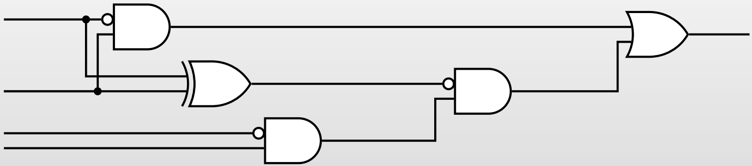
# Practical Garbled Circuit Optimizations

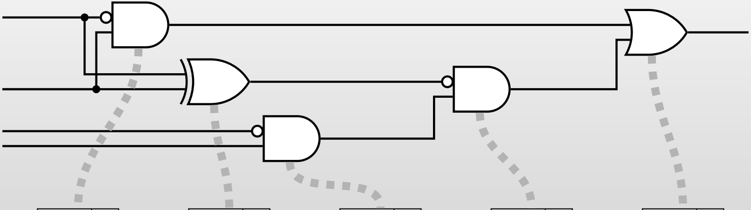
**Mike Rosulek**
Oregon State University OSU

Collaborators: David Evans / Vlad Kolesnikov / Payman Mohassel / Samee Zahur

# Garbled circuit framework [Yao86]



# Garbled circuit framework [Yao86]



| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

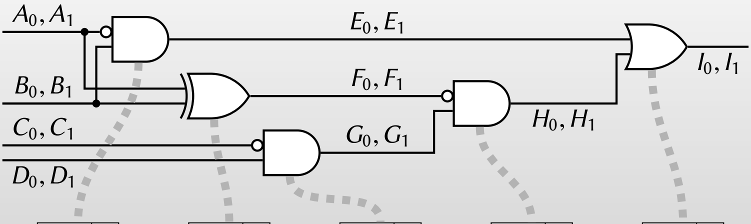
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| | | |
|---|---|---|
| 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 |

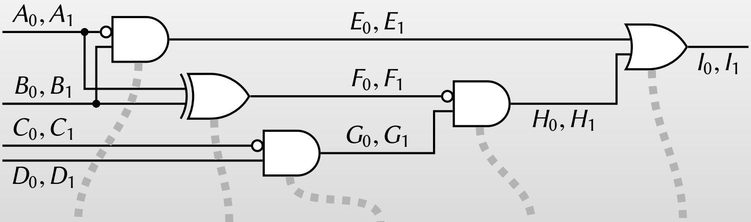
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

| | | |
|---|---|---|
| 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]




| | | |
|---|---|---|
| $A_0$ | $B_0$ | $E_0$ |
| $A_0$ | $B_1$ | $E_1$ |
| $A_1$ | $B_0$ | $E_0$ |
| $A_1$ | $B_1$ | $E_0$ |

| | | |
|---|---|---|
| $A_0$ | $B_0$ | $F_0$ |
| $A_0$ | $B_1$ | $F_1$ |
| $A_1$ | $B_0$ | $F_1$ |
| $A_1$ | $B_1$ | $F_0$ |

| | | |
|---|---|---|
| $C_0$ | $D_0$ | $G_0$ |
| $C_0$ | $D_1$ | $G_1$ |
| $C_1$ | $D_0$ | $G_0$ |
| $C_1$ | $D_1$ | $G_0$ |

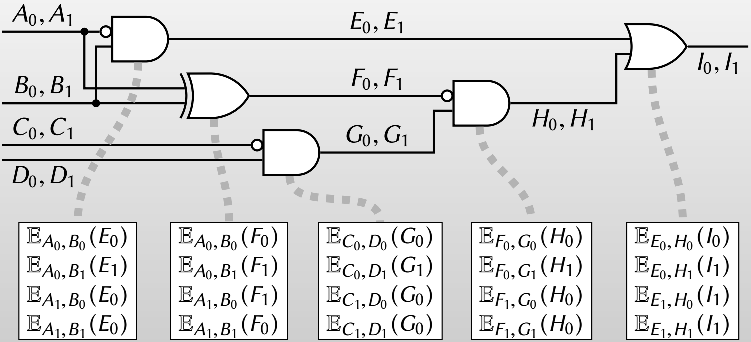
| | | |
|---|---|---|
| $F_0$ | $G_0$ | $H_0$ |
| $F_0$ | $G_1$ | $H_1$ |
| $F_1$ | $G_0$ | $H_0$ |
| $F_1$ | $G_1$ | $H_0$ |

| | | |
|---|---|---|
| $E_0$ | $H_0$ | $I_0$ |
| $E_0$ | $H_1$ | $I_1$ |
| $E_1$ | $H_0$ | $I_1$ |
| $E_1$ | $H_1$ | $I_1$ |

Garbling a circuit:

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

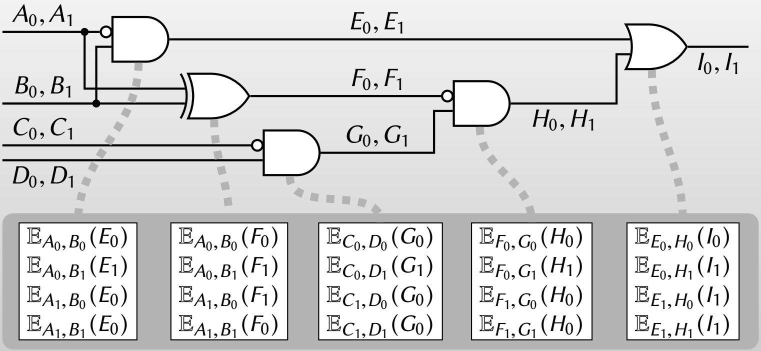
# Garbled circuit framework [Yao86]




Garbling a circuit:

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

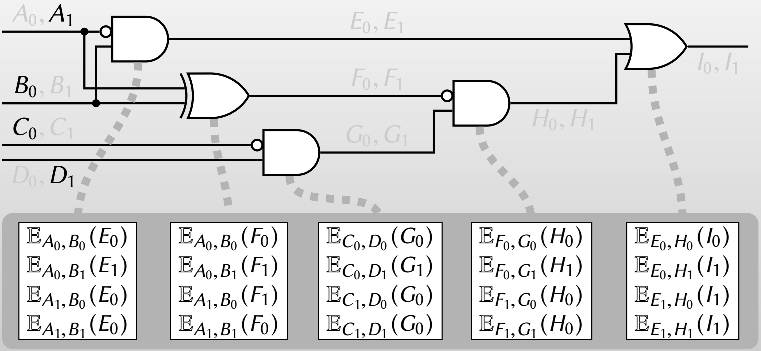
# Garbled circuit framework [Yao86]




Garbling a circuit:

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

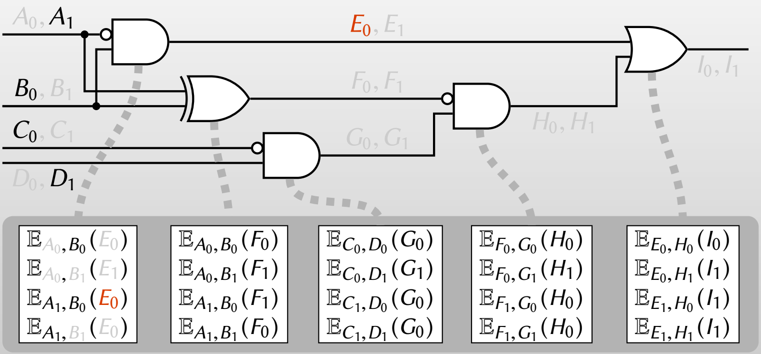
# Garbled circuit framework [Yao86]




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]



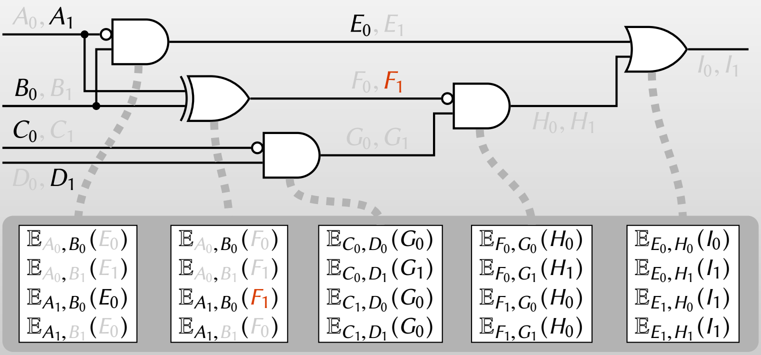

Garbling a circuit:

- 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]



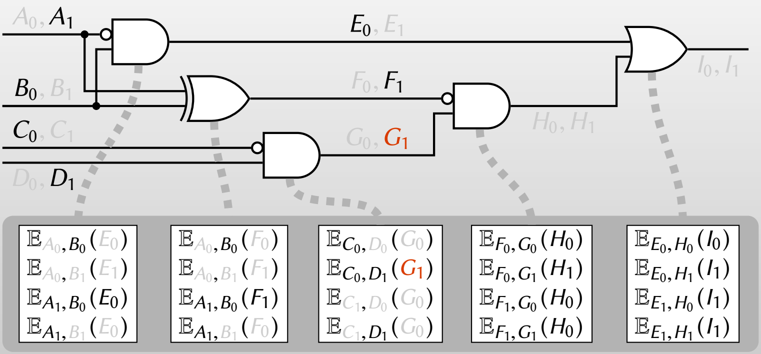

Garbling a circuit:

- 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]



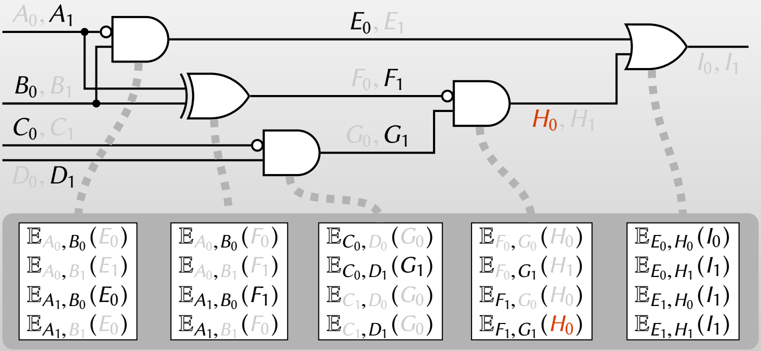

Garbling a circuit:

- 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]



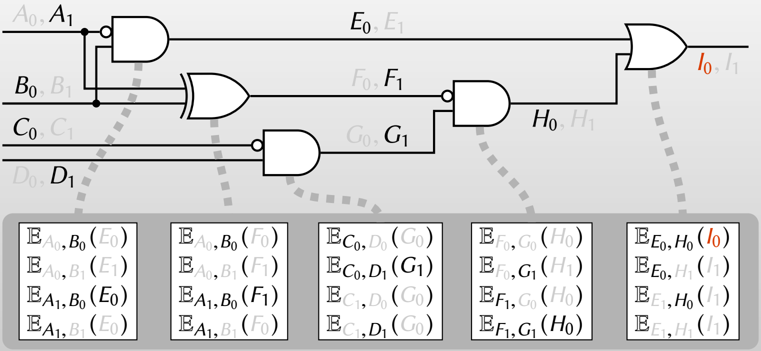

Garbling a circuit:

- 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: 2PC and more

$x$

$y$

# Applications: 2PC and more

garbled circuit $f$

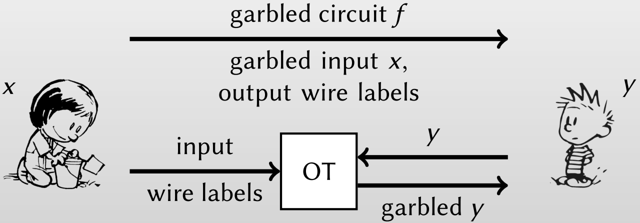
$\longrightarrow$

garbled input $x$,
output wire labels

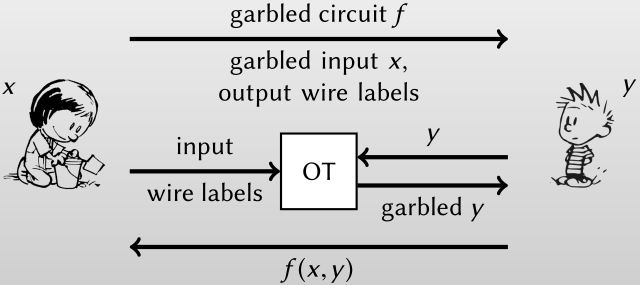
$x$

$y$

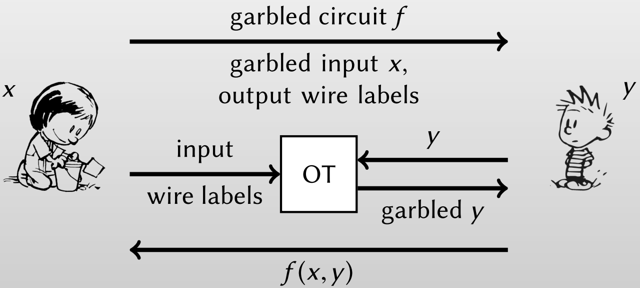
# Applications: 2PC and more



# Applications: 2PC and more

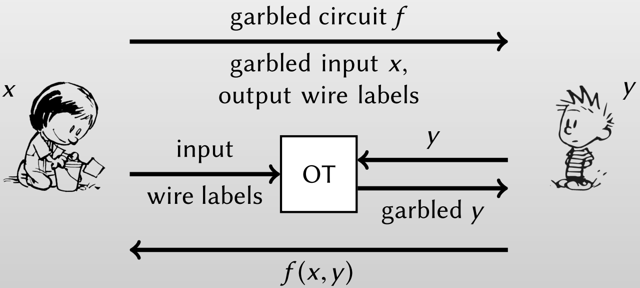


# Applications: 2PC and more




Private function evaluation, zero-knowledge proofs, encryption with key-dependent message security, randomized encodings, secure outsourcing, one-time programs, . . .

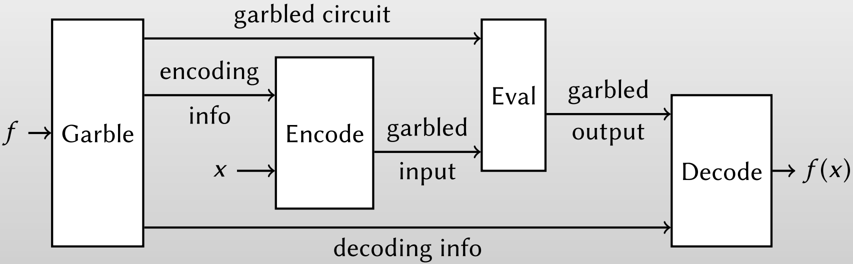
# Applications: 2PC and more



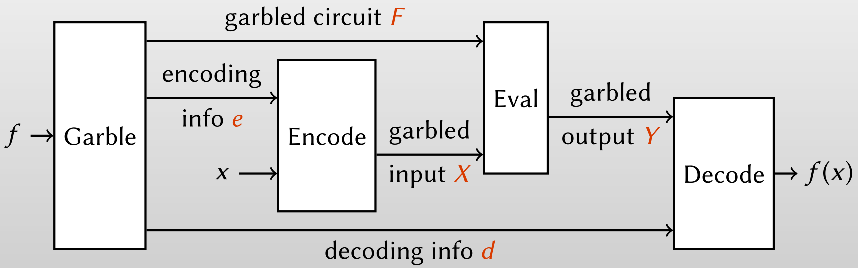

Private function evaluation, zero-knowledge proofs, encryption with key-dependent message security, randomized encodings, secure outsourcing, one-time programs, . . .

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

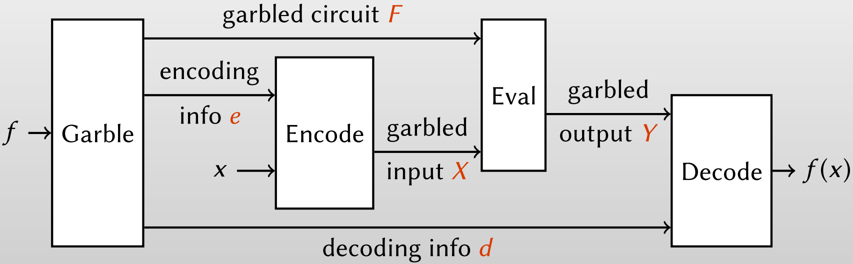
# Syntax [BellareHoangRogaway12]



# Syntax [BellareHoangRogaway12]



# Syntax [BellareHoangRogaway12]



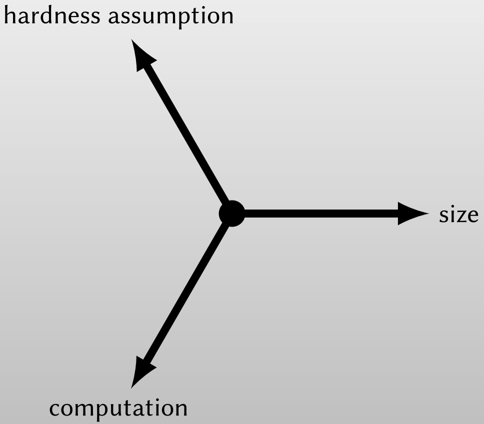

Security properties:

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

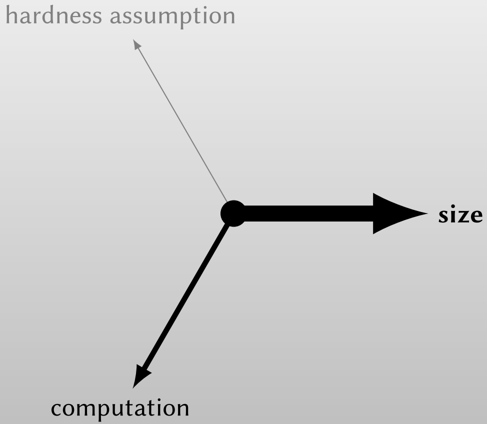
Obliviousness: $(F, X)$ reveals nothing

Authenticity: given $(F, X)$, hard to find $\widetilde{Y}$ that decodes $\notin \{f(x), \perp\}$

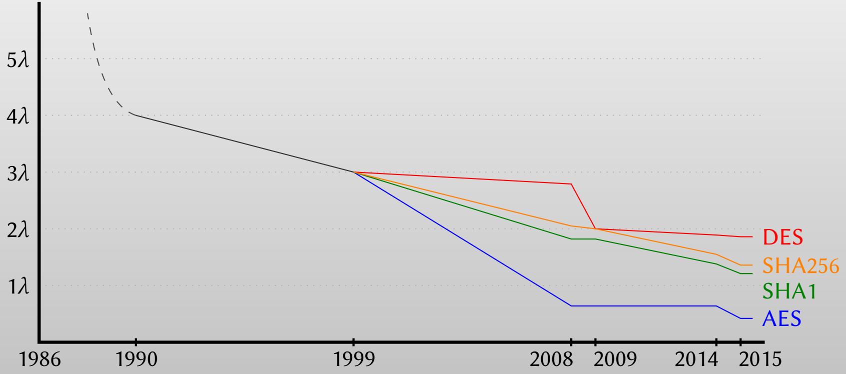
# Parameters to optimize



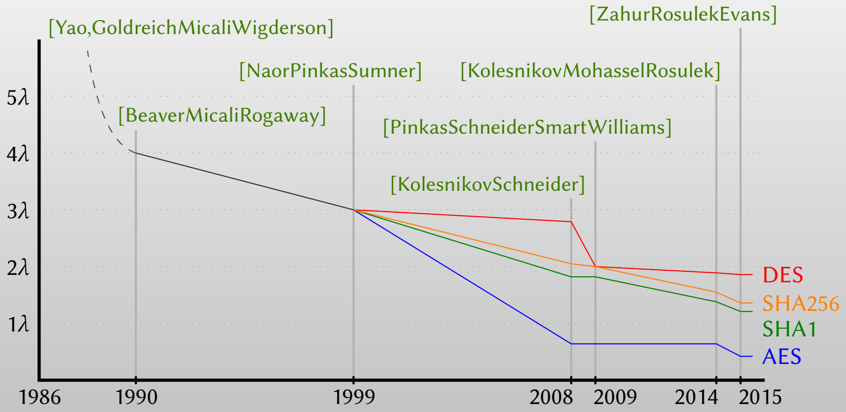
# Parameters to optimize



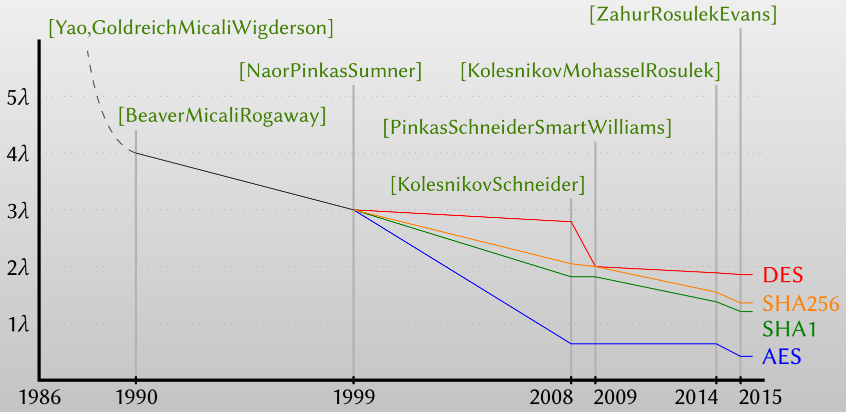
# Average bits per garbled gate



# Average bits per garbled gate

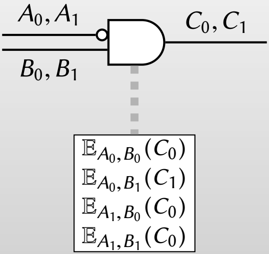


# Average bits per garbled gate



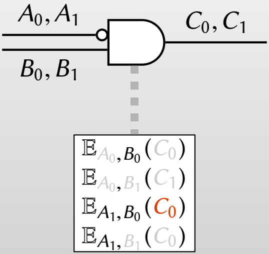

Prediction: by 2026, all garbled circuits will have zero size.

# Murky beginnings [Yao86]



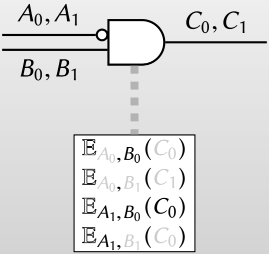

- Position in this list leaks semantic value

# Murky beginnings [Yao86]



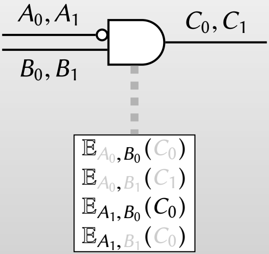

- Position in this list leaks semantic value

# Murky beginnings [Yao86]



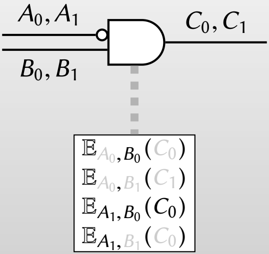

- Position in this list leaks semantic value $\implies$ permute ciphertexts

# Murky beginnings [Yao86]



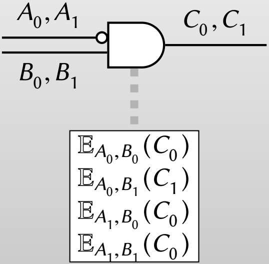

- Position in this list leaks semantic value $\implies$ permute ciphertexts
- Need to **detect** [in]correct decryption

# Murky beginnings [Yao86]

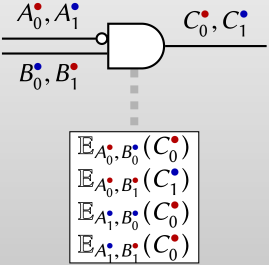



- Position in this list leaks semantic value $\implies$ permute ciphertexts
- Need to **detect** [in]correct decryption
- (Apparently) no one knows exactly what Yao had in mind:
  - $\mathbb{E}_{K_0,K_1}(M) = \langle E(K_0, S_0), E(K_1, S_1)\rangle$ where $S_0 \oplus S_1 = M$ [GoldreichMicaliWigderson87]
  - $\mathbb{E}_{K_0,K_1}(M) = E(K_1, E(K_0, M))$ [LindellPinkas09]

# Permute-and-Point [BeaverMicaliRogaway90]

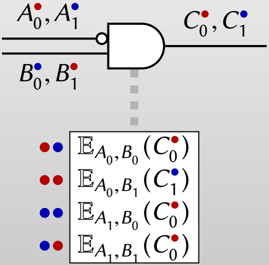


# Permute-and-Point [BeaverMicaliRogaway90]



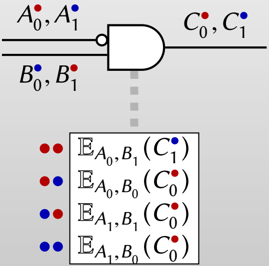

- Randomly assign $({\color{red}\bullet}, {\color{blue}\bullet})$ or $({\color{blue}\bullet}, {\color{red}\bullet})$ to each pair of wire labels
- Include color in the wire label (e.g., as last bit)

# Permute-and-Point [BeaverMicaliRogaway90]



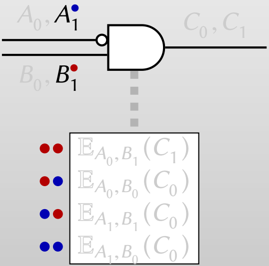

- Randomly assign (●, ●) or (●, ●) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys

# Permute-and-Point [BeaverMicaliRogaway90]



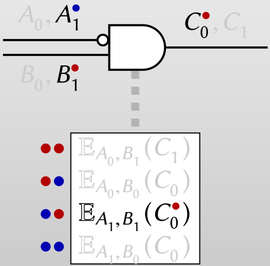

- Randomly assign (●, ●) or (●, ●) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys

# Permute-and-Point [BeaverMicaliRogaway90]



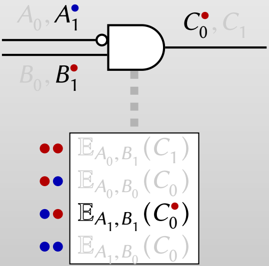

- Randomly assign (●, ●) or (●, ●) to each pair of wire labels
- Include color in the wire label (e.g., as last bit)
- Order the 4 ciphertexts canonically, by color of keys
- Evaluate by decrypting ciphertext indexed by your colors

# 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

# 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

Can use **one-time-secure** symmetric encryption!

# Computational cost of garbling

| $\mathbb{E}_{A,B}(C)$: | cost to garble AES |
|---|---|
| $\text{PRF}(A, \text{gateID}) \oplus \text{PRF}(B, \text{gateID}) \oplus C$ | ~6s [extrapolated] |
| [NaorPinkasSumner99] | time from Fairplay [MNPS04]: PRF = SHA256 |

# Computational cost of garbling

*2 hash $\gg$ 1 hash*

| $\mathbb{E}_{A,B}(C)$: | cost to garble AES |
|---|---|
| $\mathrm{PRF}(A, \mathrm{gateID}) \oplus \mathrm{PRF}(B, \mathrm{gateID}) \oplus C$ | ~6s [extrapolated] |
| [NaorPinkasSumner99] | time from Fairplay [MNPS04]: PRF = SHA256 |
| $\mathrm{H}(A \| B \| \mathrm{gateID}) \oplus C$ | 0.15s |
| [LindellPinkasSmart08] | time from [sS12]; H = SHA256 |

# Computational cost of garbling

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

| $\mathbb{E}_{A,B}(C)$: | cost to garble AES |
|---|---|
| $\mathrm{PRF}(A, \mathrm{gateID}) \oplus \mathrm{PRF}(B, \mathrm{gateID}) \oplus C$<br>[NaorPinkasSumner99] | ~6s [extrapolated]<br>time from Fairplay [MNPS04]: PRF = SHA256 |
| $\mathrm{H}(A\|B\|\mathrm{gateID}) \oplus C$<br>[LindellPinkasSmart08] | 0.15s<br>time from [sS12]; H = SHA256 |
| $\mathrm{AES256}(A\|B, \mathrm{gateID}) \oplus C$<br>[shelatShen12] | 0.12s |

# Computational cost of garbling

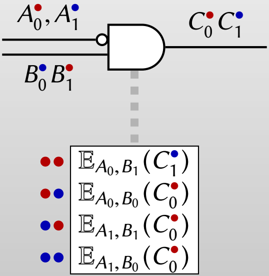
*2 hash ≫ 1 hash ≫ 1 block cipher ≫ 1 block cipher w/o key schedule*

| $\mathbb{E}_{A,B}(C)$: | cost to garble AES |
|---|---|
| $\text{PRF}(A, \text{gateID}) \oplus \text{PRF}(B, \text{gateID}) \oplus C$ <br> [NaorPinkasSumner99] | ~6s [extrapolated] <br> time from Fairplay [MNPS04]: PRF = SHA256 |
| $\text{H}(A \| B \| \text{gateID}) \oplus C$ <br> [LindellPinkasSmart08] | 0.15s <br> time from [sS12]; H = SHA256 |
| $\text{AES256}(A \| B, \text{gateID}) \oplus C$ <br> [shelatShen12] | 0.12s |
| $\text{AES}(\text{const}, K) \oplus K \oplus C$ <br> *where* $K = 2A \oplus 4B \oplus \textit{gateID}$ <br> [BellareHoangKeelveedhiRogaway13] | 0.0003s |

# Scoreboard

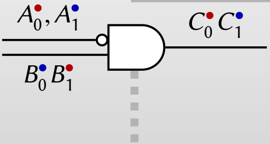
| | size ($\times\lambda$) | garble cost | eval cost | assumption |
|---|---|---|---|---|
| Classical | large? | 8 | 5 | PKE |
| P&P | 4 | 4/8 | 1/2 | hash/PRF |

# Garbled Row Reduction [NaorPinkasSumner99]



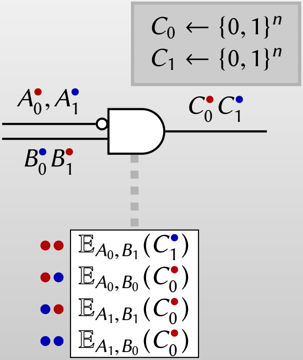
# Garbled Row Reduction [NaorPinkasSumner99]

$$C_0 \leftarrow \{0,1\}^n$$
$$C_1 \leftarrow \{0,1\}^n$$




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

# Garbled Row Reduction [NaorPinkasSumner99]

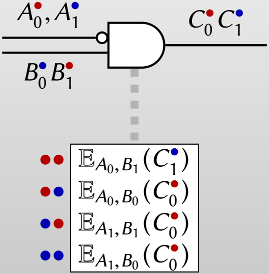



- What wire label will be payload of 1st (●●) ciphertext?

# Garbled Row Reduction [NaorPinkasSumner99]

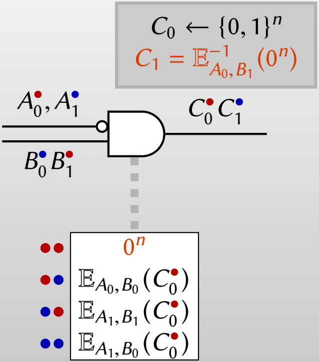
$C_0 \leftarrow \{0,1\}^n$

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




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

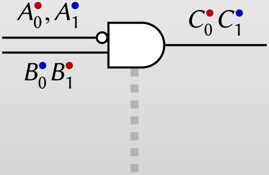
# Garbled Row Reduction [NaorPinkasSumner99]




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

# Garbled Row Reduction [NaorPinkasSumner99]

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

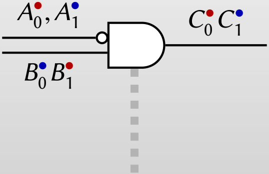



$$\mathbb{E}_{A_0,B_0}(C_0)$$
$$\mathbb{E}_{A_1,B_1}(C_0)$$
$$\mathbb{E}_{A_1,B_0}(C_0)$$

- 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]

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



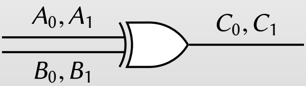

| | |
|---|---|
| ●● | $\mathbb{E}_{A_0,B_0}(C_0)$ |
| ●● | $\mathbb{E}_{A_1,B_1}(C_0)$ |
| ●● | $\mathbb{E}_{A_1,B_0}(C_0)$ |

- 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 ●●.

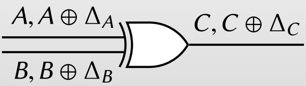
# Scoreboard

| | size ($\times\lambda$) | garble cost | eval cost | assumption |
|---|---|---|---|---|
| Classical | large? | 8 | 5 | PKE |
| P&P | 4 | 4/8 | 1/2 | hash/PRF |
| **GRR3** | **3** | **4/8** | **1/2** | **hash/PRF** |

# Free XOR [KolesnikovSchneider08]

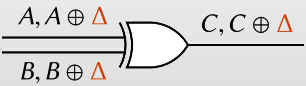


# Free XOR [KolesnikovSchneider08]



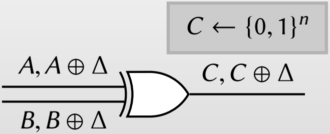

- Wire's **offset** ≡ XOR of its two labels

# Free XOR [KolesnikovSchneider08]




- Wire's **offset** ≡ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$

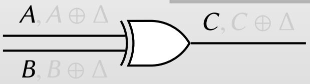
# Free XOR [KolesnikovSchneider08]




- Wire's **offset** ≡ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$

# Free XOR [KolesnikovSchneider08]

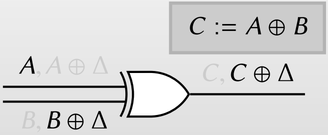
$C := A \oplus B$




$$\underbrace{A}_{\text{FALSE}} \oplus \underbrace{B}_{\text{FALSE}} = \underbrace{A \oplus B}_{\text{FALSE}}$$

- Wire's **offset** ≡ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose FALSE output = FALSE input ⊕ FALSE input

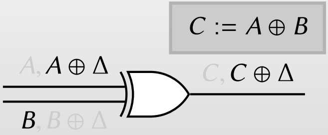
# Free XOR [KolesnikovSchneider08]




$$\underbrace{A\quad}_{\text{FALSE}} \oplus \underbrace{B \oplus \Delta}_{\text{TRUE}} = \underbrace{A \oplus B \oplus \Delta}_{\text{TRUE}}$$

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

# Free XOR [KolesnikovSchneider08]

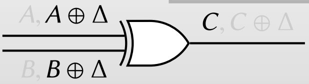



$$\underbrace{A \oplus \Delta}_{\text{TRUE}} \oplus \underbrace{B \quad}_{\text{FALSE}} = \underbrace{A \oplus B \oplus \Delta}_{\text{TRUE}}$$

- Wire's **offset** ≡ XOR of its two labels
- Choose all wires to have same (secret) offset Δ
- Choose FALSE output = FALSE input ⊕ FALSE input
- Evaluate by XORing input wire labels (no crypto)

# Free XOR [KolesnikovSchneider08]

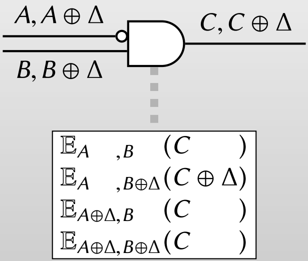
$$C := A \oplus B$$




$$\underbrace{A \oplus \Delta}_{\text{TRUE}} \oplus \underbrace{B \oplus \Delta}_{\text{TRUE}} = \underbrace{A \oplus B \quad}_{\text{FALSE}}$$

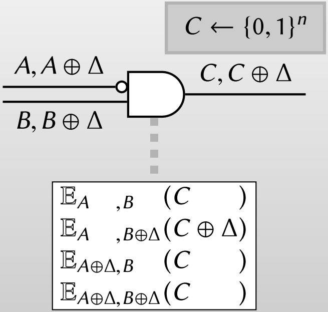
- Wire's **offset** ≡ XOR of its two labels
- Choose all wires to have same (secret) offset $\Delta$
- Choose FALSE output = FALSE input ⊕ FALSE input
- Evaluate by XORing input wire labels (no crypto)

# Freedom at a cost. . .



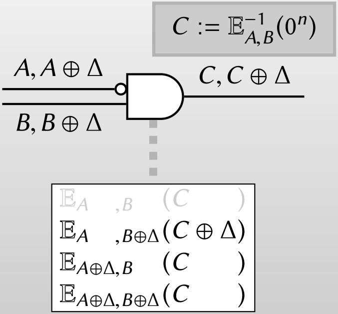

- Still need to garble AND gates

# Freedom at a cost. . .



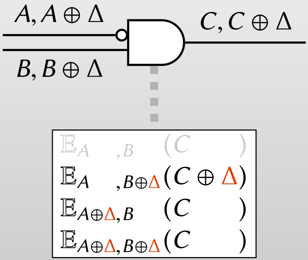

- Still need to garble AND gates
- Compatible with garbled row-reduction

# Freedom at a cost. . .



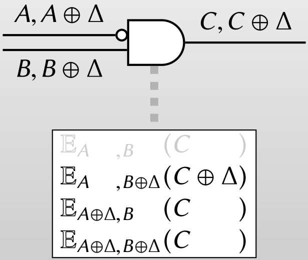

- Still need to garble AND gates
- Compatible with garbled row-reduction

# 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. . .



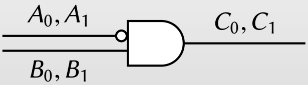

- Still need to garble AND gates
- Compatible with garbled row-reduction
- Secret $\Delta$ used in key and payload of ciphertexts!
- Requires related-key + circularity assumption [ChoiKatzKumaresanZhou12]

# Scoreboard

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| **Free XOR** | **0** | **3** | **0** | **4** | **0** | **1** | **circ. hash** |

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

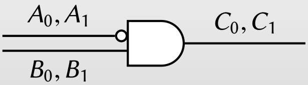


# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

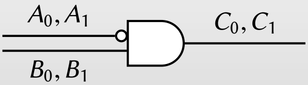


# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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



# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

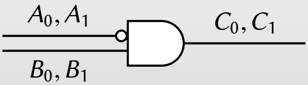
- Evaluator can know exactly one of:

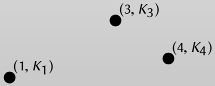
$$K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow \text{learn } C_0$$

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

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

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




$$(1, K_1), (3, K_3), (4, K_4)$$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

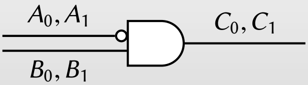
- Evaluator can know exactly one of:

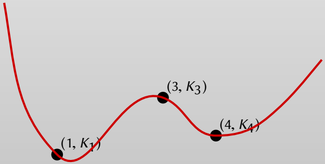
$$K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow \text{learn } C_0$$

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

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

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




$P$ = uniq deg-2 poly thru

$(1, K_1), (3, K_3), (4, K_4)$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

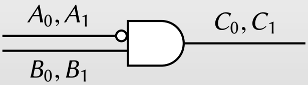
- Evaluator can know exactly one of:

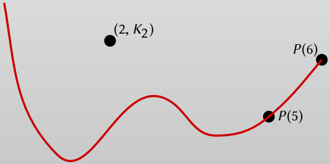
$$K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow \text{learn } C_0$$

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

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

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




$P$ = uniq deg-2 poly thru

$(1, K_1), (3, K_3), (4, K_4)$

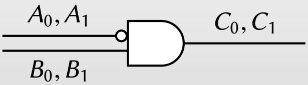
$(2, K_2), (5, P(5)), (6, P(6))$

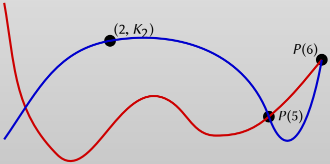
# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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




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

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

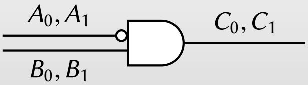
# Row reduction ++ [PinkasSchneiderSmartWilliams09]

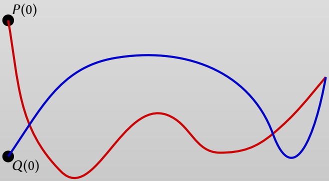
$C_0 = P(0); C_1 = Q(0)$

Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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




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

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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

Garbled gates with only **2** ciphertexts!

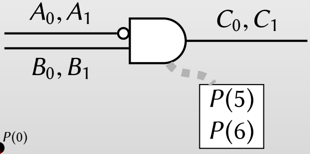
- Evaluator can know exactly one of:

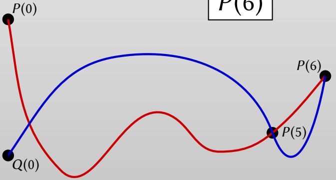
$$K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow \text{learn } C_0$$

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

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

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




$P =$ uniq deg-2 poly thru

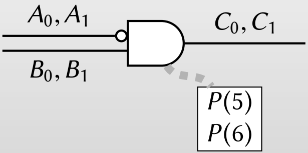
$(1, K_1), (3, K_3), (4, K_4)$

$Q =$ uniq deg-2 poly thru

$(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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

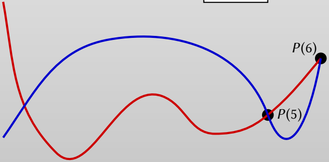



Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

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




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

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

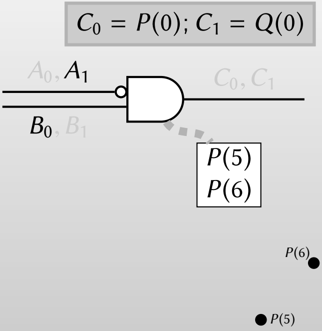
# Row reduction ++ [PinkasSchneiderSmartWilliams09]

Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

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

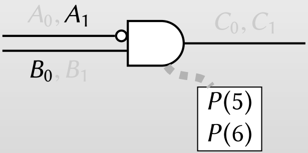



$P$ = uniq deg-2 poly thru $(1, K_1), (3, K_3), (4, K_4)$

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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

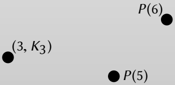



Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

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

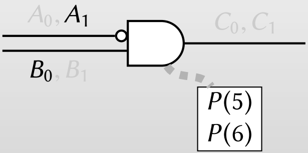



$P$ = uniq deg-2 poly thru $(1, K_1), (3, K_3), (4, K_4)$

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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




Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

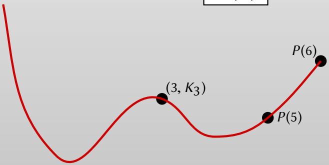
$$K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow \text{learn } C_0$$

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

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

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

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




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

$Q =$ 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:

  $K_1 = \mathbb{E}^{-1}_{A_0,B_0}(0^n) \rightsquigarrow$ learn $C_0$

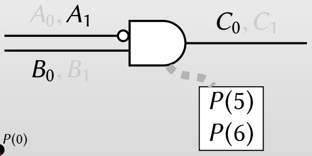
  $K_2 = \mathbb{E}^{-1}_{A_0,B_1}(0^n) \rightsquigarrow$ learn $C_1$

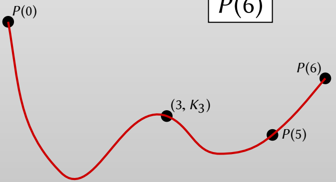
  $K_3 = \mathbb{E}^{-1}_{A_1,B_0}(0^n) \rightsquigarrow$ learn $C_0$

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

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

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



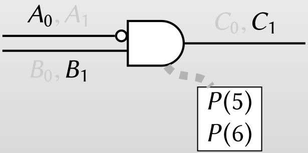




$P$ = uniq deg-2 poly thru $(1, K_1), (3, K_3), (4, K_4)$

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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

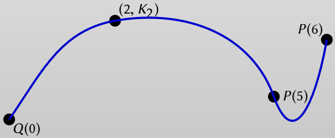



Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

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

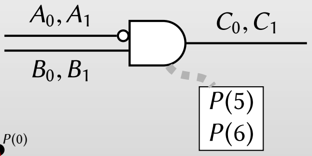



$P$ = uniq deg-2 poly thru $(1, K_1), (3, K_3), (4, K_4)$

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Row reduction ++ [PinkasSchneiderSmartWilliams09]

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

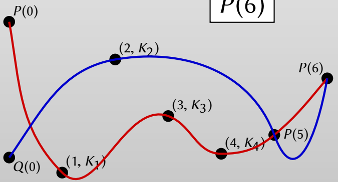



Garbled gates with only **2** ciphertexts!

- Evaluator can know exactly one of:

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

- Evaluate by interpolating poly thru $K_i$, $P(5)$ and $P(6)$
- **Incompatible** with Free-XOR: can't ensure $C_0 \oplus C_1 = \Delta$




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

$Q$ = uniq deg-2 poly thru $(2, K_2), (5, P(5)), (6, P(6))$

# Scoreboard

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| Free XOR | 0 | 3 | 0 | 4 | 0 | 1 | circ. hash |
| **GRR2** | **2** | **2** | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |

# FleXOR [KolesnikovMohasselRosulek14]

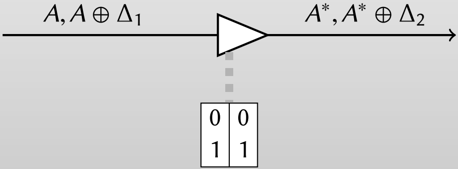
$$A, A \oplus \Delta_1$$
$$\longrightarrow$$

# FleXOR [KolesnikovMohasselRosulek14]

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

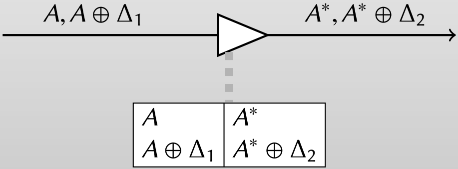
- Translate to a new wire offset

# FleXOR [KolesnikovMohasselRosulek14]



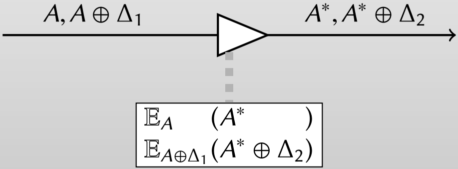

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]



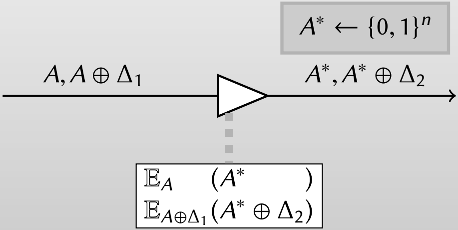

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]



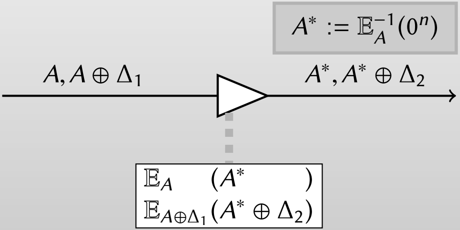

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]



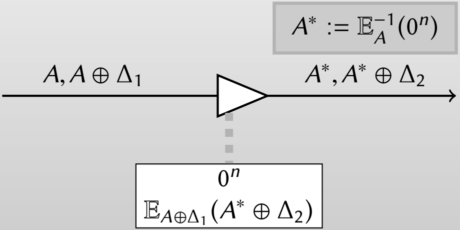

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]



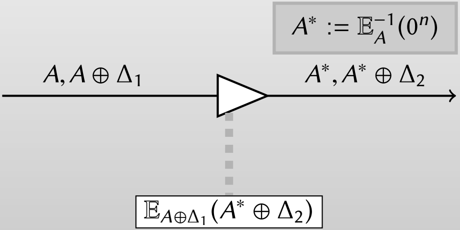

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]



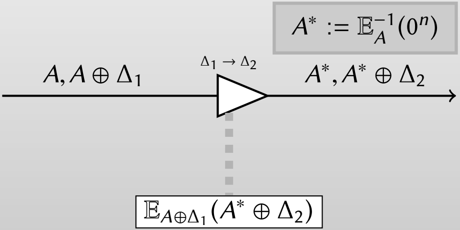

- Translate to a new wire offset (unary $a \mapsto a$ gate)

# FleXOR [KolesnikovMohasselRosulek14]




- Translate to a new wire offset (unary $a \mapsto a$ gate) using 1 ciphertext

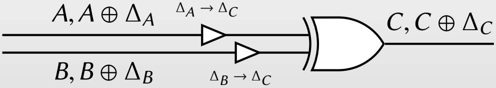
# FleXOR [KolesnikovMohasselRosulek14]




- Translate to a new wire offset (unary $a \mapsto a$ gate) using 1 ciphertext

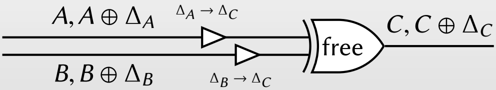
# FleXOR [KolesnikovMohasselRosulek14]

# FleXOR [KolesnikovMohasselRosulek14]



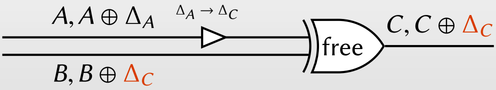

- Adjust inputs to target offset $\Delta_C$ (1 ciphertext each)

# FleXOR [KolesnikovMohasselRosulek14]



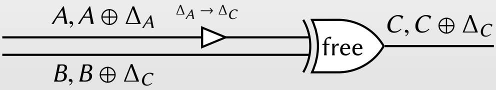

- Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free

# FleXOR [KolesnikovMohasselRosulek14]



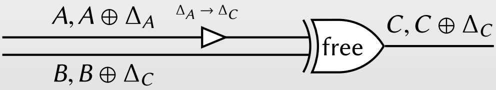

- Adjust inputs to target offset $\Delta_C$ (1 ciphertext each), then XOR is free
- If input wire already suitable, no need to adjust

# FleXOR [KolesnikovMohasselRosulek14]




- 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 [KolesnikovMohasselRosulek14]




- 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

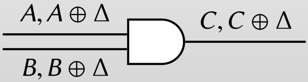
- Subj. to compatibility with 2-ciphertext row-reduction of AND gates
- (or) Subj. to removing circularity property of free-XOR

# Scoreboard

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| Free XOR | 0 | 3 | 0 | 4 | 0 | 1 | circ. hash |
| GRR2 | 2 | 2 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| **FleXOR** | $\{0, 1, 2\}$ | **2** | $\{0, 1, 2\}$ | **4** | $\{0, 1, 2\}$ | **1** | **circ. hash** |

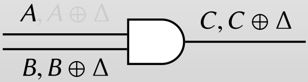
# Half Gates [ZahurRosulekEvans15]

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



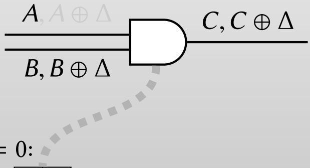
# Half Gates [ZahurRosulekEvans15]

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



# Half Gates [ZahurRosulekEvans15]

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



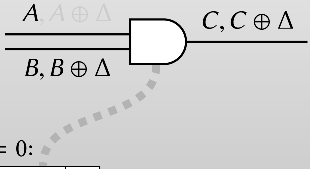

if $a = 0$:

| | |
|---|---|
| 0 | 0 |
| 1 | 0 |

unary gate $b \mapsto 0$

# Half Gates [ZahurRosulekEvans15]

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



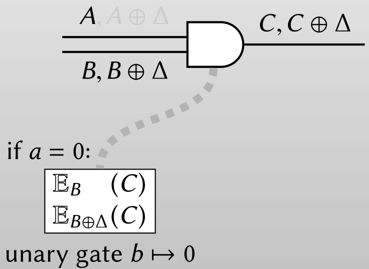

if $a = 0$:

| | |
|---|---|
| $B$ | $C$ |
| $B \oplus \Delta$ | $C$ |

unary gate $b \mapsto 0$

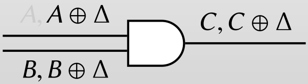
# Half Gates [ZahurRosulekEvans15]

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



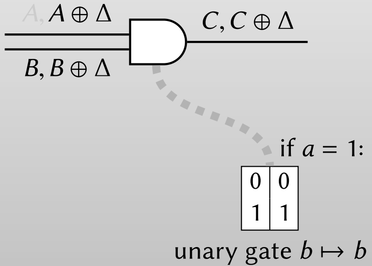
# Half Gates [ZahurRosulekEvans15]

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



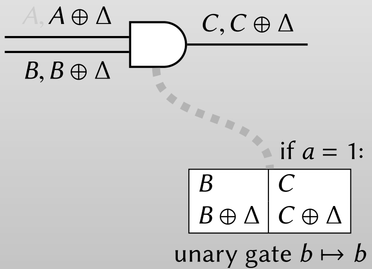
# Half Gates [ZahurRosulekEvans15]

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



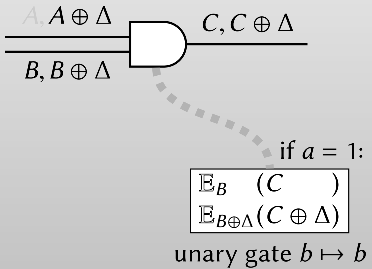
# Half Gates [ZahurRosulekEvans15]

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



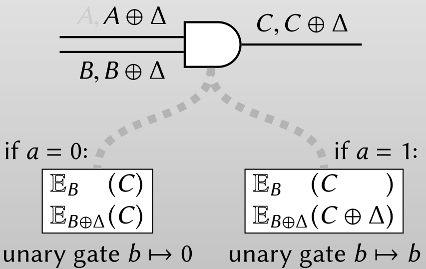
# Half Gates [ZahurRosulekEvans15]

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



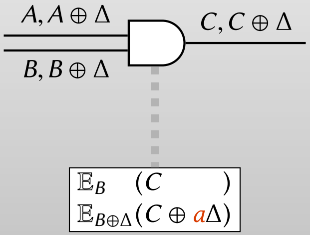
# Half Gates [ZahurRosulekEvans15]

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



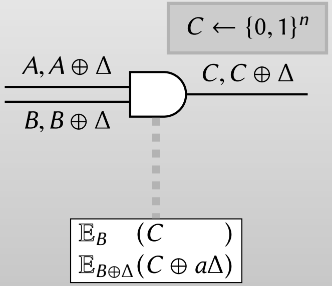
# Half Gates [ZahurRosulekEvans15]

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



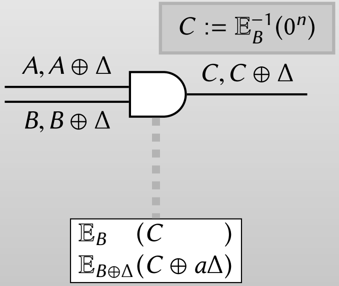
# Half Gates [ZahurRosulekEvans15]

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



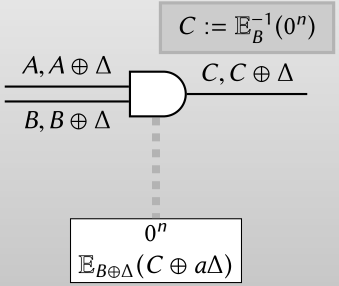
# Half Gates [ZahurRosulekEvans15]

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



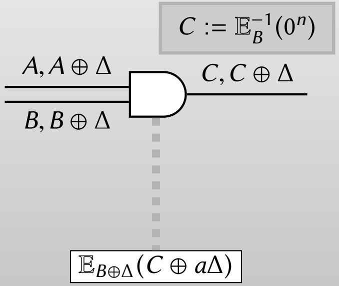
# Half Gates [ZahurRosulekEvans15]

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



# Half Gates [ZahurRosulekEvans15]

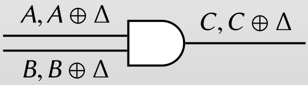
*What if garbler knows in advance the truth value on one input wire?*




Fine print: permute ciphertexts with permute-and-point.

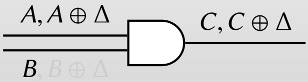
# Half Gates [ZahurRosulekEvans15]

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



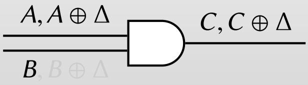
# Half Gates [ZahurRosulekEvans15]

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



# Half Gates [ZahurRosulekEvans15]

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

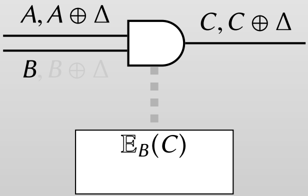



Evaluator has $B$ (knows FALSE):

⇒ should obtain $C$ (FALSE)

# Half Gates [ZahurRosulekEvans15]

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

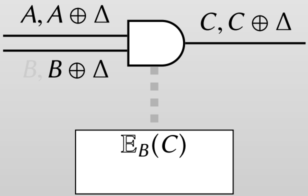



Evaluator has $B$ (knows FALSE):

$\Rightarrow$ should obtain $C$ (FALSE)

# Half Gates [ZahurRosulekEvans15]

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



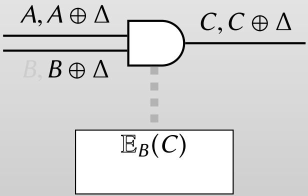

Evaluator has $B$ (knows FALSE):

$\Rightarrow$ should obtain $C$ (FALSE)

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

# Half Gates [ZahurRosulekEvans15]

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




Evaluator has $B$ (knows FALSE):

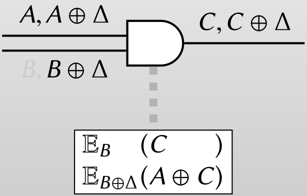
⇒ should obtain $C$ (FALSE)

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

⇒ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

# Half Gates [ZahurRosulekEvans15]

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




Evaluator has $B$ (knows FALSE):

$\Rightarrow$ should obtain $C$ (FALSE)

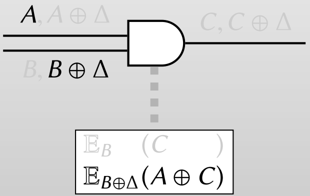
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

- 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)

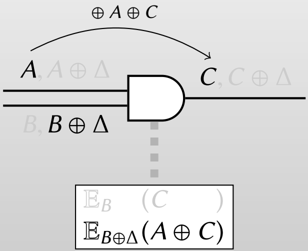
Evaluator has $B \oplus \Delta$ (knows TRUE):

⇒ 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):

$\Rightarrow$ should obtain $C$ (FALSE)

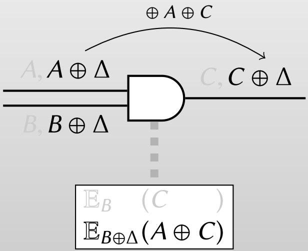
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "*a*" wire to "*c*" wire

- 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):

$\Rightarrow$ should obtain $C$ (FALSE)

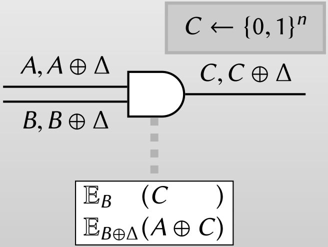
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

- 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):

$\Rightarrow$ should obtain $C$ (FALSE)

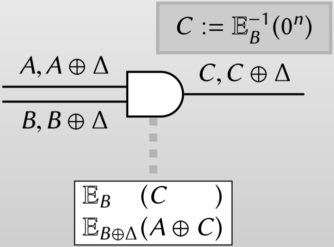
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "*a*" wire to "*c*" wire

- 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):

$\Rightarrow$ should obtain $C$ (FALSE)

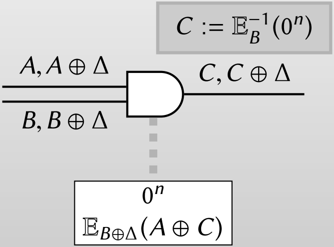
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

- 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):

$\Rightarrow$ should obtain $C$ (FALSE)

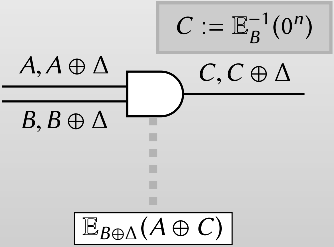
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

- 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):

$\Rightarrow$ should obtain $C$ (FALSE)

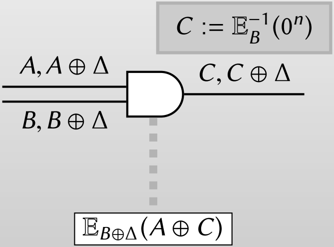
Evaluator has $B \oplus \Delta$ (knows TRUE):

$\Rightarrow$ should be able to *transfer* truth value from "$a$" wire to "$c$" wire

- 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):

$\Rightarrow$ 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

- Suffices to learn $A \oplus C$

Fine print: no need for permute-and-point here

# Two halves make a whole!

$$a \wedge b$$

# Two halves make a whole!

$$a \wedge b \; = (a \oplus r \oplus r) \wedge b$$

- Garbler chooses random bit $r$

# Two halves make a whole!

$$\begin{aligned} a \wedge b &= (a \oplus r \oplus r) \wedge b \\ &= [(a \oplus r) \wedge b] \oplus [r \wedge b] \end{aligned}$$

- Garbler chooses random bit $r$

# Two halves make a whole!

$$\begin{aligned} a \wedge b &= (a \oplus r \oplus r) \wedge b \\ &= [(a \oplus r) \wedge b] \oplus [r \wedge b] \end{aligned}$$

- Garbler chooses random bit $r$
- Arrange for evaluator to learn $a \oplus r$ in the clear

# Two halves make a whole!

$$\begin{aligned} a \wedge b &= (a \oplus r \oplus r) \wedge b \\ &= \underbrace{[(a \oplus r) \wedge b]}_{\text{one input known to evaluator}} \oplus [r \wedge b] \end{aligned}$$

- Garbler chooses random bit $r$
- Arrange for evaluator to learn $a \oplus r$ in the clear

# Two halves make a whole!

$$a \wedge b = (a \oplus r \oplus r) \wedge b$$
$$= [(a \oplus r) \wedge b] \oplus \underbrace{[r \wedge b]}_{\text{one input known to garbler}}$$

- Garbler chooses random bit $r$
- Arrange for evaluator to learn $a \oplus r$ in the clear

# Two halves make a whole!

$$\begin{aligned} a \wedge b &= (a \oplus r \oplus r) \wedge b \\ &= [(a \oplus r) \wedge b] \oplus \underbrace{[r \wedge b]}_{\text{one input known to garbler}} \end{aligned}$$

- 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!

$$a \wedge b = (a \oplus r \oplus r) \wedge b$$
$$= [(a \oplus r) \wedge b] \oplus \underbrace{[r \wedge b]}_{\text{one input known to garbler}}$$

- Garbler chooses random bit $r$
  - $r$ = color bit of 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

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| Free XOR | 0 | 3 | 0 | 4 | 0 | 1 | circ. hash |
| GRR2 | 2 | 2 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| FleXOR | $\{0, 1, 2\}$ | 2 | $\{0, 1, 2\}$ | 4 | $\{0, 1, 2\}$ | 1 | circ. symm |
| **HalfGates** | **0** | **2** | **0** | **4** | **0** | **2** | **circ. hash** |

# Scoreboard

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| Free XOR | 0 | 3 | 0 | 4 | 0 | 1 | circ. hash |
| GRR2 | 2 | 2 | 4/8 | 4/8 | 1/2 | 1/2 | PRF/hash |
| FleXOR | $\{0,1,2\}$ | 2 | $\{0,1,2\}$ | 4 | $\{0,1,2\}$ | 1 | circ. symm |
| **HalfGates** | **0** | **2** | **0** | **4** | **0** | **2** | **circ. hash** |
| [XYZ26]? | 0 | $<2$? | ? | ? | ? | ? | ? |

# Optimality

**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)

# Optimality

**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)

**Theorem ([ZahurRosulekEvans15])**

*Garbling a single* AND *gate requires 2 ciphertexts ($2\lambda$ bits), if garbling scheme is "linear" in this sense.*

# Optimality

**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)

**Theorem ([ZahurRosulekEvans15])**

*Garbling a single* AND *gate requires 2 ciphertexts ($2\lambda$ bits), if garbling scheme is "linear" in this sense.*

Half-gates construction is *size-optimal* among schemes that:

... use "known techniques"

... work gate-by-gate in {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 {XOR, AND, NOT} basis?

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

# Ways forward?

1: Consider larger "chunks" of circuit, beyond {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 {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.

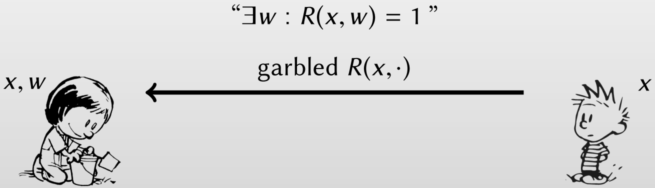
# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

“$\exists w : R(x, w) = 1$”

$x, w$

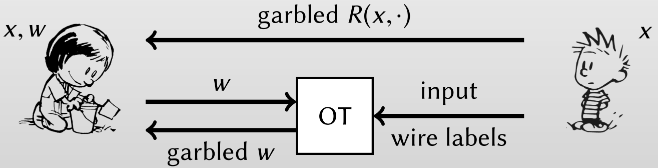
$x$

# ZK via garbled circuits [JawurekKerschbaumOrlandi13]



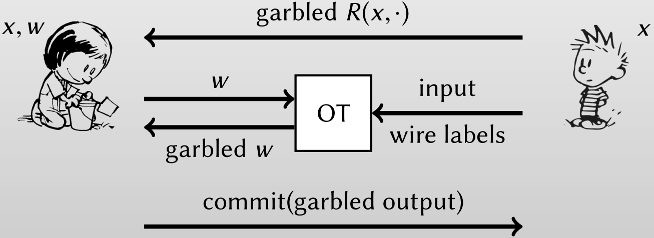
# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"$\exists w : R(x, w) = 1$"



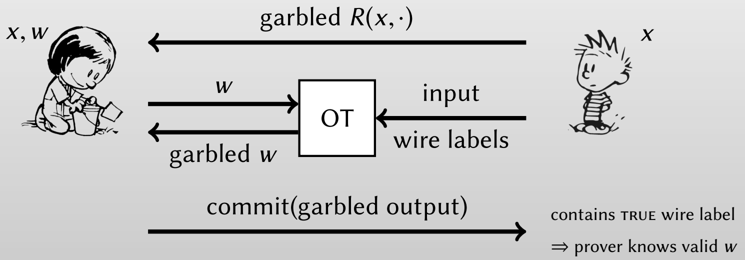
# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

“$\exists w : R(x, w) = 1$”



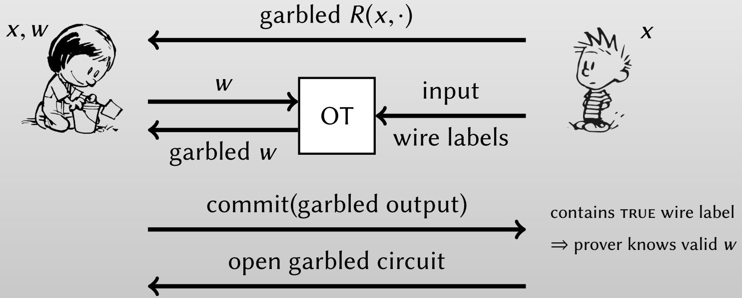
# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"$\exists w : R(x, w) = 1$"



# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

$$\text{“}\exists w : R(x, w) = 1\text{”}$$



# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"$\exists w : R(x, w) = 1$"

$x, w$

garbled $R(x, \cdot)$

$w$

OT

input

wire labels

garbled $w$

$x$

commit(garbled output)

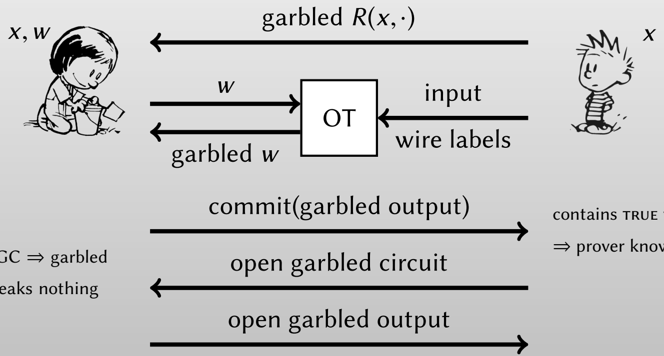
contains TRUE wire label $\Rightarrow$ prover knows valid $w$

open garbled circuit

correct GC $\Rightarrow$ garbled output leaks nothing about $w$

# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"$\exists w : R(x, w) = 1$"



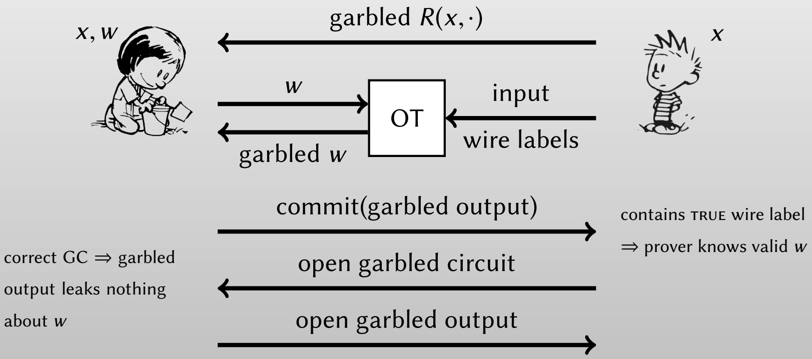

contains TRUE wire label

$\Rightarrow$ prover knows valid $w$

correct GC $\Rightarrow$ garbled output leaks nothing about $w$

# ZK via garbled circuits [JawurekKerschbaumOrlandi13]

"$\exists w : R(x, w) = 1$"




Prover knows entire input to garbled circuit!

# 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

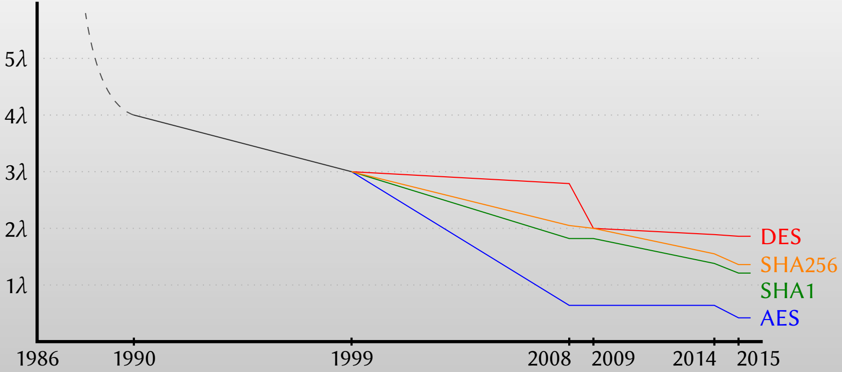
# Privacy-free garbling [FrederiksenNielsenOrlandi15]

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

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

| | size ($\times\lambda$) | | garble cost | | eval cost | | assumption |
|---|---|---|---|---|---|---|---|
| | XOR | AND | XOR | AND | XOR | AND | |
| Classical | large? | | 8 | | 5 | | PKE |
| P&P | 4 | 4 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| GRR3 | 3 | 3 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| Free XOR | 0 | 3 | 0 | 4 | 0 | 1 | circ. hash |
| GRR2 | 2 | 2 | 4/8 | 4/8 | 1/2 | 1/2 | hash/PRF |
| FleXOR | $\{0,1,2\}$ | 2 | $\{0,1,2\}$ | 4 | $\{0,1,2\}$ | 1 | circ. hash |
| HalfGates | 0 | 2 | 0 | 4 | 0 | 2 | circ. hash |
| **PrivFree** * | **0** | **1** | **0** | **2** | **0** | **1** | circ. hash |

# A success story!




- Reduction in size by 10x
- Reduction in computation by 10000x

# the end!