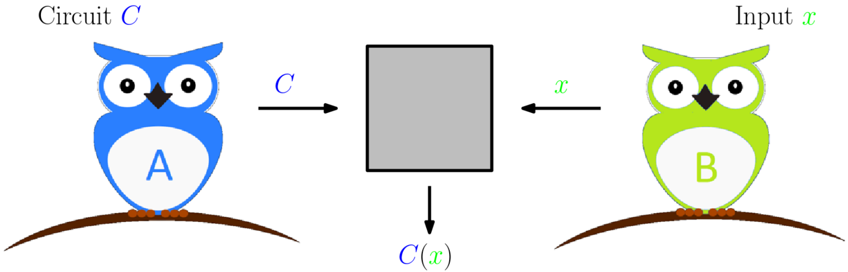
Limits on the Adaptive Security of Yao's Garbling

Chethan Kamath, **Karen Klein**[1], Krzysztof Pietrzak[1], Daniel Wichs[2]

1 - IST Austria

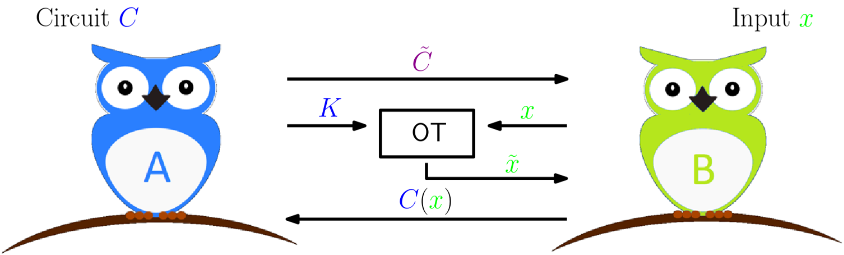
2 - Northeastern University, NTT Research

Background



Background

Yao's solution [Yao86]:



$(\tilde{C}, K) \leftarrow \mathsf{Garble}(C)$

$\mathsf{Eval}(\tilde{C}, \tilde{x}) = C(x)$

Security of Yao's Garbling

LP09: **selective** security proof (input known ahead of time)

$\Rightarrow$ adaptive security via *randomly guessing* the **input of length** n:

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } 2^n \cdot \varepsilon\text{-secure}$$

Security of Yao's Garbling

LP09: **selective** security proof (input known ahead of time)

$\Rightarrow$ adaptive security via *randomly guessing* the **input of length** n:

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \Rightarrow \text{Yao's scheme } 2^n \cdot \varepsilon\text{-secure}$$

JW16: **adaptive** security proof for circuits of **depth** D:

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \Rightarrow \text{Yao's scheme } 2^D \cdot \varepsilon\text{-secure}$$

Security of Yao's Garbling

LP09: **selective** security proof (input known ahead of time)

$\Rightarrow$ adaptive security via *randomly guessing* the **input of length** n:

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } 2^n \cdot \varepsilon\text{-secure}$$

JW16: **adaptive** security proof for circuits of **depth** D:

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } 2^D \cdot \varepsilon\text{-secure}$$

Theorem (Our work)

Any **black-box proof** *of* **adaptive indistinguishability** *for Yao's garbling scheme for circuits with n-bit input, 1-bit output, and depth* $D \leq 2n$ **from** *an* **IND-CPA secure SKE** *incurs a security loss of* $2^{\Omega(\sqrt{D})}$.

Discussion of our Results

Our results

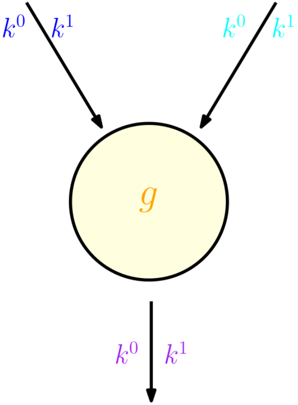
- **only** apply to **Yao's construction**, we do **not** prove a **separation of garbled circuits from one-way functions**
  - HJO+16: adaptively secure garbling from one-way functions using "somewhere equivocal" encryption
    (online complexity increases with the *pebble complexity* of the circuit)

Discussion of our Results

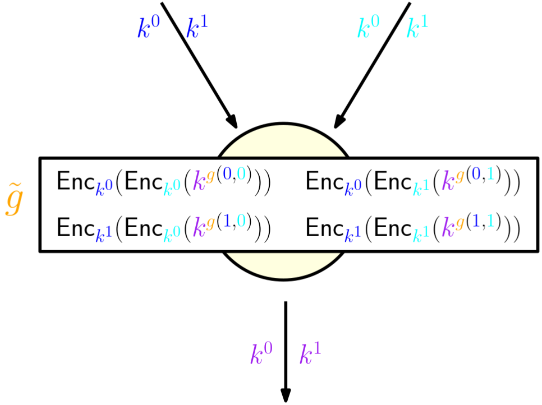
Our results

- **only** apply to **Yao's construction**, we do **not** prove a **separation of garbled circuits from one-way functions**
  - HJO+16: adaptively secure garbling from one-way functions using "somewhere equivocal" encryption
    (online complexity increases with the *pebble complexity* of the circuit)
- hold even for **indistinguishability** (a weaker security notion than simulatability) and a **variant of Yao** (JW16) where the output map is sent *online*
  - AIKW13: Yao's original scheme is not adaptively simulatable (for circuits with *large* output)

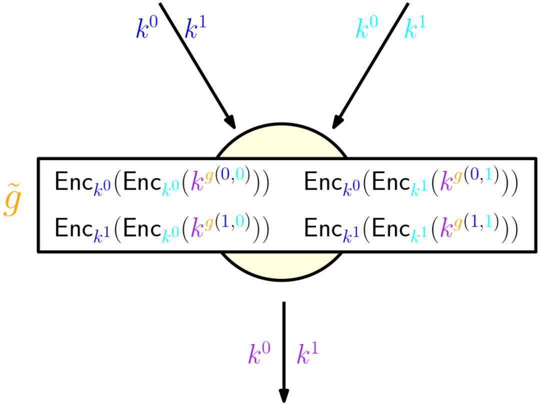
Yao's Garbling



Yao's Garbling

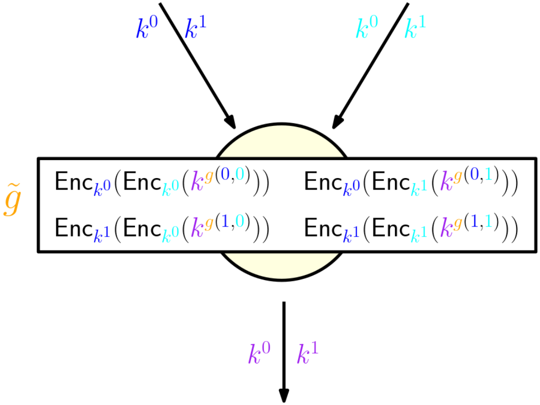


Yao's Garbling



$\tilde{C} = \{\tilde{g}\}_{g \in C}, \; K = \{k^0, k^1, k^0, k^1, \ldots\}$ can be computed offline

Yao's Garbling

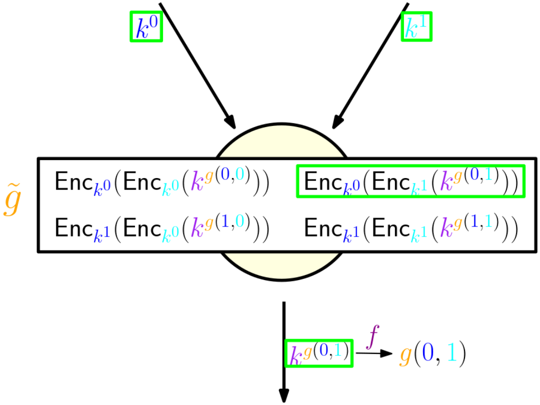


$\tilde{C} = \{\tilde{g}\}_{g \in C}$, $K = \{k^0, k^1, k^0, k^1, \ldots\}$ can be computed offline

For $x = (x_1, x_2, \ldots)$: $\tilde{x} = (k^{x_1}, k^{x_2}, \ldots)$

Output mapping: $f = \{k^0 \to 0, k^1 \to 1, \ldots\}$

Yao's Garbling



$\tilde{C} = \{\tilde{g}\}_{g \in C}$, $K = \{k^0, k^1, k^0, k^1, \ldots\}$ can be computed offline

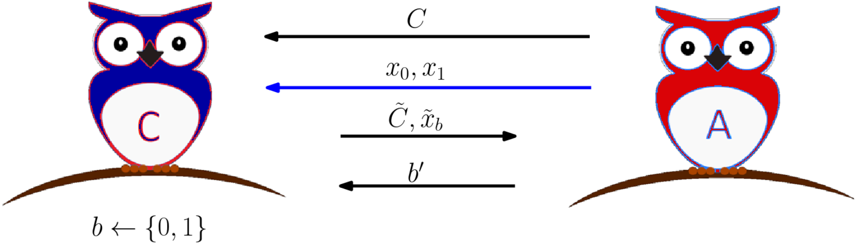
For $x = (x_1, x_2, \ldots)$: $\tilde{x} = (k^{x_1}, k^{x_2}, \ldots)$

Output mapping: $f = \{k^0 \to 0, k^1 \to 1, \ldots\}$

Security Definition for Garbling

selective indistinguishability

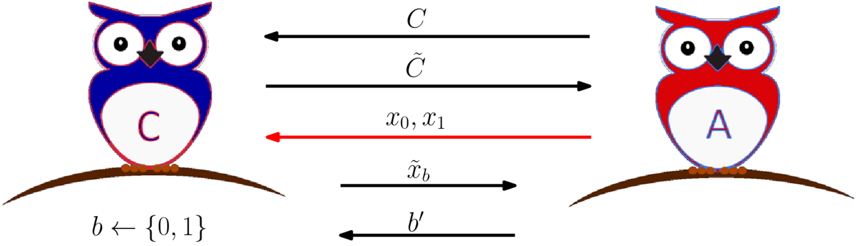
(weaker than simulation-based security)



Security Definition for Garbling

adaptive indistinguishability

(weaker than simulation-based security)



Proof Idea

Theorem (Our work)

Any **black-box proof** *of* **adaptive indistinguishability** *for Yao's garbling scheme for circuits with n-bit input, 1-bit output, and depth* $D \leq 2n$ **from** *an* **IND-CPA secure SKE** *incurs a security loss of* $2^{\Omega(\sqrt{D})}$.

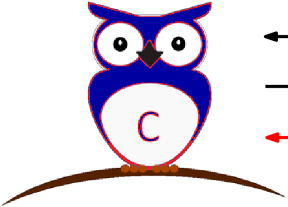
Define oracles $\mathcal{F}$ and $\mathcal{A}$ such that

- $\mathcal{F} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is an **ideal SKE scheme**
- $\mathcal{A}$ is an (inefficient) **adversary** breaking Yao's scheme, but "not too helpful" in breaking $\mathcal{F}$.

Proof Idea: The Adversary $\mathcal{A}$

adaptive indistinguishability

(weaker than simulation-based security)



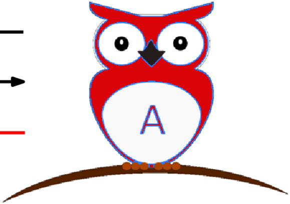
$\xleftarrow{C : \{0,1\}^n \to 0}$

$\xrightarrow{\tilde{C}}$

$\xleftarrow{x_0, x_1 \text{ uniformly random}}$

$\xrightarrow{\tilde{x}_b}$

$\xleftarrow{b'}$



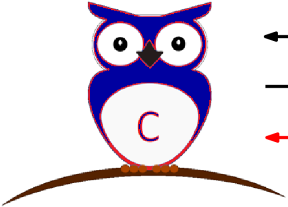
$b \leftarrow \{0,1\}$

$b' = 0 \text{ iff } (\tilde{C}, \tilde{x}_b)$ *good* w.r.t. x_0

Proof Idea: The Adversary $\mathcal{A}$

adaptive indistinguishability

(weaker than simulation-based security)



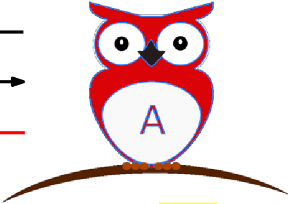
$\boxed{C}: \{0,1\}^n \to 0$ ←

$\tilde{C}$ →

x_0, x_1 uniformly random ←

$\tilde{x}_b$ →

b' ←



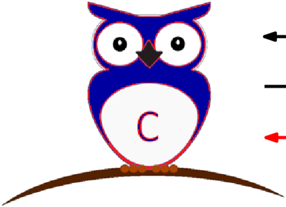
$b \leftarrow \{0,1\}$

$b' = 0$ iff $(\tilde{C}, \tilde{x}_b)$ $\boxed{good}$ w.r.t. x_0

Proof Idea: The Adversary $\mathcal{A}$

adaptive indistinguishability

(weaker than simulation-based security)



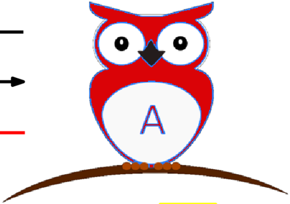
$\boxed{C}: \{0,1\}^n \to 0$ ←

$\tilde{C}$ →

x_0, x_1 uniformly random ←

$\tilde{x}_b$ →

b' ←



$b \leftarrow \{0,1\}$

$b' = 0$ iff $(\tilde{C}, \tilde{x}_b)$ $\boxed{good}$ w.r.t. x_0

defined through some **pebble game** on graphs, guarantees that $\mathcal{A}$ succeeds

Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.

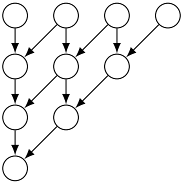
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



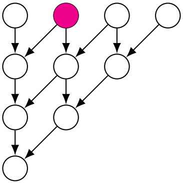
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



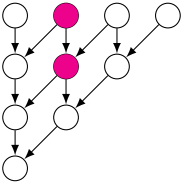
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



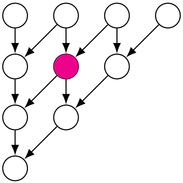
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



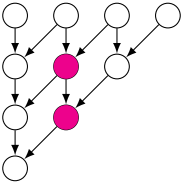
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



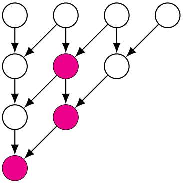
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



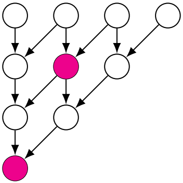
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



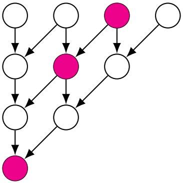
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



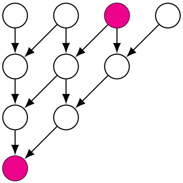
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



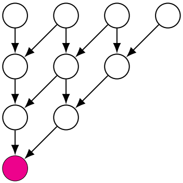
Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.



Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.

$\mathcal{P}$ is *good* if it is reachable with less than d pebbles (where $d = \Theta(D)$).

Proof Idea: The *good* Predicate

Given $(\tilde{C}, \tilde{x}_b)$, $\mathcal{A}$ extracts a *pebble configuration* $\mathcal{P}$ on C:

- Check (via brute-force) each garbling table in $\tilde{C}$, if incorrect (w.r.t. $\tilde{x}_b, x_0$) assign a pebble.

Consider the following **pebble game**:

- In each step can place/remove a pebble on a node, if at least one of its parents carries a pebble.

$\mathcal{P}$ is *good* if it is reachable with less than d pebbles (where $d = \Theta(D)$).

Lemma ($\mathcal{A}$ breaks Yao's scheme)

For appropriately chosen circuit C with high pebble complexity:
$\emptyset = \mathcal{P}_0 \leftarrow \mathcal{A}(\tilde{C}, \tilde{x}_0)$ *good and* $\mathcal{P}_1 \leftarrow \mathcal{A}(\tilde{C}, \tilde{x}_1)$ *bad.*

Proof Idea: $\mathcal{A}$ is "not too useful"

$\mathcal{A}[c^*]$: **punctured adversary**, IND-CPA challenge ciphertext c^* hardcoded and never decrypted

$\rightarrow$ not useful for any reduction.

Proof Idea: $\mathcal{A}$ is "not too useful"

$\mathcal{A}[c^*]$: **punctured adversary**, IND-CPA challenge ciphertext c^* hardcoded and never decrypted

$\rightarrow$ not useful for any reduction.

Can only distinguish $\mathcal{A}[c^*]$ from $\mathcal{A}$ if $\mathcal{P} \leftarrow \mathcal{A}$ good and $\mathcal{P}^* \leftarrow \mathcal{A}[c^*]$ bad.

Proof Idea: $\mathcal{A}$ is "not too useful"

$\mathcal{A}[c^*]$: **punctured adversary**, IND-CPA challenge ciphertext c^* hardcoded and never decrypted

$\rightarrow$ not useful for any reduction.

Can only distinguish $\mathcal{A}[c^*]$ from $\mathcal{A}$ if $\mathcal{P} \leftarrow \mathcal{A}$ good and $\mathcal{P}^* \leftarrow \mathcal{A}[c^*]$ bad.

Lemma

$\mathcal{P}$ and $\mathcal{P}^*$ differ in at most one pebbling step.

$\rightarrow$ $\mathcal{P}$ contains $d-1$ pebbles (by definition of *good*)

Proof Idea: $\mathcal{A}$ is "not too useful"

$\mathcal{A}[c^*]$: **punctured adversary**, IND-CPA challenge ciphertext c^* hardcoded and never decrypted

$\rightarrow$ not useful for any reduction.

Can only distinguish $\mathcal{A}[c^*]$ from $\mathcal{A}$ if $\mathcal{P} \leftarrow \mathcal{A}$ good and $\mathcal{P}^* \leftarrow \mathcal{A}[c^*]$ bad.

Lemma

$\mathcal{P}$ and $\mathcal{P}^*$ differ in at most one pebbling step.

$\rightarrow$ $\mathcal{P}$ contains $d-1$ pebbles (by definition of *good*)

Lemma (Unlikely to reach a threshold configuration)

For any $\tilde{C}$ the probability (over uniformly random x_0) that there exists $\tilde{x}_b$ such that $\mathcal{P}$ good and $\mathcal{P}^*$ bad is small.

Proof Idea: Establishing the Lemma

Lemma (Unlikely to reach a threshold configuration)

For any $\tilde{C}$ the probability (over uniformly random x_0) that there exists $\tilde{x}_b$ such that $\mathcal{P}$ good and $\mathcal{P}^*$ bad is small.

Proof Idea: Establishing the Lemma

Lemma (Unlikely to reach a threshold configuration)

For any $\tilde{C}$ the probability (over uniformly random x_0) that there exists $\tilde{x}_b$ such that $\mathcal{P}$ good and $\mathcal{P}^*$ bad is small.

- $\mathcal{P}$ contains many pebbles.
- The reduction needs to correctly guess the output of pebbled gates during evaluation $C(x_0)$.

Proof Idea: Establishing the Lemma

Lemma (Unlikely to reach a threshold configuration)

For any $\tilde{C}$ the probability (over uniformly random x_0) that there exists $\tilde{x}_b$ such that $\mathcal{P}$ good and $\mathcal{P}^*$ bad is small.

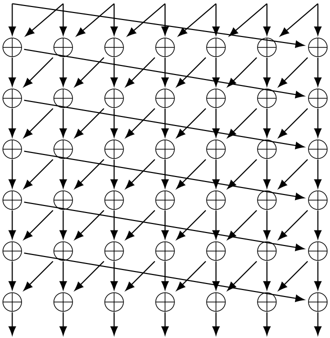
- $\mathcal{P}$ contains many pebbles.
- The reduction needs to correctly guess the output of pebbled gates during evaluation $C(x_0)$.

To guarantee these properties, define C such that

- C has high pebbling complexity $d = \Theta(D)$,
- contains a block of XOR gates, which maintains high entropy, pebbles on this block correspond to guessing x_0,
- contains subsequent AND gates as "control" mechanism, pebbles on these gates mean that some guess was incorrect.

Proof Idea: The Circuit C

$C^{\oplus}\ldots$ **tower graph** of depth d

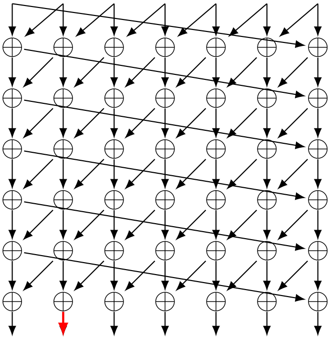


Proof Idea: The Circuit C

$C^{\oplus}$ . . . **tower graph** of depth d

Implement gates as XOR

$\Rightarrow C^{\oplus}(x_0) \neq C^{\oplus}(x_1)$



Proof Idea: The Circuit C

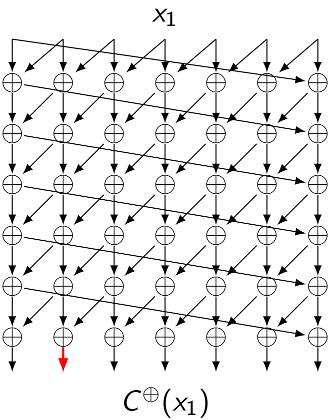
$C^{\oplus}$ . . . **tower graph** of depth d

Implement gates as XOR

$\Rightarrow C^{\oplus}(x_0) \neq C^{\oplus}(x_1)$

AND gates are *asymmetric* w.r.t. input

$\rightarrow$ use them as **control gates**:



Proof Idea: The Circuit C

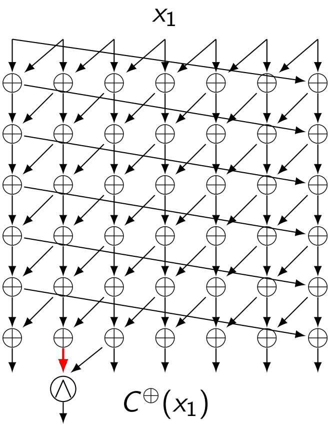
$C^{\oplus}$... **tower graph** of depth d

Implement gates as XOR

$\Rightarrow C^{\oplus}(x_0) \neq C^{\oplus}(x_1)$

AND gates are *asymmetric* w.r.t. input

$\rightarrow$ use them as **control gates**:



Proof Idea: The Circuit C

$C^{\oplus}$ . . . **tower graph** of depth d

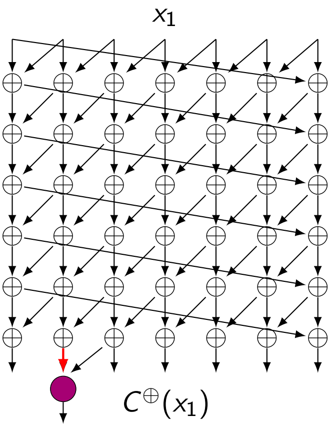
Implement gates as XOR

$\Rightarrow C^{\oplus}(x_0) \neq C^{\oplus}(x_1)$

AND gates are *asymmetric* w.r.t. input

$\rightarrow$ use them as **control gates**:

wrong input $\Rightarrow$ AND pebbled



Proof Idea: The Circuit C

$C^{\oplus}$ . . . **tower graph** of depth d

Implement gates as XOR

$\Rightarrow C^{\oplus}(x_0) \neq C^{\oplus}(x_1)$

AND gates are *asymmetric* w.r.t. input

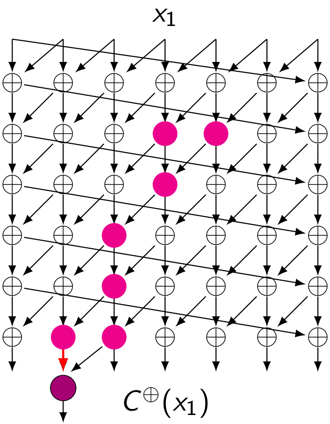
$\rightarrow$ use them as **control gates**:

wrong input $\Rightarrow$ AND pebbled

Pebbling lower bound: Placing a pebble on a gate on layer d requires d pebbles

$\Rightarrow (\tilde{C}, \tilde{x}_1)$ is bad w.r.t. x_0

$\Rightarrow \mathcal{A}$ breaks the garbling scheme

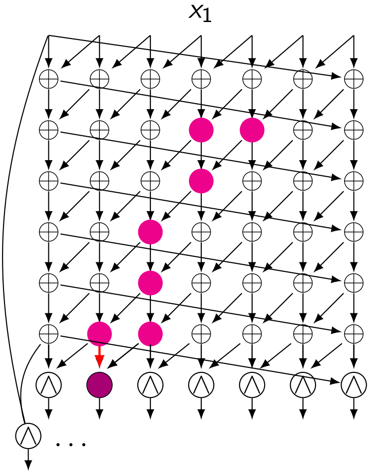


Proof Idea: The Circuit C

AND gate for each input and XOR gate

$\Rightarrow$ whenever a gate evaluates wrong: corresponding AND gate pebbled

$\Rightarrow$ bad configuration

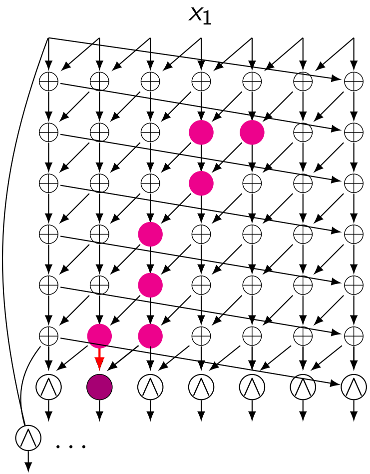


Proof Idea: The Circuit C

AND gate for each input and XOR gate

- $\Rightarrow$ whenever a gate evaluates wrong: corresponding AND gate pebbled
- $\Rightarrow$ bad configuration

Reduction needs to "place" $d-1$ pebbles, and guess output of these gates correctly



Proof Idea: The Circuit C

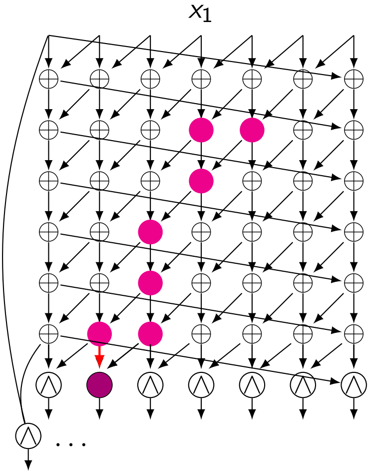
AND gate for each input and XOR gate

- $\Rightarrow$ whenever a gate evaluates wrong: corresponding AND gate pebbled
- $\Rightarrow$ bad configuration

Reduction needs to "place" $d-1$ pebbles, and guess output of these gates correctly

For any subset S of d gates: $\exists S' \subset S$, $|S'| = \sqrt{d}$: output bits of S' independent

- $\Rightarrow$ Reduction succeeds w.p. $\leq 1/2^{\sqrt{d}}$



Proof Idea: The Circuit C

AND gate for each input and XOR gate

$\Rightarrow$ whenever a gate evaluates wrong: corresponding AND gate pebbled

$\Rightarrow$ bad configuration

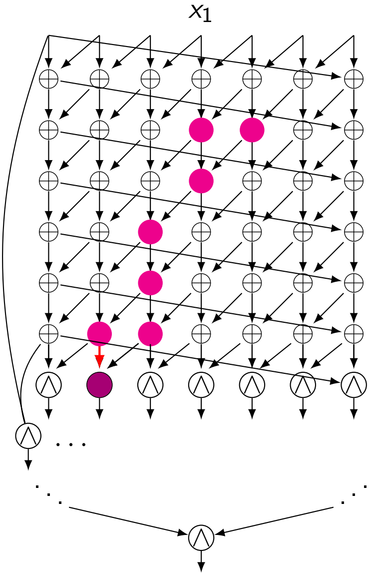
Reduction needs to "place" $d-1$ pebbles, and guess output of these gates correctly

For any subset S of d gates: $\exists S' \subset S$, $|S'| = \sqrt{d}$: output bits of S' independent

$\Rightarrow$ Reduction succeeds w.p. $\leq 1/2^{\sqrt{d}}$

Add binary tree of AND gates

$\Rightarrow$ constant output 0



Conclusion

SKE ε-IND-CPA secure $\Rightarrow$ Yao's scheme ε'-secure

JW16: $\varepsilon'/\varepsilon \leq 2^{O(D)}$

Our work: $\varepsilon'/\varepsilon \geq 2^{\Omega(\sqrt{D})}$ (D ... depth of the circuit)

More details and precise proofs: `https://eprint.iacr.org/2021/945`.

Conclusion

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } \varepsilon'\text{-secure}$$

JW16: $\varepsilon'/\varepsilon \leq 2^{O(D)}$

Our work: $\varepsilon'/\varepsilon \geq 2^{\Omega(\sqrt{D})}$ (D ... depth of the circuit)

More details and precise proofs: `https://eprint.iacr.org/2021/945`.

Open Problems:

- Is it possible to close the gap?

Conclusion

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } \varepsilon'\text{-secure}$$

JW16: $\varepsilon'/\varepsilon \leq 2^{O(D)}$

Our work: $\varepsilon'/\varepsilon \geq 2^{\Omega(\sqrt{D})}$ (D ... depth of the circuit)

More details and precise proofs: `https://eprint.iacr.org/2021/945`.

Open Problems:

- Is it possible to close the gap?
- Can we obtain stronger lower bounds for Yao's original construction, where the output mapping is sent in the *offline* phase?
  (AIKW13: lower bound for simulatability for circuits w. large output, KKP21: upper bound for indistinguishability for small treewidth.)

Conclusion

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } \varepsilon'\text{-secure}$$

JW16: $\varepsilon'/\varepsilon \leq 2^{O(D)}$
Our work: $\varepsilon'/\varepsilon \geq 2^{\Omega(\sqrt{D})}$ (D ... depth of the circuit)

More details and precise proofs: `https://eprint.iacr.org/2021/945`.

Open Problems:

- Is it possible to close the gap?
- Can we obtain stronger lower bounds for Yao's original construction, where the output mapping is sent in the *offline* phase?
  (AIKW13: lower bound for simulatability for circuits w. large output, KKP21: upper bound for indistinguishability for small treewidth.)
- Can we turn this lower bound into a counter example? Under which assumptions?

Conclusion

$$\text{SKE } \varepsilon\text{-IND-CPA secure} \;\Rightarrow\; \text{Yao's scheme } \varepsilon'\text{-secure}$$

JW16: $\varepsilon'/\varepsilon \leq 2^{O(D)}$

Our work: $\varepsilon'/\varepsilon \geq 2^{\Omega(\sqrt{D})}$ (D ... depth of the circuit)

More details and precise proofs: `https://eprint.iacr.org/2021/945`.

Open Problems:

- Is it possible to close the gap?
- Can we obtain stronger lower bounds for Yao's original construction, where the output mapping is sent in the *offline* phase?
  (AIKW13: lower bound for simulatability for circuits w. large output, KKP21: upper bound for indistinguishability for small treewidth.)
- Can we turn this lower bound into a counter example? Under which assumptions?
- Can we use similar ideas for other constructions of garbling or even other cryptographic primitives?