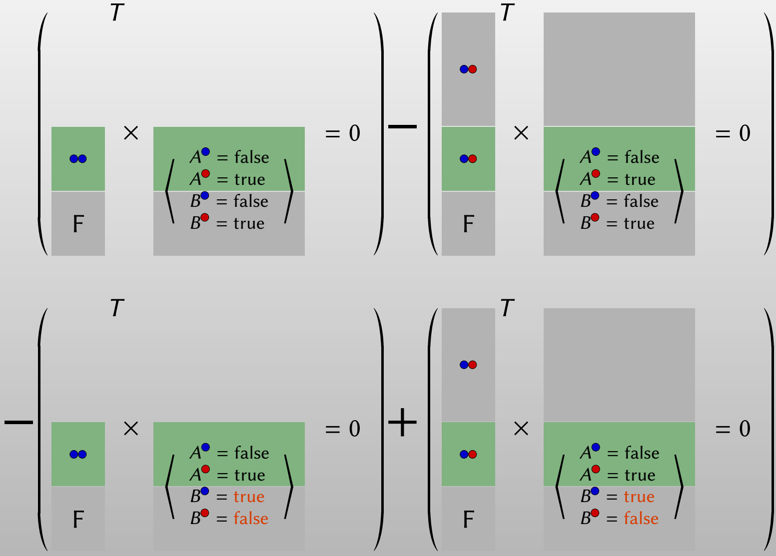
Lower bound (part 1)



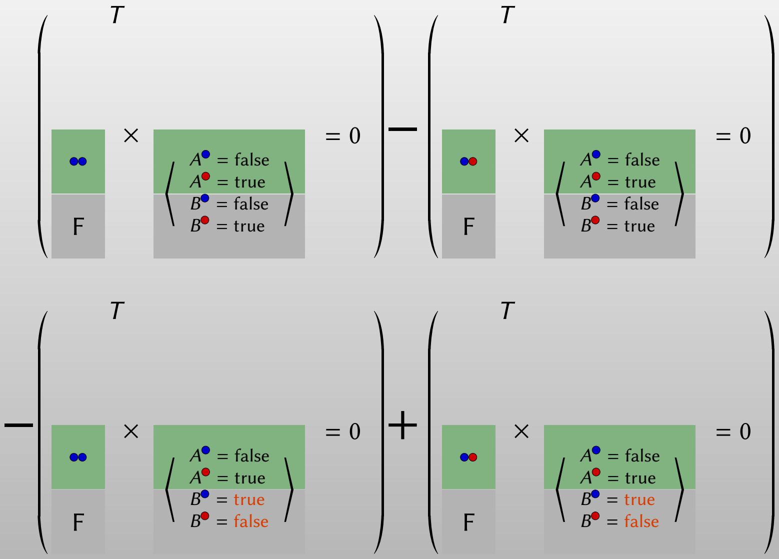
Lower bound (part 1)



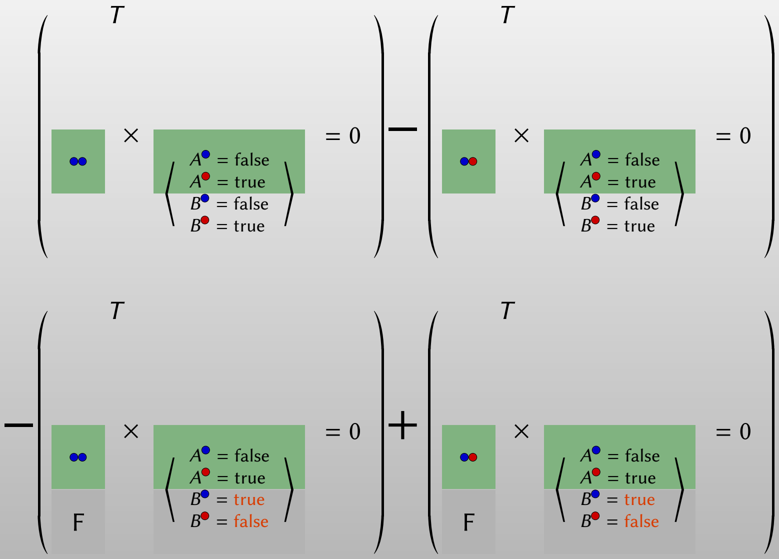
Lower bound (part 1)



Lower bound (part 1)



Lower bound (part 1)



Lower bound (part 1)

$$
\left({\color{blue}\bullet}{\color{blue}\bullet} \times \left\langle \begin{array}{l} A^{\color{blue}\bullet} = \text{false} \\ A^{\color{red}\bullet} = \text{true} \\ B^{\color{blue}\bullet} = \text{false} \\ B^{\color{red}\bullet} = \text{true} \end{array} \right\rangle = 0 \right)^{T}
-
\left({\color{blue}\bullet}{\color{red}\bullet} \times \left\langle \begin{array}{l} A^{\color{blue}\bullet} = \text{false} \\ A^{\color{red}\bullet} = \text{true} \\ B^{\color{blue}\bullet} = \text{false} \\ B^{\color{red}\bullet} = \text{true} \end{array} \right\rangle = 0 \right)^{T}
$$

$$
-
\left({\color{blue}\bullet}{\color{blue}\bullet} \times \left\langle \begin{array}{l} A^{\color{blue}\bullet} = \text{false} \\ A^{\color{red}\bullet} = \text{true} \\ B^{\color{blue}\bullet} = {\color{red}\text{true}} \\ B^{\color{red}\bullet} = {\color{red}\text{false}} \end{array} \right\rangle = 0 \right)^{T}
+
\left({\color{blue}\bullet}{\color{red}\bullet} \times \left\langle \begin{array}{l} A^{\color{blue}\bullet} = \text{false} \\ A^{\color{red}\bullet} = \text{true} \\ B^{\color{blue}\bullet} = {\color{red}\text{true}} \\ B^{\color{red}\bullet} = {\color{red}\text{false}} \end{array} \right\rangle = 0 \right)^{T}
$$

Lower bound (part 2)

$$(\boldsymbol{v} - \boldsymbol{v}')(\boldsymbol{M} - \boldsymbol{M}') = 0$$

Lower bound (part 2)

$$(\boldsymbol{v} - \boldsymbol{v}')(\boldsymbol{M} - \boldsymbol{M}') = 0$$

Rest of proof:

- v, v' distinct (otherwise privacy can be violated)

$\Rightarrow$ $\dim(\ker(M - M')) \geq 1$

Lower bound (part 2)

$$(\boldsymbol{v} - \boldsymbol{v}')(\boldsymbol{M} - \boldsymbol{M}') = 0$$

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)

$\Rightarrow$ $\dim(\mathrm{rowspace}(M - M')) \geq 1$

Lower bound (part 2)

$$(\boldsymbol{v} - \boldsymbol{v}')(\boldsymbol{M} - \boldsymbol{M}') = 0$$

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 2λ bits

Just the beginning...

Theorem

A linear garbling scheme requires 2λ bits to garble a single AND *gate.*

Just the beginning...

Theorem

A linear garbling scheme requires 2λ 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 2λ 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

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

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

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

Limitations of model

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

Limitation:	**Opportunity:**
point-and-permute is sole source of non-linearity	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.: $H(H(A) \oplus B)$
linear!	maybe non-linear (but still practical) techniques lead to smaller GC? e.g., compute $GF(2^\lambda)$-inverse of a wire label

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!