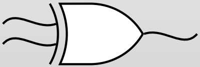
flexOR: flexible garbling for XOR gates that beats free-XOR

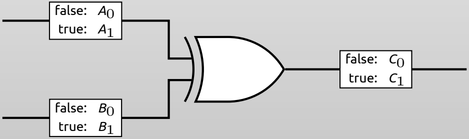


- ▷ Vladimir Kolesnikov ≫ Alcatel·Lucent
- ▷ Payman Mohassel ≫ University of Calgary
- ▶ Mike Rosulek ≫ Oregon State University OSU

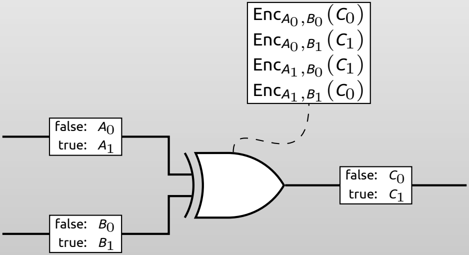
background

1

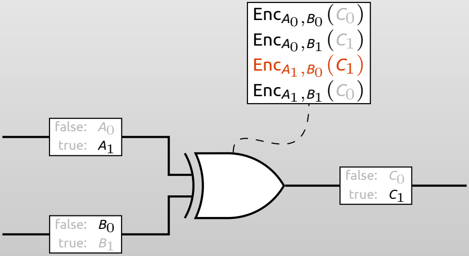
background: garbled circuit



background: garbled circuit

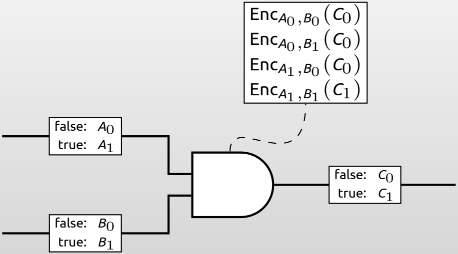


background: garbled circuit



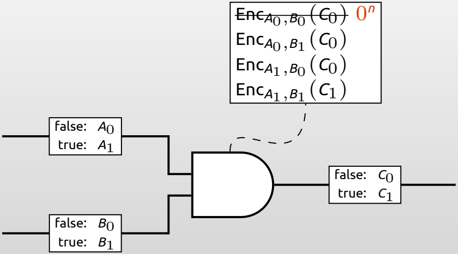
background: row reduction

background: row reduction



Garbled row reduction [NaorPinkasSumner99,PinkasSchneiderSmartWilliams09]

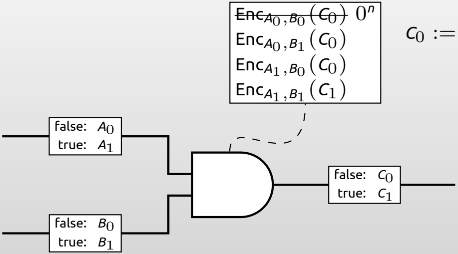
background: row reduction



Garbled row reduction [NaorPinkasSumner99,PinkasSchneiderSmartWilliams09]

- Fix one of the ciphertexts to be all zeroes

background: row reduction

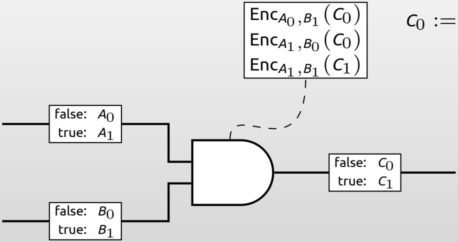


$C_0 := \mathsf{Dec}_{A_0,B_0}(0^n)$

Garbled row reduction [NaorPinkasSumner99,PinkasSchneiderSmartWilliams09]

- Fix one of the ciphertexts to be all zeroes
- Corresponding wire label must be $\mathsf{Dec}(0^n)$, not uniform

background: row reduction



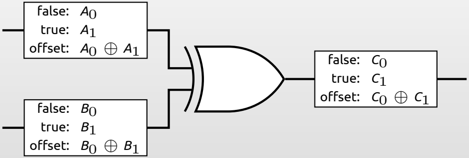
$C_0 := \mathsf{Dec}_{A_0,B_0}(0^n)$

Garbled row reduction [NaorPinkasSumner99,PinkasSchneiderSmartWilliams09]

- Fix one of the ciphertexts to be all zeroes
- Corresponding wire label must be $\mathsf{Dec}(0^n)$, not uniform
- Only 3 ciphertexts needed for garbled gate
- More advanced technique reduces size to 2 ciphertexts

background: offsets & free XOR

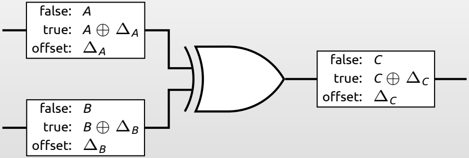
background: offsets & free XOR



Definition

Offset of a wire = XOR of its two wire labels

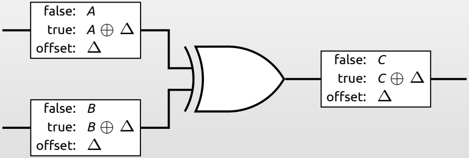
background: offsets & free XOR



Definition

Offset of a wire = XOR of its two wire labels

background: offsets & free XOR



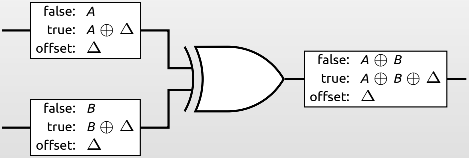
Definition

Offset of a wire = XOR of its two wire labels

Free XOR optimization [KolesnikovSchneider08]:

- all wires have *same* (secret) offset Δ

background: offsets & free XOR



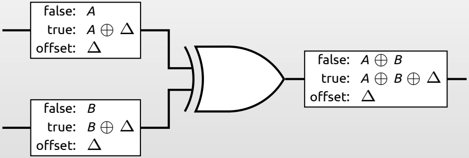
Definition

Offset of a wire = XOR of its two wire labels

Free XOR optimization [KolesnikovSchneider08]:

- all wires have *same* (secret) offset Δ
- wire *labels* for XOR gate satisfy $C = A \oplus B$

background: offsets & free XOR



Definition

Offset of a wire = XOR of its two wire labels

Free XOR optimization [KolesnikovSchneider08]:

- all wires have *same* (secret) offset Δ
- wire *labels* for XOR gate satisfy $C = A \oplus B$
- compute output wire label by XOR'ing input wire labels (no crypto!)

free XOR

Free XOR limitations:

1. Requires strong circularity hardness assumption [ChoiKatzKumaresanZhou12]
2. Incompatible with 4-to-2 row reduction [PinkasSchneiderSmartWilliams09]

free XOR

Free XOR limitations:

1. Requires strong circularity hardness assumption [ChoiKatzKumaresanZhou12]
2. Incompatible with 4-to-2 row reduction [PinkasSchneiderSmartWilliams09]

Motivating Question

Can we overcome these limitations, while retaining Free XOR's benefits (as much as possible)?

free XOR

Free XOR limitations:

1. Requires strong circularity hardness assumption [ChoiKatzKumaresanZhou12]
2. Incompatible with 4-to-2 row reduction [PinkasSchneiderSmartWilliams09]

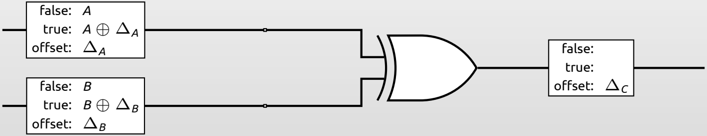
Motivating Question

Can we overcome these limitations, while retaining Free XOR's benefits (as much as possible)? *Hint: yes!*

fleXOR garbling

2

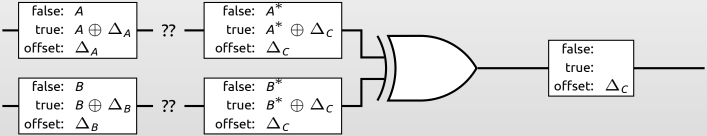
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C

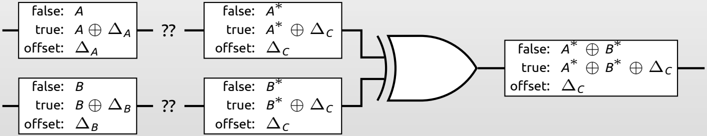
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C

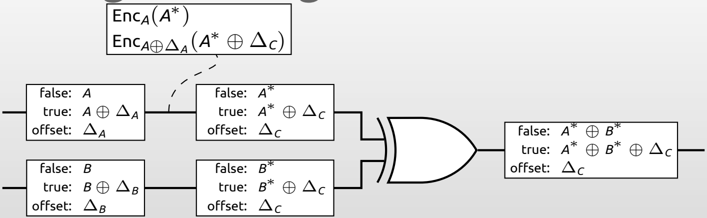
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR

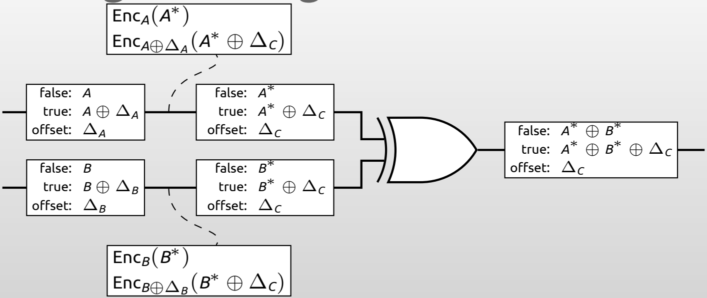
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR

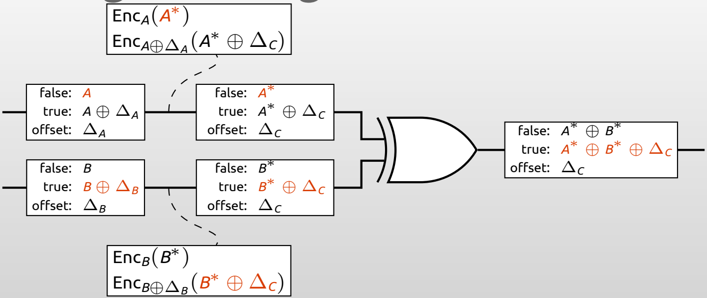
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR

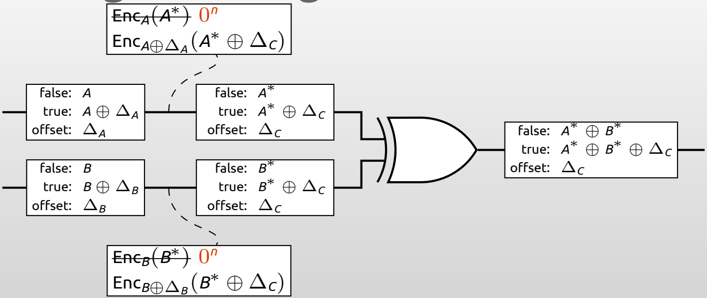
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR

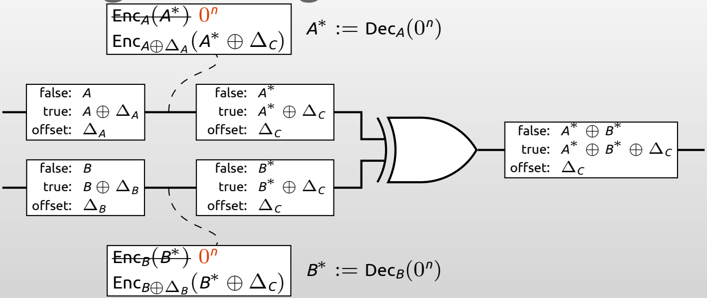
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- “adjust” offsets of both input wires to Δ_C, then use free XOR
- apply row reduction

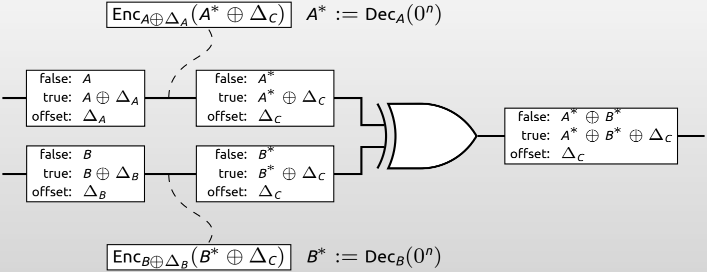
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR
- apply row reduction

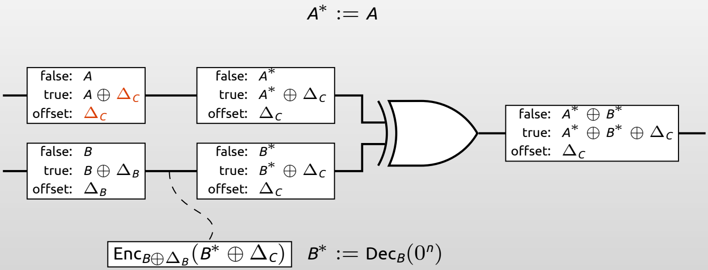
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR
- apply row reduction: each "adjustment" requires 1 ciphertext

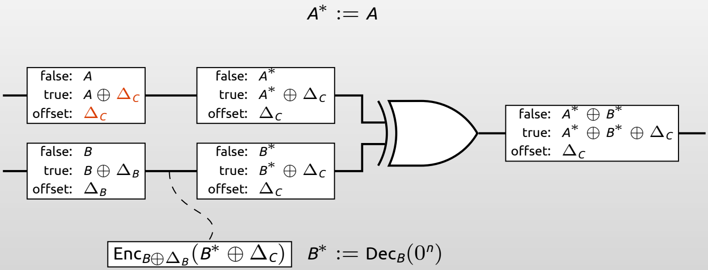
fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

- "adjust" offsets of both input wires to Δ_C, then use free XOR
- apply row reduction: each "adjustment" requires 1 ciphertext
- if $\Delta_A = \Delta_C$, no need to "adjust" first wire at all!

fleXOR garbling



Flexible XOR (fleXOR) technique [this work]:

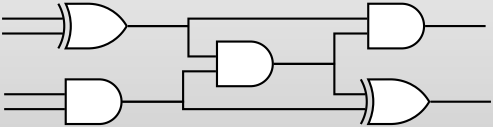
- "adjust" offsets of both input wires to Δ_C, then use free XOR
- apply row reduction: each "adjustment" requires 1 ciphertext
- if $\Delta_A = \Delta_C$, no need to "adjust" first wire at all!
- **garble XOR gate using 0, 1, or 2 ciphertexts**

  $\cdots$ depending on how many of $\{\Delta_A, \Delta_B, \Delta_C\}$ are distinct

wire orderings

Wire ordering:

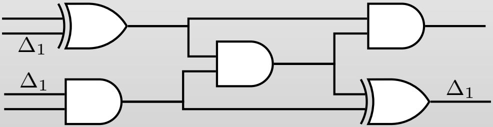
Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



wire orderings

Wire ordering:

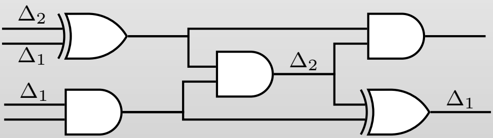
Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



wire orderings

Wire ordering:

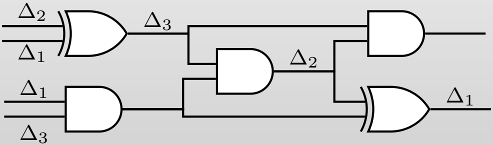
Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



wire orderings

Wire ordering:

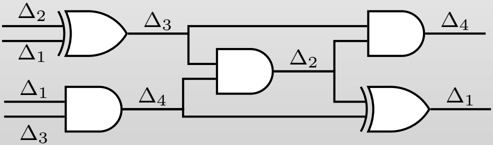
Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



wire orderings

Wire ordering:

Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)

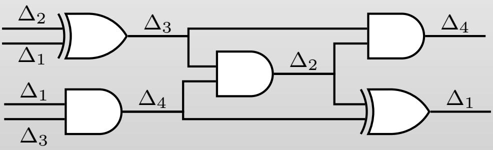


How should we choose wire orderings to minimize total cost of garbling XOR gates?

wire orderings

Wire ordering:

Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



How should we choose wire orderings to minimize total cost of garbling XOR gates?

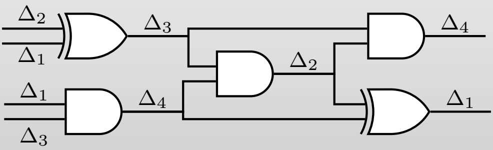
⋯ while avoiding circularity assumption of Free-XOR?

⋯ keeping compatibility with 2-row-reduction for non-XOR gates?

wire orderings

Wire ordering:

Group circuit's **wires** into equivalence classes (same class $\Leftrightarrow$ same offset)



How should we choose wire orderings to minimize total cost of garbling XOR gates?

$\cdots$ while avoiding circularity assumption of Free-XOR?

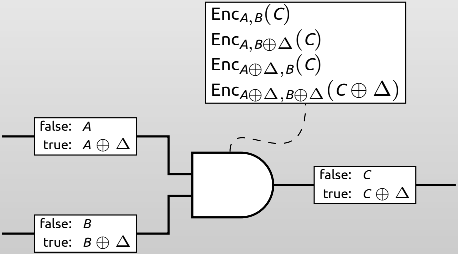
$\cdots$ keeping compatibility with 2-row-reduction for non-XOR gates?

combinatorial constraints of wire ordering

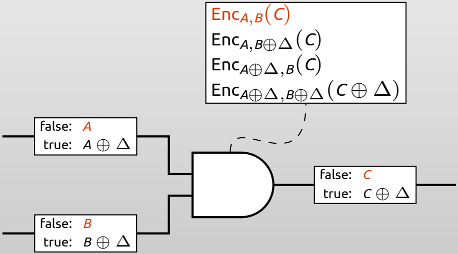
removing circularity

3

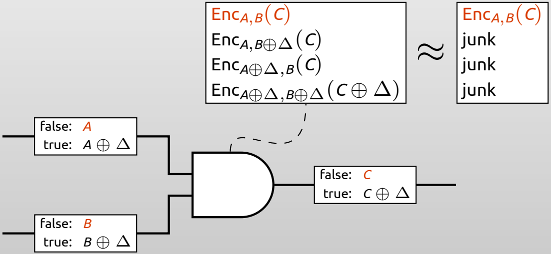
why does free-XOR require circularity assumption?



why does free-XOR require circularity assumption?

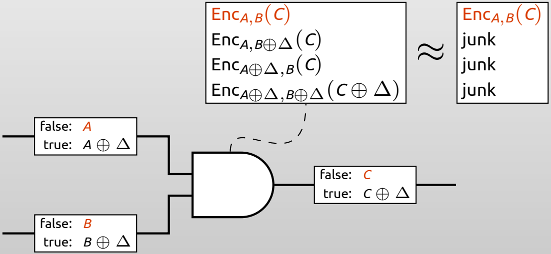


why does free-XOR require circularity assumption?



$$\mathsf{Enc}_{A\oplus\Delta,B\oplus\Delta}(C\oplus\Delta) \approx \text{junk},$$

why does free-XOR require circularity assumption?



$\mathsf{Enc}_{A\oplus\Delta,B\oplus\Delta}(C\oplus\Delta)\approx$ junk,

- **Key cycle:** same secret Δ in key and message!

main idea: removing circularity

Recipe: how to avoid a "key cycle"

1. Order all the wire-offsets: $\Delta_1, \Delta_2, \ldots$
2. Enforce: $\mathsf{Enc}_{\cdots A \oplus \Delta_i \cdots}(\cdots B \oplus \Delta_j \cdots)$ allowed $\Leftrightarrow i < j$

main idea: removing circularity

Recipe: how to avoid a “key cycle”

1. Order all the wire-offsets: $\Delta_1, \Delta_2, \ldots$
2. Enforce: $\mathsf{Enc}_{\ldots A \oplus \Delta_i \ldots}(\cdots B \oplus \Delta_j \cdots)$ allowed $\Leftrightarrow i < j$

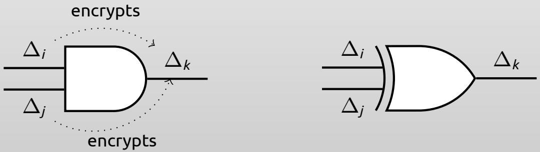
In FleXOR:

main idea: removing circularity

Recipe: how to avoid a "key cycle"

1. Order all the wire-offsets: $\Delta_1, \Delta_2, \ldots$
2. Enforce: $\mathrm{Enc}_{\ldots A \oplus \Delta_i \ldots}(\cdots B \oplus \Delta_j \cdots)$ allowed $\Leftrightarrow i < j$

In FleXOR:

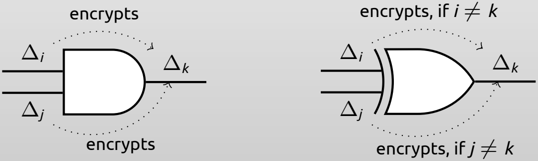


main idea: removing circularity

Recipe: how to avoid a "key cycle"

1. Order all the wire-offsets: $\Delta_1, \Delta_2, \ldots$
2. Enforce: $\text{Enc}_{\ldots A \oplus \Delta_i \ldots}(\cdots B \oplus \Delta_j \cdots)$ allowed $\Leftrightarrow i < j$

In FleXOR:

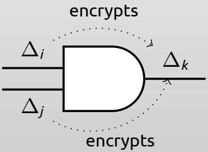


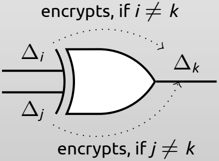
main idea: removing circularity

Recipe: how to avoid a "key cycle"

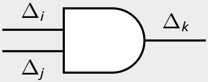
1. Order all the wire-offsets: $\Delta_1, \Delta_2, \ldots$
2. Enforce: $\mathsf{Enc}_{\ldots A \oplus \Delta_i \ldots}(\cdots B \oplus \Delta_j \cdots)$ allowed $\Leftrightarrow i < j$

In FleXOR:

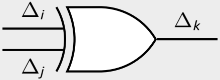




Definition: monotone wire ordering



$\Rightarrow k > \max\{i, j\}$



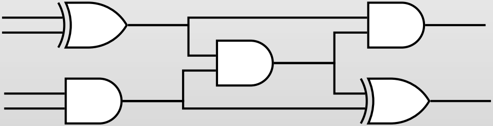
$\Rightarrow k \geq \max\{i, j\}$

results

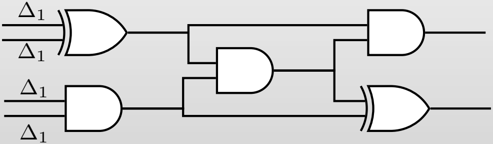
Results

1. FleXOR garbling does not require circularity assumption, when offsets chosen via **monotone** wire ordering
   - Same assumption required for OT-extension [Ishai+03]
2. NP-hard to find **optimal** monotone wire ordering

finding some monotone ordering

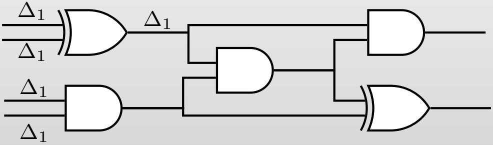


finding some monotone ordering



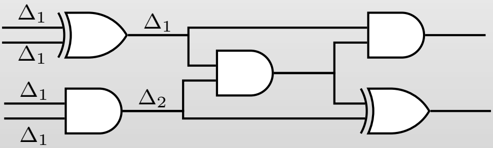
1. Input wires get Δ_1

finding some monotone ordering



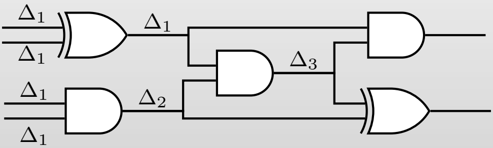
1. Input wires get Δ_1
2. For each gate in topological order, assign smallest legal Δ_i
    - XOR gates: $k \geq \max\{i, j\}$

finding some monotone ordering



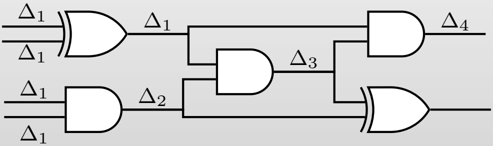
1. Input wires get Δ_1
2. For each gate in topological order, assign smallest legal Δ_i
   - XOR gates: $k \geq \max\{i, j\}$
   - other gates: $k > \max\{i, j\}$

finding some monotone ordering



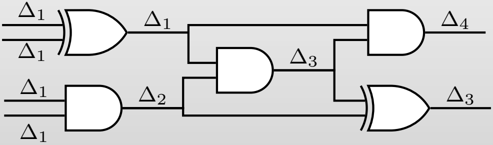
1. Input wires get Δ_1
2. For each gate in topological order, assign smallest legal Δ_i
   - XOR gates: $k \geq \max\{i, j\}$
   - other gates: $k > \max\{i, j\}$

finding some monotone ordering



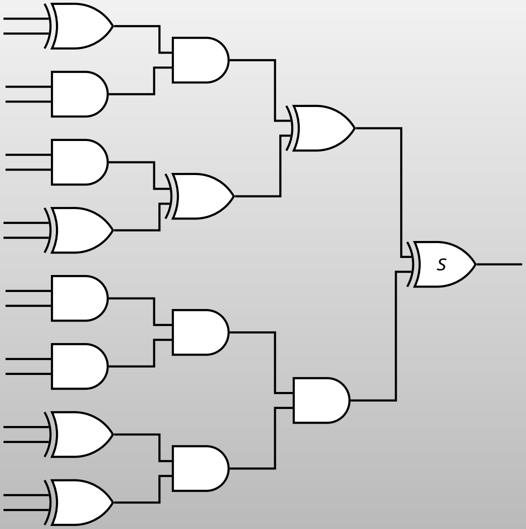
1. Input wires get Δ_1
2. For each gate in topological order, assign smallest legal Δ_i
   - XOR gates: $k \geq \max\{i, j\}$
   - other gates: $k > \max\{i, j\}$

finding some monotone ordering



1. Input wires get Δ_1
2. For each gate in topological order, assign smallest legal Δ_i
   - XOR gates: $k \geq \max\{i, j\}$
   - other gates: $k > \max\{i, j\}$

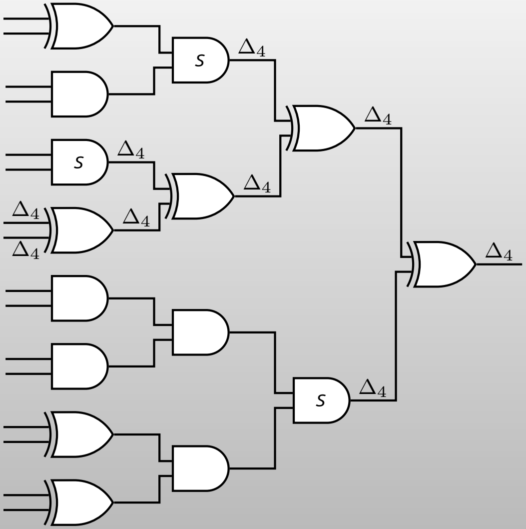
optimal solution for formulas



$S :=$ {output gate}
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S :=$ {barrier AND-gates}
3. $i := i - 1$

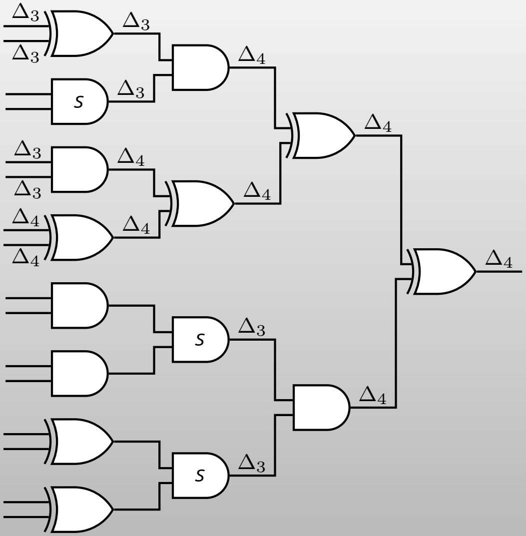
optimal solution for formulas



$S := \{\text{output gate}\}$
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S := \{\text{barrier AND-gates}\}$
3. $i := i - 1$

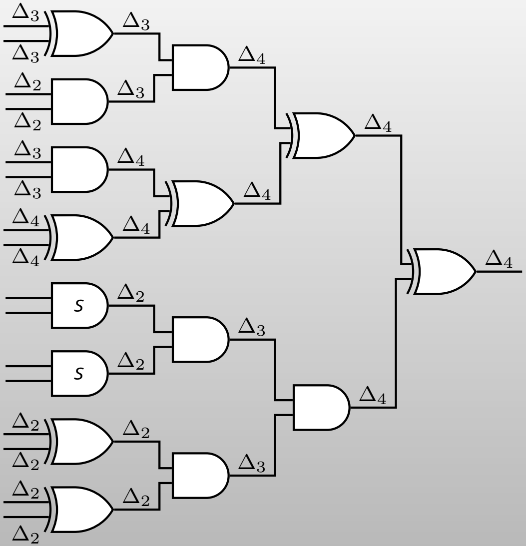
optimal solution for formulas



$S := \{$output gate$\}$
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S := \{$barrier AND-gates$\}$
3. $i := i - 1$

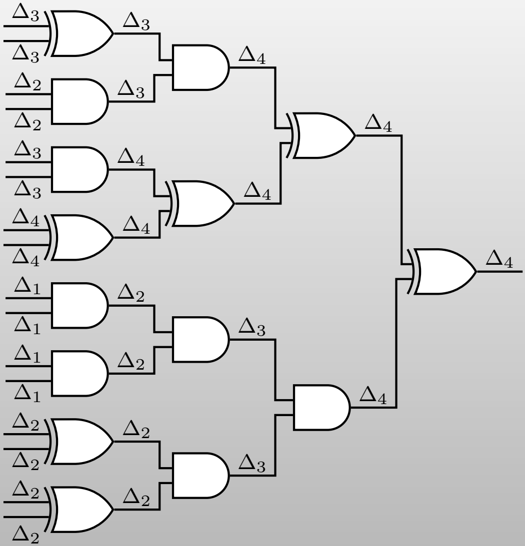
optimal solution for formulas



$S := \{\text{output gate}\}$
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S := \{\text{barrier AND-gates}\}$
3. $i := i - 1$

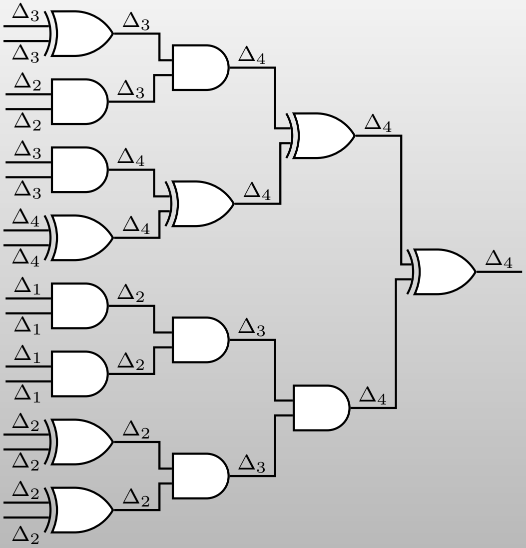
optimal solution for formulas



$S := \{\text{output gate}\}$
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S := \{\text{barrier AND-gates}\}$
3. $i := i - 1$

optimal solution for formulas



$S :=$ {output gate}
$i :=$ *depth*
repeat:

1. assign i to wires that reach S via XOR paths
2. $S :=$ {barrier AND-gates}
3. $i := i - 1$

All XOR gates are free!

heuristic for general circuits

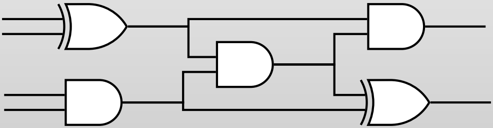
Observation

[offset given to wire w] + [# AND gates between w and output] = constant

heuristic for general circuits

Observation

[offset given to wire w] + [# AND gates between w and output] = constant



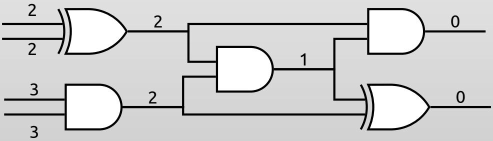
Smarter heuristic:

1. $d(w) =$ max of # AND gates in a path from w to output

heuristic for general circuits

Observation

[offset given to wire w] + [# AND gates between w and output] = constant



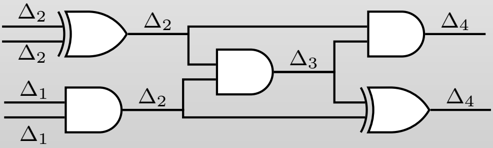
Smarter heuristic:

1. $d(w) =$ max of # AND gates in a path from w to output

heuristic for general circuits

Observation

[offset given to wire w] + [# AND gates between w and output] = constant



Smarter heuristic:

1. $d(w) =$ max of # AND gates in a path from w to output
2. assign Δ_i to w so that $i + d(w) =$ constant

concrete results (using our heuristic)

Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	**2.00**	2.00	**2.00**	2.00	2.00
Free XOR	circular	**0.64**	2.79	**1.82**	2.05	**0.50**	**0.90**

concrete results (using our heuristic)

Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	**2.00**	2.00	**2.00**	2.00	2.00
Free XOR	circular	**0.64**	2.79	**1.82**	2.05	**0.50**	**0.90**
FleXOR	related-key	0.76	2.84	2.02	2.26	0.67	1.15

concrete results (using our heuristic)

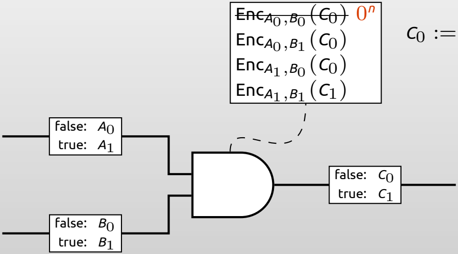
Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	**2.00**	2.00	**2.00**	2.00	2.00
Free XOR	circular	**0.64**	2.79	**1.82**	2.05	**0.50**	**0.90**
FleXOR	related-key	0.76	2.84	2.02	2.26	0.67	1.15
		+19%	+2%	+11%	+10%	+34%	+28%

row-reduction compatibility

4

why is free-XOR incompatible with 4→2-row-reduction?

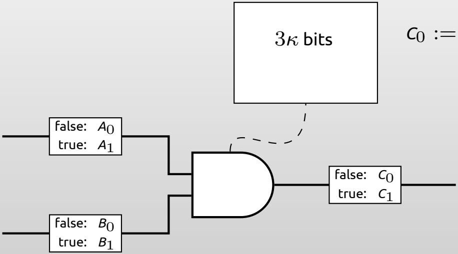


$C_0 := \mathsf{Dec}_{A_0,B_0}(0^n)$

Row reductions

- $4 \to 3$ reduction

why is free-XOR incompatible with 4→2-row-reduction?

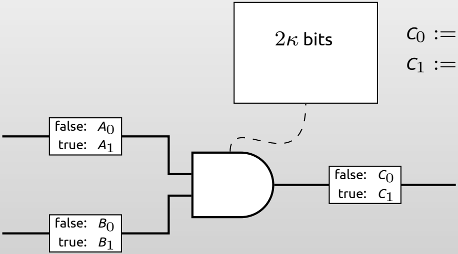


$C_0 := F(A_0, A_1, B_0, B_1)$

Row reductions

- $4 \to 3$ reduction: C_0 set implicitly

why is free-XOR incompatible with 4→2-row-reduction?

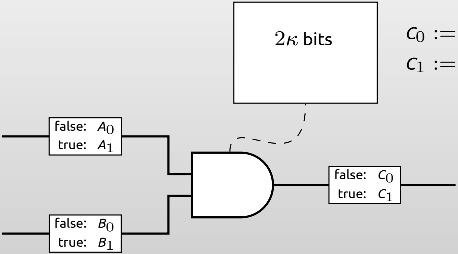


$$C_0 := F(A_0, A_1, B_0, B_1)$$
$$C_1 := F'(A_0, A_1, B_0, B_1)$$

Row reductions

- $4 \to 3$ reduction: C_0 set implicitly
- $4 \to 2$ reduction: both C_0, C_1 set implicitly

why is free-XOR incompatible with 4→2-row-reduction?



$$C_0 := F(A_0, A_1, B_0, B_1)$$
$$C_1 := F'(A_0, A_1, B_0, B_1)$$

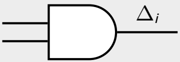
Row reductions

- $4 \to 3$ reduction: C_0 set implicitly
- $4 \to 2$ reduction: both C_0, C_1 set implicitly

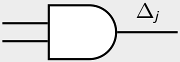
$\Rightarrow$ no control over offset of output wire!

compatibility with fleXOR

Definition: safe wire ordering



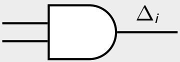
&



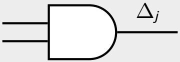
$\Rightarrow \quad i \neq j$

compatibility with fleXOR

Definition: safe wire ordering



&

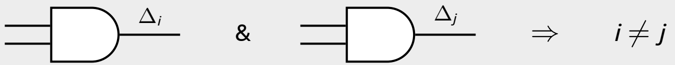


$\Rightarrow \quad i \neq j$

... plus some fine print

compatibility with fleXOR

Definition: safe wire ordering



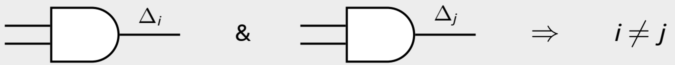
... plus some fine print

Results

1. Can garble using FleXOR + $4 \to 2$ row reduction, when offsets chosen via **safe** wire ordering
   - XOR gates cost 0, 1, or 2; other gates cost **2**
2. We suggest a very simple heuristic to find safe orderings:

compatibility with fleXOR

Definition: safe wire ordering



... plus some fine print

Results

1. Can garble using FleXOR + $4 \to 2$ row reduction, when offsets chosen via **safe** wire ordering
   - XOR gates cost 0, 1, or 2; other gates cost **2**
2. We suggest a very simple heuristic to find safe orderings:
   - Output wires of AND-gates get distinct offsets (in topological order)
   - All other wires get offset Δ_0

concrete results (using our heuristic)

Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	**2.00**	2.00	**2.00**	2.00	2.00
Free XOR	circular	**0.64**	2.79	**1.82**	2.05	**0.50**	**0.90**
FleXOR	related-key	0.76	2.84	2.02	2.26	0.67	1.15

concrete results (using our heuristic)

Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	2.00	2.00	2.00	2.00	2.00
Free XOR	circular	**0.64**	2.79	1.82	2.05	**0.50**	**0.90**
FleXOR	related-key	0.76	2.84	2.02	2.26	0.67	1.15
FleXOR	circular	0.72	**1.89**	**1.39**	**1.56**	**0.50**	0.94

concrete results (using our heuristic)

Garbled circuit size (ciphertexts per gate)

scheme	assump	AES	DES	SHA1	SHA2	HamDst	IntMult
classical	**OWF**	2.00	2.00	2.00	2.00	2.00	2.00
Free XOR	circular	**0.64**	2.79	1.82	2.05	**0.50**	**0.90**
FleXOR	related-key	0.76	2.84	2.02	2.26	0.67	1.15
FleXOR	circular	0.72	**1.89**	**1.39**	**1.56**	**0.50**	0.94
		+12%	-32%	-24%	-24%	+0%	+4%

extensions

5

extensions: wire orderings

1. Trivial wire ordering $\Rightarrow$ collapse to Free-XOR
2. Monotone wire ordering $\Rightarrow$ eliminate circular assumption
3. Safe wire ordering $\Rightarrow$ compatibility with aggressive row reduction

extensions: wire orderings

1. Trivial wire ordering $\Rightarrow$ collapse to Free-XOR
2. Monotone wire ordering $\Rightarrow$ eliminate circular assumption
3. Safe wire ordering $\Rightarrow$ compatibility with aggressive row reduction
4. Monotone + safe wire ordering $\Rightarrow$ both!

extensions: wire orderings

1. Trivial wire ordering $\Rightarrow$ collapse to Free-XOR
2. Monotone wire ordering $\Rightarrow$ eliminate circular assumption
3. Safe wire ordering $\Rightarrow$ compatibility with aggressive row reduction
4. Monotone + safe wire ordering $\Rightarrow$ both!
5. Constrain input/output wires only $\Rightarrow$ compatibility with protocols that break “garbling scheme” abstraction boundary

extensions: wire orderings

1. Trivial wire ordering $\Rightarrow$ collapse to Free-XOR
2. Monotone wire ordering $\Rightarrow$ eliminate circular assumption
3. Safe wire ordering $\Rightarrow$ compatibility with aggressive row reduction
4. Monotone + safe wire ordering $\Rightarrow$ both!
5. Constrain input/output wires only $\Rightarrow$ compatibility with protocols that break "garbling scheme" abstraction boundary
6. Other interesting properties?

wrap-up

6

summary

FleXOR = Flexible XOR!

- New way to garble XOR gates: costs 0, 1, or 2 ciphertexts per gate
- Get results competitive with Free-XOR, from weaker assumption
- Get results often better than Free-XOR, by leveraging $4 \to 2$ row-reduction

open problems

1. Better FleXOR on existing circuits
   - better wire-ordering heuristics? guaranteed approximation ratio?
   - hardness of approximation?
   - use ideas from approximations of “multi-cut” problem

open problems

1. Better FleXOR on existing circuits
   - better wire-ordering heuristics? guaranteed approximation ratio?
   - hardness of approximation?
   - use ideas from approximations of "multi-cut" problem

2. Better circuits targeted for FleXOR
   - instead of simply minimizing # of non-XOR gates

open problems

1. Better FleXOR on existing circuits
   - better wire-ordering heuristics? guaranteed approximation ratio?
   - hardness of approximation?
   - use ideas from approximations of “multi-cut” problem

2. Better circuits targeted for FleXOR
   - instead of simply minimizing # of non-XOR gates

3. Implementation
   - fastest garbling scheme (JustGarble) uses fixed-key AES: need to re-analyze FleXOR security
   - wire orderings computed on the fly, or stored with circuit?
   - revisit $4 \to 2$ row reduction?


THANK YOU