How to Construct Random Unitaries

Fermi Ma

joint work with Hsin-Yuan Huang

Haar measure: unique unitarily invariant measure on $SU(2^n)$

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

learn a state

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

learn a state

generate entanglement

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

learn a state

generate entanglement

complexity theory

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

learn a state	generate entanglement	complexity theory	model black holes	uncloneable crypto

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes
- uncloneable crypto
- complexity growth

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes
- uncloneable crypto
- complexity growth
- quantum Shannon theory

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes
- uncloneable crypto
- complexity growth
- quantum Shannon theory
- randomized benchmarking

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes
- uncloneable crypto
- complexity growth
- quantum Shannon theory
- randomized benchmarking
- commit to a state

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

Haar-random unitaries show up everywhere:

learn a state | generate entanglement | complexity theory | model black holes | uncloneable crypto

complexity growth | quantum Shannon theory | randomized benchmarking | commit to a state | …

Haar measure: unique unitarily invariant measure on $\mathrm{SU}(2^n)$

i.e., for any unitary W, if $U \sim \mathrm{Haar}$, then $W \cdot U \sim \mathrm{Haar}$

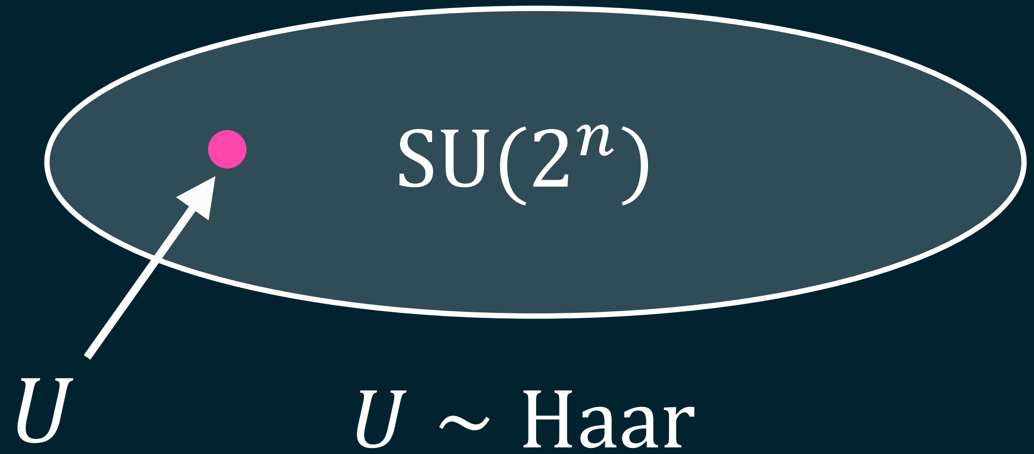
Haar-random unitaries show up everywhere:

- learn a state
- generate entanglement
- complexity theory
- model black holes
- uncloneable crypto
- complexity growth
- quantum Shannon theory
- randomized benchmarking
- commit to a state
- …

Philosophy: *"When good choices abound, guessing randomly can be surprisingly fruitful" (Quanta Magazine)*

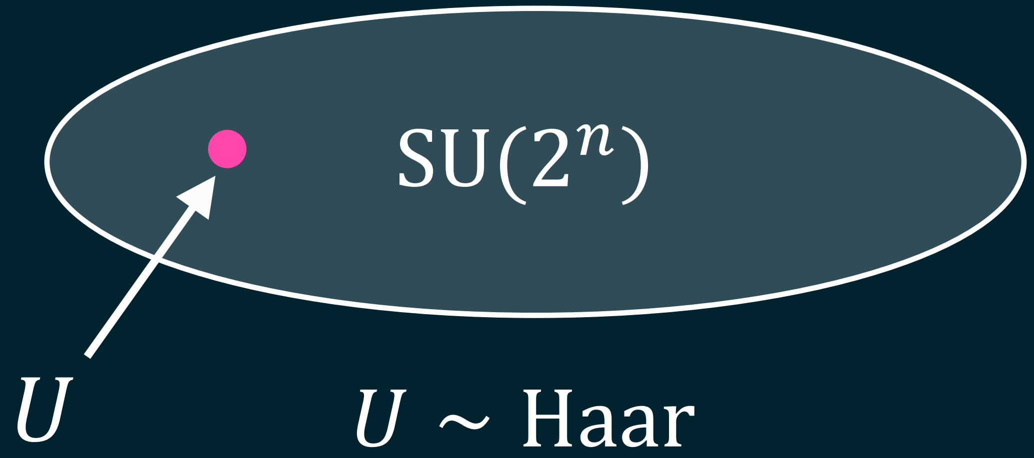
Challenge: Haar-random unitaries are exponentially complex

Challenge: Haar-random unitaries are exponentially complex

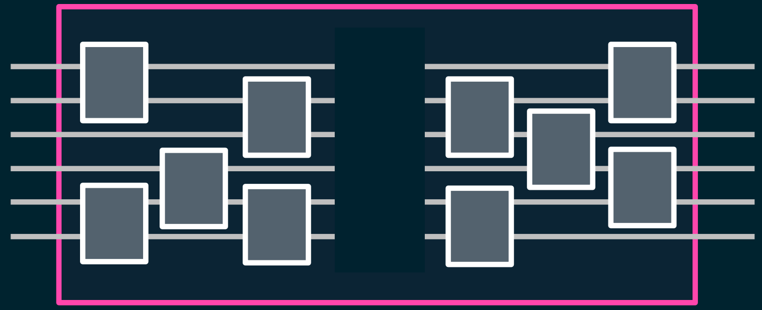


$U \sim \mathrm{Haar}$

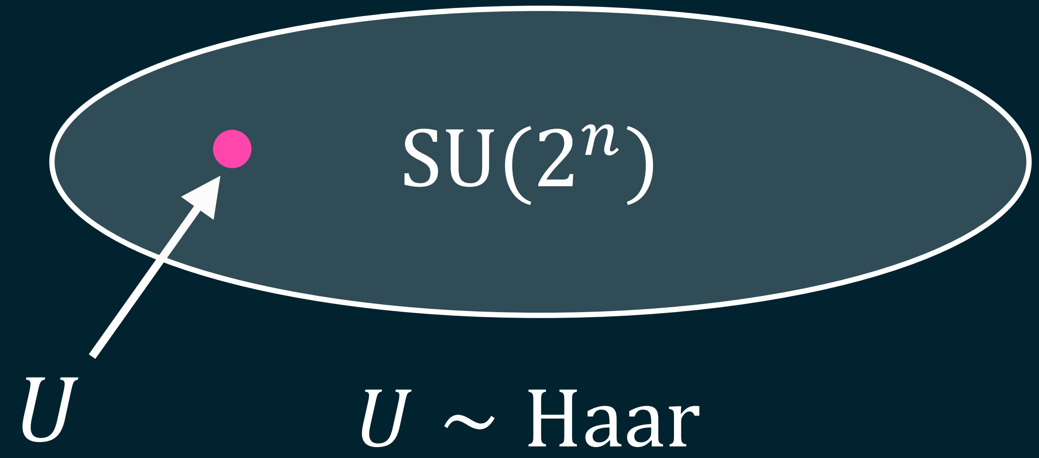
Challenge: Haar-random unitaries are exponentially complex



minimal circuit for U

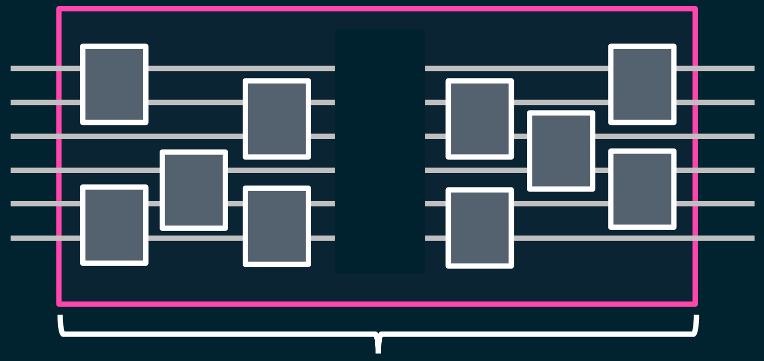


Challenge: Haar-random unitaries are exponentially complex



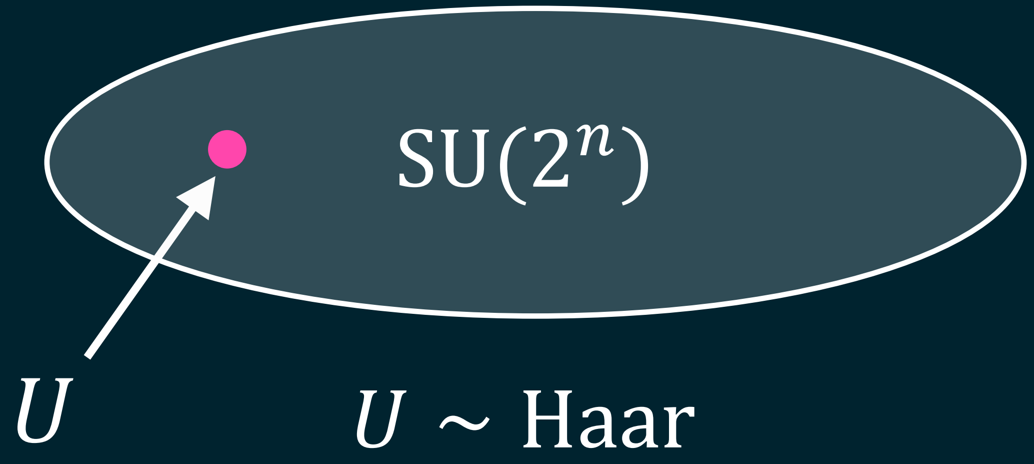
$U \sim \mathrm{Haar}$

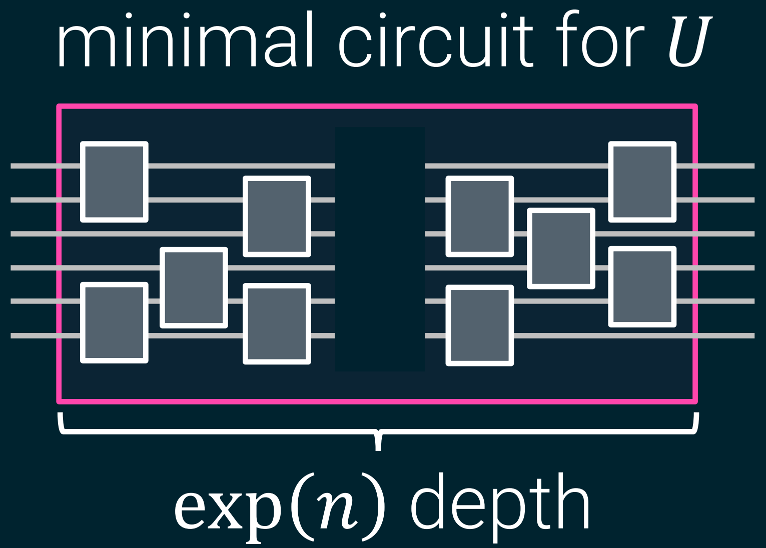
minimal circuit for U



$\exp(n)$ depth

Challenge: Haar-random unitaries are exponentially complex





So even if a Haar-random unitary "solves" your problem, this often isn't good enough!

This motivates pseudorandom unitaries (PRUs).

This motivates pseudorandom unitaries (PRUs).

Pseudorandom unitaries (PRUs): efficient quantum circuits $\{U_k\}$ s.t. no poly-time alg A can distinguish

[JLS18]

This motivates pseudorandom unitaries (PRUs).

Pseudorandom unitaries (PRUs): efficient quantum circuits $\{U_k\}$ s.t. no poly-time alg A can distinguish

- $U \leftarrow \{U_k\}$

[JLS18]

This motivates pseudorandom unitaries (PRUs).

Pseudorandom unitaries (PRUs): efficient quantum circuits $\{U_k\}$ s.t. no poly-time alg A can distinguish

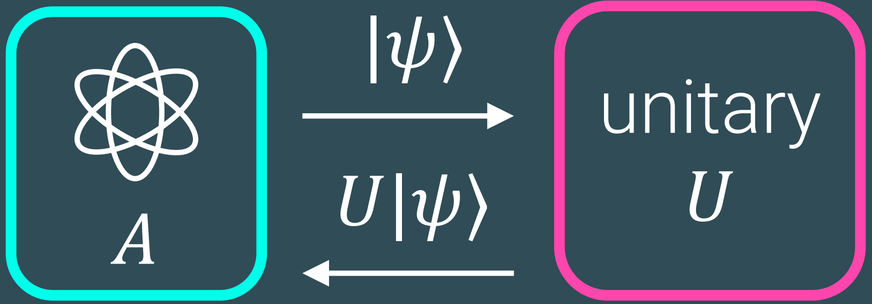
- $U \leftarrow \{U_k\}$
- $U \leftarrow \mathrm{Haar}$

[JLS18]

This motivates pseudorandom unitaries (PRUs).

Pseudorandom unitaries (PRUs): efficient quantum circuits $\{U_k\}$ s.t. no poly-time alg A can distinguish

- $U \leftarrow \{U_k\}$
- $U \leftarrow \text{Haar}$

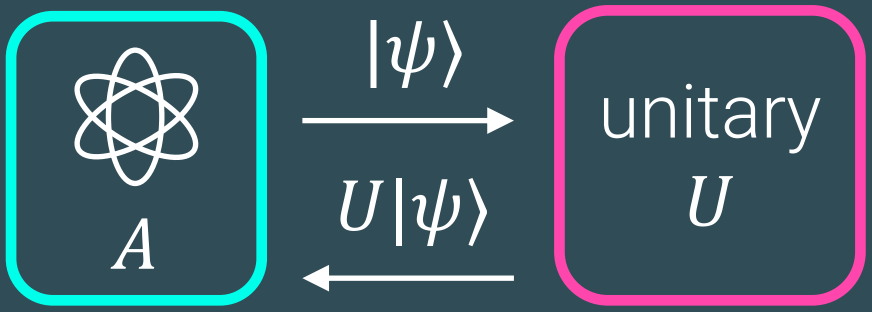


[JLS18]

This motivates pseudorandom unitaries (PRUs).

Pseudorandom unitaries (PRUs): efficient quantum circuits $\{U_k\}$ s.t. no poly-time alg A can distinguish

- $U \leftarrow \{U_k\}$
- $U \leftarrow \mathrm{Haar}$



[JLS18]

Classical analogue: pseudorandom functions (PRFs) or pseudorandom permutations (PRPs)

Obtaining provably-secure PRUs has been a central open question.

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$\underset{\text{function}}{F} \cdot \underset{\text{Hadamard}}{H} \cdots F \cdot H$$

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$F \cdot H \cdots F \cdot H$$

function Hadamard

[MPSY24]

$$P \cdot F \cdot C$$

permutation function Clifford

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$F \cdot H \cdots F \cdot H$$

function Hadamard

[MPSY24]

$$P \cdot F \cdot C$$

permutation function Clifford

Also:
[LQSYZ23]
[CBBDHX24]
...

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$F \cdot H \cdots F \cdot H$$

function Hadamard

[MPSY24]

$$P \cdot F \cdot C$$

permutation function Clifford

Also:
[LQSYZ23]
[CBBDHX24]
...

State of the art:

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$\underset{\text{function}}{F} \cdot \underset{\text{Hadamard}}{H} \cdots F \cdot H$$

[MPSY24]

$$\underset{\text{permutation}}{P} \cdot \underset{\text{function}}{F} \cdot \underset{\text{Clifford}}{C}$$

Also:
[LQSYZ23]
[CBBDHX24]
...

State of the art:

[MPSY24, CBBDHX23] obtain PRUs secure against **non-adaptive** algorithms

Obtaining provably-secure PRUs has been a central open question.

Many candidate constructions:

[JLS18]

$$F \cdot H \cdots F \cdot H$$

function Hadamard

[MPSY24]

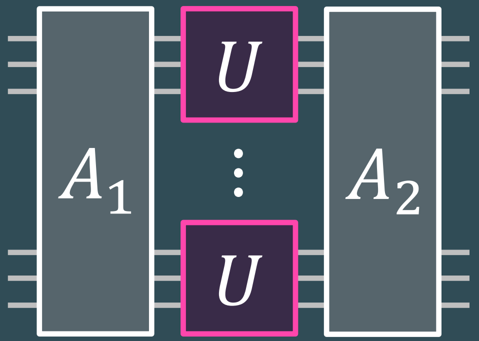
$$P \cdot F \cdot C$$

permutation function Clifford

Also:
[LQSYZ23]
[CBBDHX24]
…

State of the art:

[MPSY24, CBBDHX23] obtain PRUs secure against **non-adaptive** algorithms



Why has it been hard to prove PRU security?

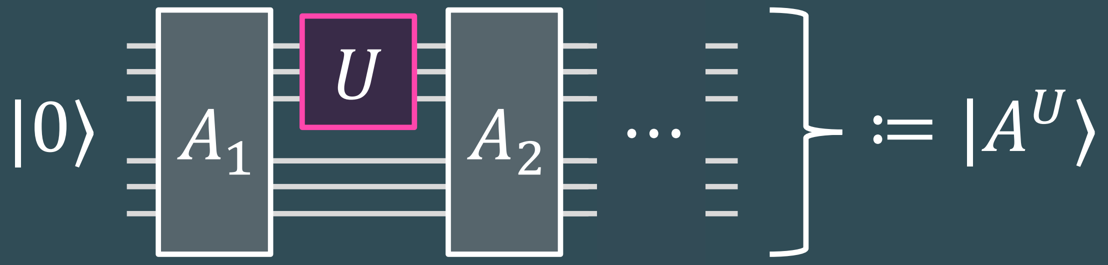
Why has it been hard to prove PRU security?

Possible reason: we lack "user-friendly" techniques for analyzing Haar-random unitaries.

Why has it been hard to prove PRU security?

Possible reason: we lack “user-friendly” techniques for analyzing Haar-random unitaries.

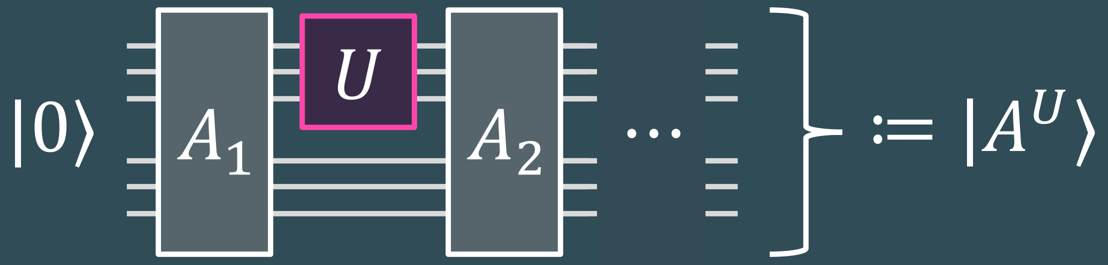
Goal:
$$\mathbb{E}_{U \leftarrow \mathrm{PRU}} |A^U\rangle\langle A^U| \approx \mathbb{E}_{U \leftarrow \mathrm{Haar}} |A^U\rangle\langle A^U|$$



Why has it been hard to prove PRU security?

Possible reason: we lack "user-friendly" techniques for analyzing Haar-random unitaries.

Goal: $\mathbb{E}_{U \leftarrow \mathtt{PRU}} |A^U\rangle\langle A^U| \approx \mathbb{E}_{U \leftarrow \mathtt{Haar}} |A^U\rangle\langle A^U|$

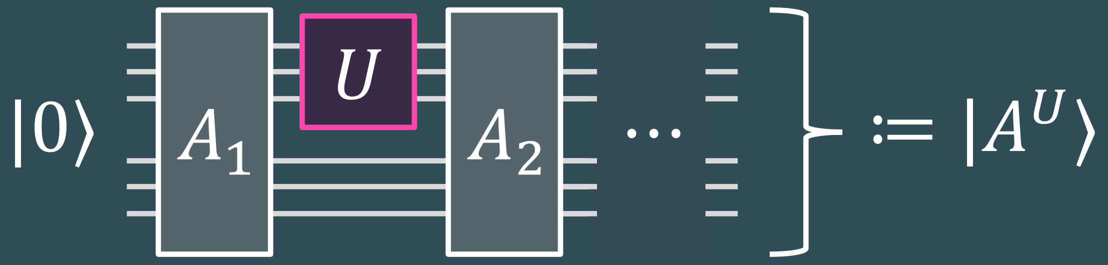


bounding moments of Haar-random U is often quite involved!

Why has it been hard to prove PRU security?

Possible reason: we lack “user-friendly” techniques for analyzing Haar-random unitaries.

Goal: $$\mathbb{E}_{U\leftarrow \mathrm{PRU}} |A^U\rangle\langle A^U| \approx \mathbb{E}_{U\leftarrow \mathrm{Haar}} |A^U\rangle\langle A^U|$$



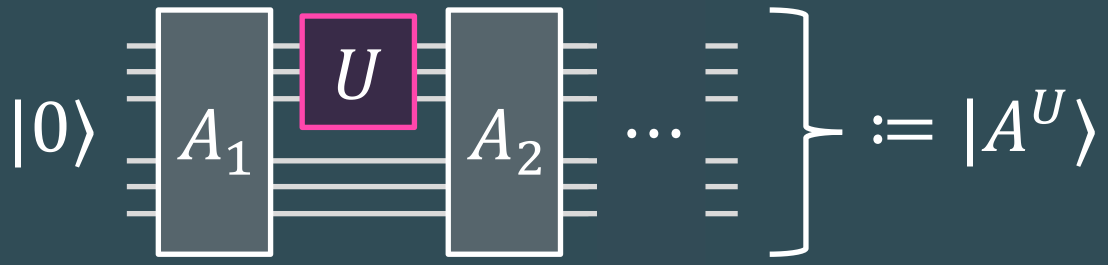
bounding moments of Haar-random U is often quite involved!

e.g., Weingarten calculus

Why has it been hard to prove PRU security?

Possible reason: we lack “user-friendly” techniques for analyzing Haar-random unitaries.

Goal: $\mathbb{E}_{U \leftarrow \mathrm{PRU}} |A^U\rangle\langle A^U| \approx \mathbb{E}_{U \leftarrow \mathrm{Haar}} |A^U\rangle\langle A^U|$



bounding moments of Haar-random U is often quite involved!

e.g., Weingarten calculus

Theorem 3.1. *Let* k *be a positive integer. For any permutation* $\sigma \in \mathcal{S}_k$ *and nonnegative integer* g, *we have*

$$(k-1)^g \#P(\sigma, |\sigma|) \leq \#P(\sigma, |\sigma| + 2g) \leq (6k^{7/2})^g \#P(\sigma, |\sigma|).$$

Theorem 3.2. *For any* $\sigma \in \mathcal{S}_k$ *and* $d > \sqrt{6}k^{7/4}$,

$$\frac{1}{1 - \frac{k-1}{d^2}} \leq \frac{(-1)^{|\sigma|} d^{k+|\sigma|} \mathrm{Wg}^U(\sigma, d)}{\#P(\sigma, |\sigma|)} \leq \frac{1}{1 - \frac{6k^{7/2}}{d^2}}.$$

In addition, the l.h.s inequality is valid for any $d \geq k$.

This work

This work

Theorem: PRUs exist (assuming one-way functions)

$$U = P \cdot F \cdot C$$ [MPSY24]

permutation function Clifford

This work

Theorem: PRUs exist
(assuming one-way functions)

$$\underbrace{}_{}U = P \cdot F \cdot C$$

[MPSY24]

permutation function Clifford

We prove this via a new technique: **the path-recording oracle**.

This work

Theorem: PRUs exist
(assuming one-way functions)

$$U = P \cdot F \cdot C$$ [MPSY24]

permutation function Clifford

We prove this via a new technique: **the path-recording oracle**.

- analyze random unitaries using purification

This work

Theorem: PRUs exist (assuming one-way functions)

$$U = P \cdot F \cdot C$$ [MPSY24]

permutation function Clifford

We prove this via a new technique: **the path-recording oracle**.

- analyze random unitaries using purification
- **we also show:** any algorithm that queries a Haar-random U can be efficiently implemented (to inverse-exp error)

This work

In fact, we go a step further.

This work

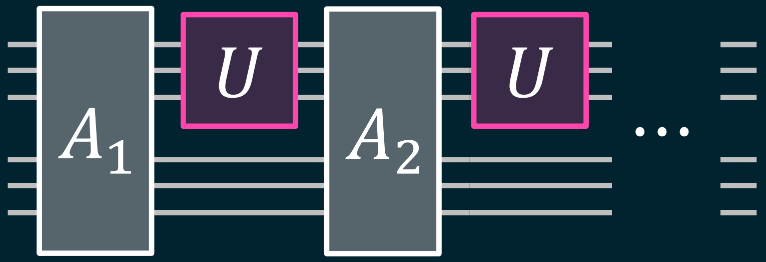
In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U.

This work

In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U. **What if it queries both U and $U^\dagger$?**

This work

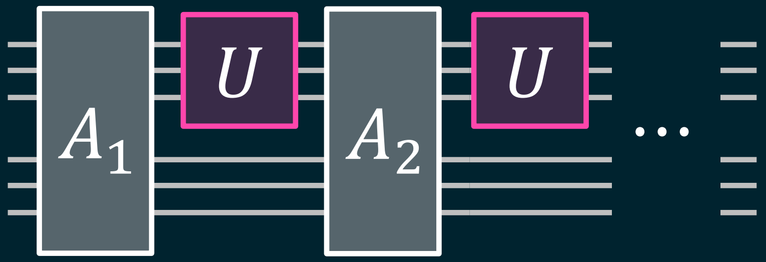
In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U. **What if it queries both U and $U^\dagger$?**



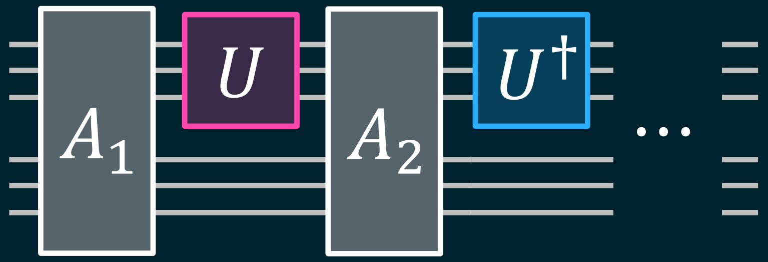
Standard PRU distinguisher

This work

In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U. **What if it queries both U and $U^\dagger$?**



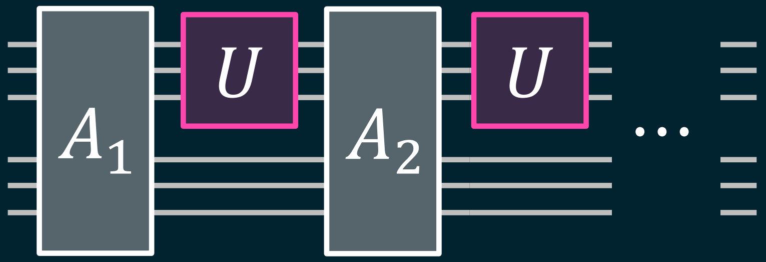
Standard PRU distinguisher



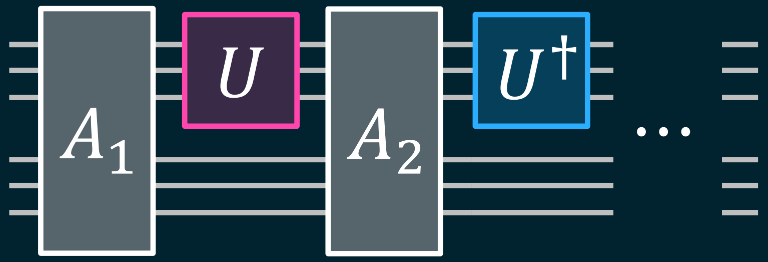
Strong PRU distinguisher

This work

In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U. **What if it queries both U and $U^\dagger$?**



Standard PRU distinguisher



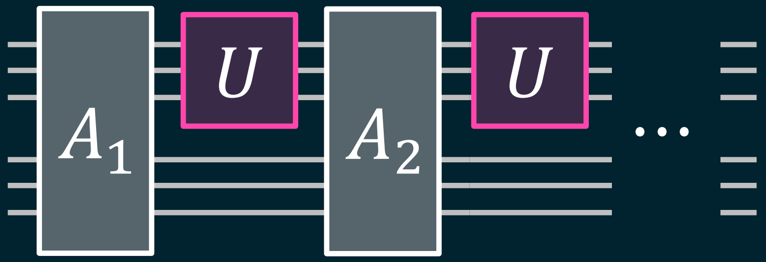
Strong PRU distinguisher

Theorem: ***Strong*** PRUs exist (assuming one-way functions)

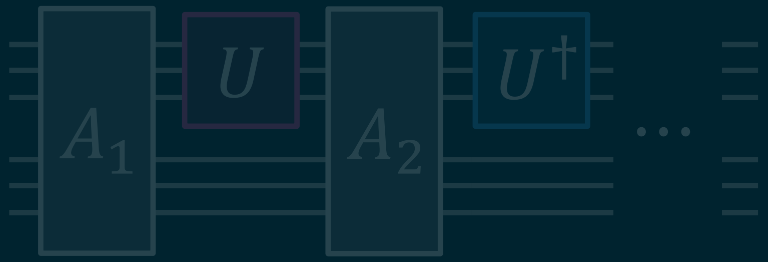
$$U = C_1 \cdot P \cdot F \cdot C_2$$

This work

In fact, we go a step further. In the [JLS18] PRU definition, the distinguisher can only query U. **What if it queries both U and $U^\dagger$?**



Standard PRU distinguisher



Strong PRU distinguisher

This talk

Theorem: ***Strong*** PRUs exist (assuming one-way functions)

$$U = C_1 \cdot P \cdot F \cdot C_2$$

Key technical idea: a new way to **efficiently simulate** Haar-random unitaries

Key technical idea: a new way to **efficiently simulate** Haar-random unitaries

Plan:

Key technical idea: a new way to **efficiently simulate** Haar-random unitaries

Plan:

(1) efficiently simulating a random **function**

Key technical idea: a new way to **efficiently simulate** Haar-random unitaries

Plan:

(1) efficiently simulating a random **function**

(2) efficiently simulating a Haar-random **unitary**

Key technical idea: a new way to **efficiently simulate** Haar-random unitaries

Plan:

(1) efficiently simulating a random **function**

(2) efficiently simulating a Haar-random **unitary**

(3) two proofs at once:
- our simulator works
- PRUs exist

Up next:

How to simulate a random **function**

Simulating a random **function**

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \rightarrow \{0,1\}$.

Standard

A $\qquad \boxed{f}$

random f

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

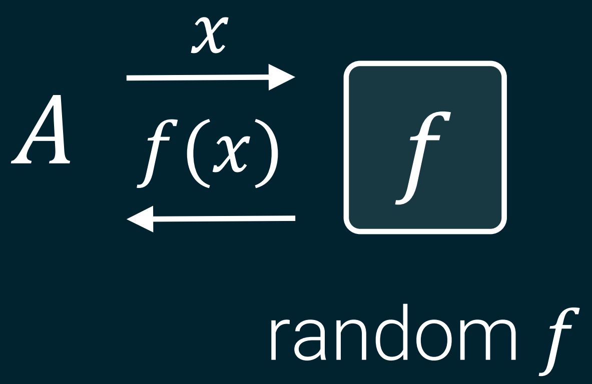
$$A \xrightarrow{x} \boxed{f}$$

random f

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f:\{0,1\}^n \to \{0,1\}$.

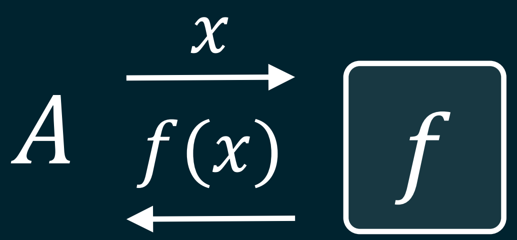
Standard



Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard



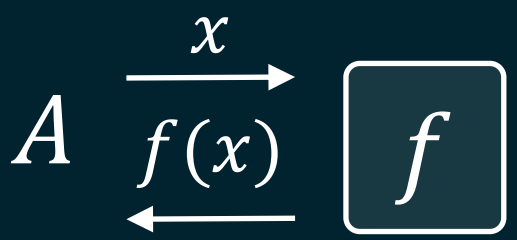
random f

(exponential time)

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard



random f

(exponential time)

Simulation

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$A \xrightarrow{x} \boxed{f}$, $\boxed{f} \xrightarrow{f(x)} A$

random f

(exponential time)

Simulation

A

Initialization: $R = \emptyset$

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$A \underset{f(x)}{\overset{x}{\rightleftarrows}} \boxed{f}$

random f

(exponential time)

Simulation

$A \xrightarrow{x}$

Initialization: $R = \emptyset$

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f\colon\{0,1\}^n \to \{0,1\}$.

Standard

$$A \;\underset{f(x)}{\overset{x}{\rightleftarrows}}\; \boxed{f}$$

random f

(exponential time)

Simulation

$$A \xrightarrow{x}$$

Initialization: $R = \emptyset$

If x was not queried before:

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$A \xrightarrow{x}$ $\boxed{f}$

$\xleftarrow{f(x)}$

random f

(exponential time)

Simulation

$A \xrightarrow{x}$

Initialization: $R = \emptyset$

If x was not queried before:

- sample $y \leftarrow \{0,1\}$

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$$A \underset{f(x)}{\overset{x}{\rightleftarrows}} \boxed{f}$$

random f

(exponential time)

Simulation

$$A \underset{y}{\overset{x}{\rightleftarrows}}$$

Initialization: $R = \emptyset$

If x was not queried before:

- sample $y \leftarrow \{0,1\}$
- insert (x, y) into R

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f\colon \{0,1\}^n \to \{0,1\}$.

Standard

$A \underset{f(x)}{\overset{x}{\rightleftarrows}} \boxed{f}$

random f

(exponential time)

Simulation

$A \underset{y}{\overset{x}{\rightleftarrows}}$

Initialization: $R = \emptyset$

If x was not queried before:

- sample $y \leftarrow \{0,1\}$
- insert (x, y) into R

If x was queried before:

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$$A \underset{f(x)}{\overset{x}{\rightleftarrows}} \boxed{f}$$

random f

(exponential time)

Simulation

$$A \underset{y}{\overset{x}{\rightleftarrows}}$$

Initialization: $R = \emptyset$

If x was not queried before:

- sample $y \leftarrow \{0,1\}$
- insert (x, y) into R

If x was queried before:

- look up (x, y), return y

Simulating a random function

Warmup: A^f is a **classical** alg querying a random $f: \{0,1\}^n \to \{0,1\}$.

Standard

$$A \underset{f(x)}{\overset{x}{\rightleftarrows}} \boxed{f}$$

random f

(exponential time)

Simulation

$$A \underset{y}{\overset{x}{\rightleftarrows}}$$

Initialization: $R = \emptyset$

If x was not queried before:

- sample $y \leftarrow \{0,1\}$
- insert (x, y) into R

If x was queried before:

- look up (x, y), return y

(polynomial-time + stateful)

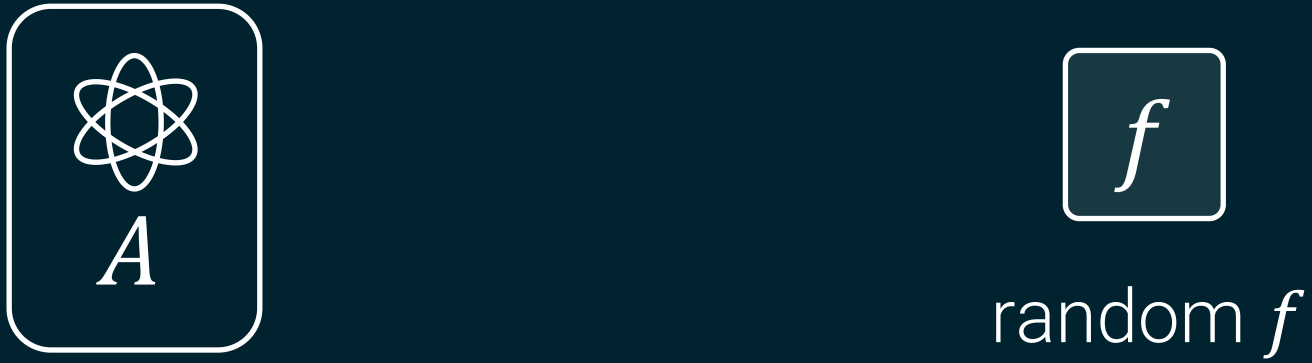
Simulating a random function

Simulating a random function

What if A^f is quantum and queries f in **superposition**?

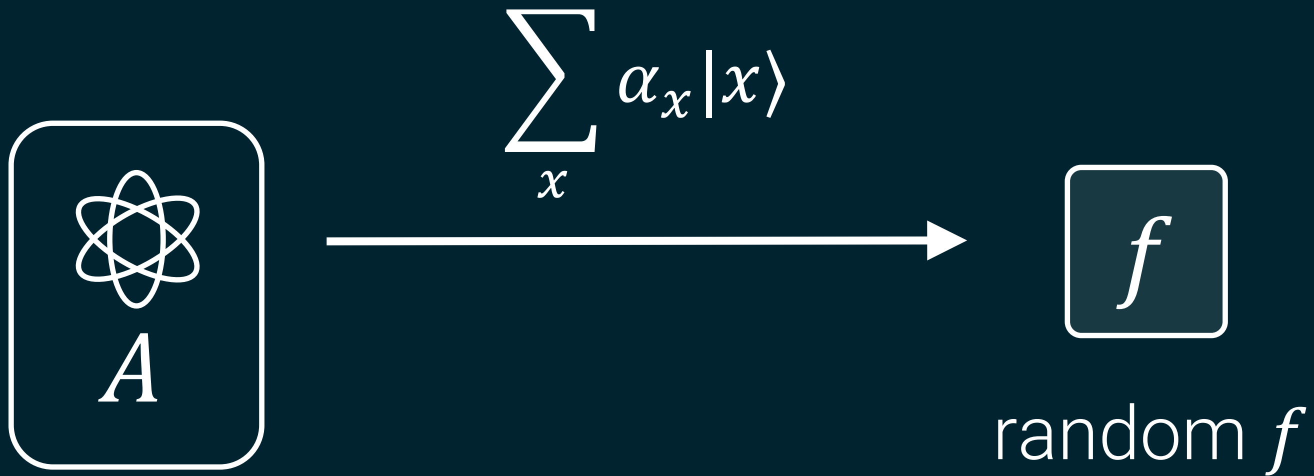
Simulating a random function

What if A^f is quantum and queries f in **superposition**?



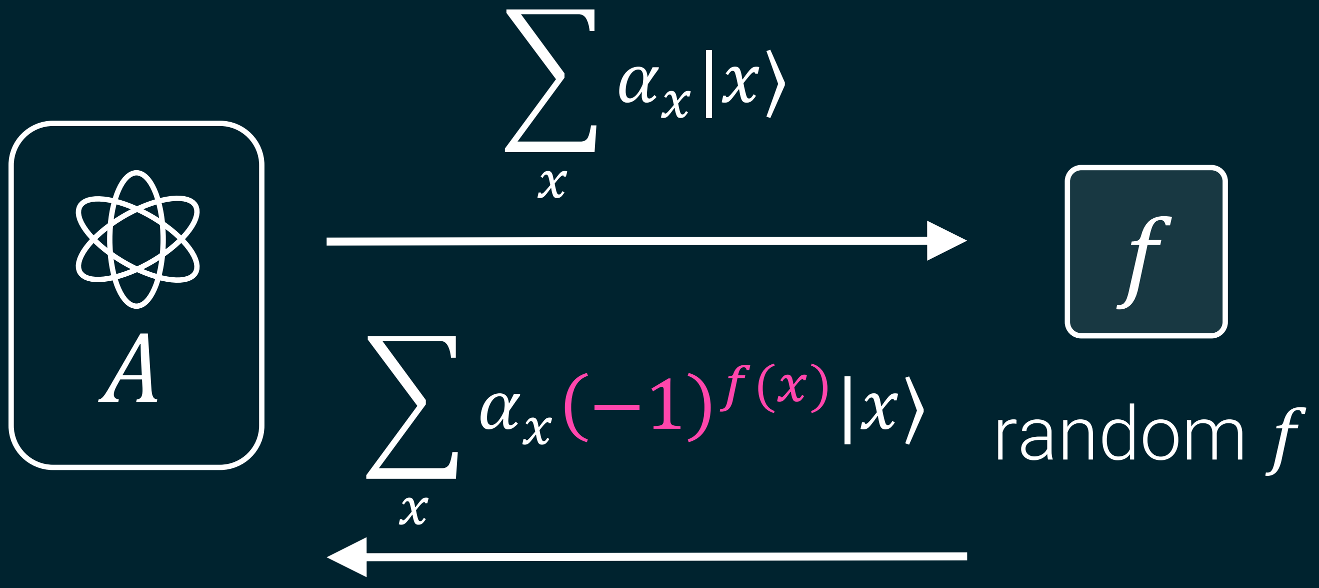
Simulating a random **function**

What if A^f is quantum and queries f in **superposition**?



Simulating a random **function**

What if A^f is quantum and queries f in **superposition**?



Simulating a random **function**

What if A^f is quantum and queries f in **superposition**?

$$\sum_x \alpha_x |x\rangle$$

A

f

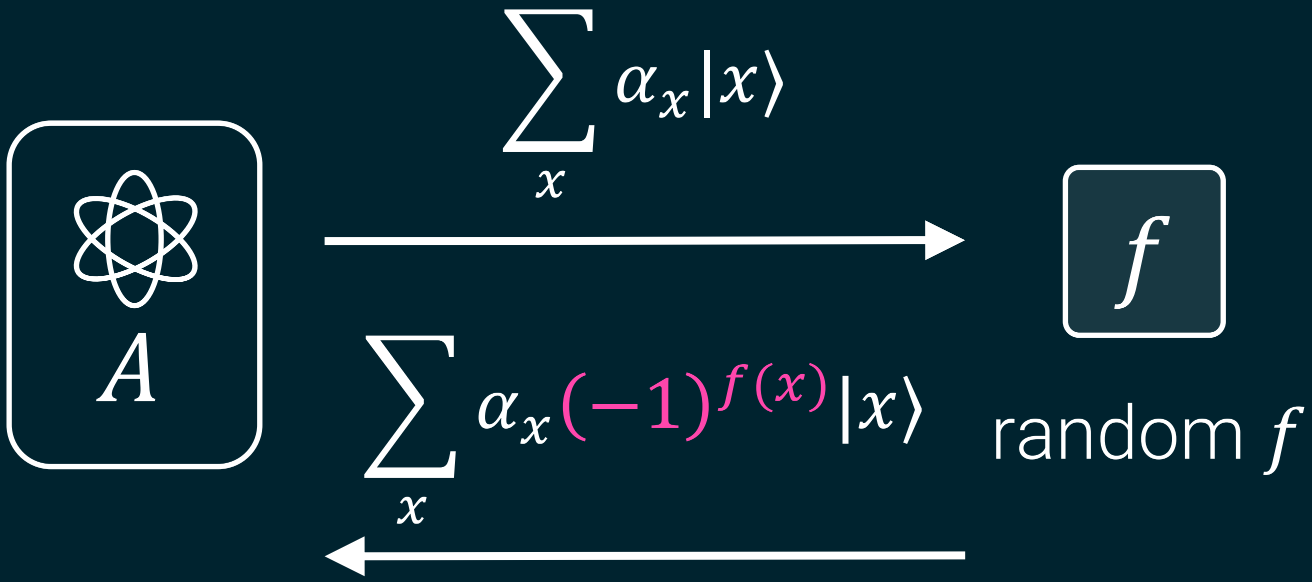
$$\sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

random f

Unclear how to sample $f(x)$ “on the fly.”

Simulating a random **function**

What if A^f is quantum and queries f in **superposition**?

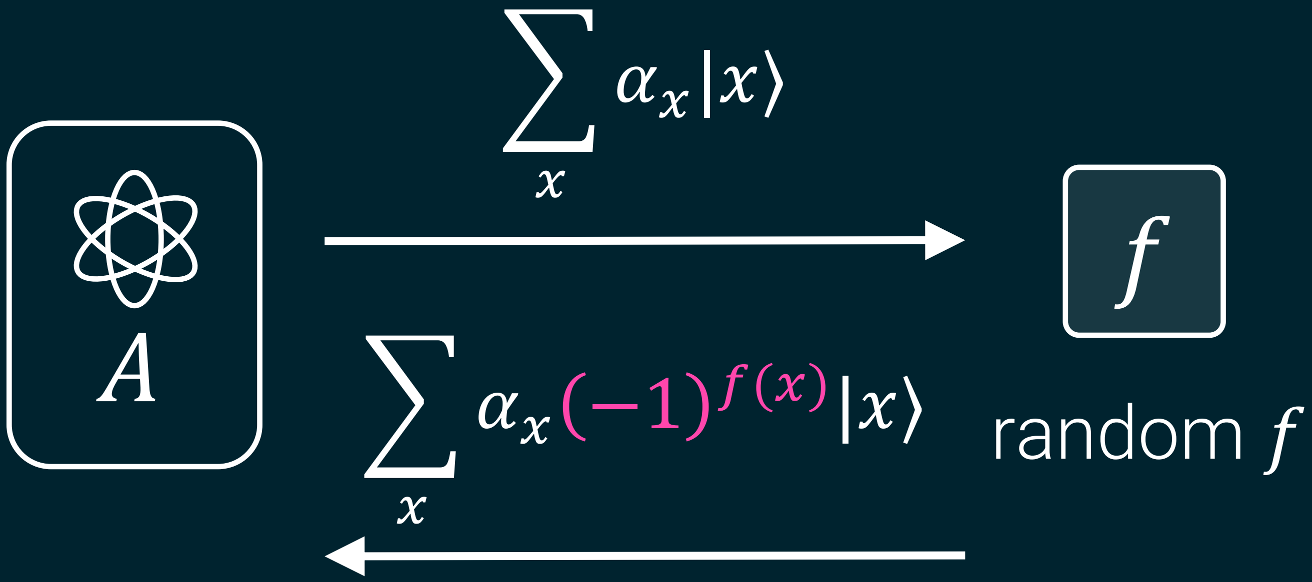


Unclear how to sample $f(x)$ "on the fly."

Simulation seems to require knowing x, but measuring x destroys the superposition!

Simulating a random **function**

What if A^f is quantum and queries f in **superposition**?



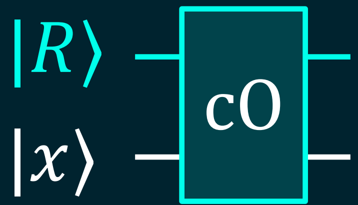
Unclear how to sample $f(x)$ "on the fly."

Simulation seems to require knowing x, but measuring x destroys the superposition!

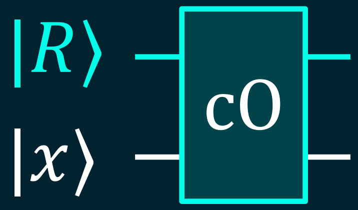
Solution: the compressed oracle [Zhandry18]

Compressed oracle [Z18]:

Compressed oracle [Z18]:

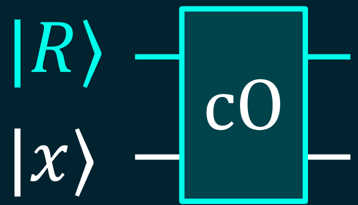


Compressed oracle [Z18]:



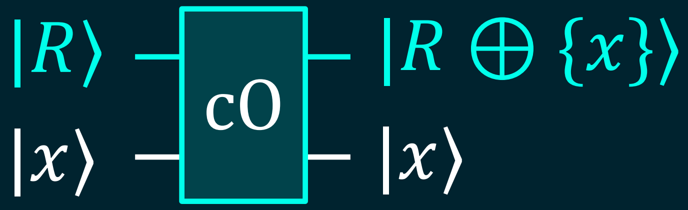
- $R = \{x_1, \dots, x_t\}$ is a **set**

Compressed oracle [Z18]:



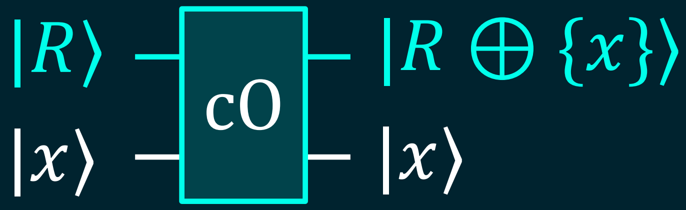
- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R

Compressed oracle [Z18]:



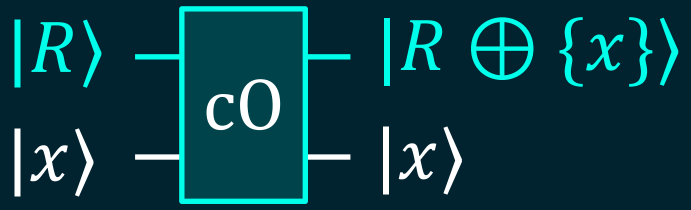
- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R

Compressed oracle [Z18]:



- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

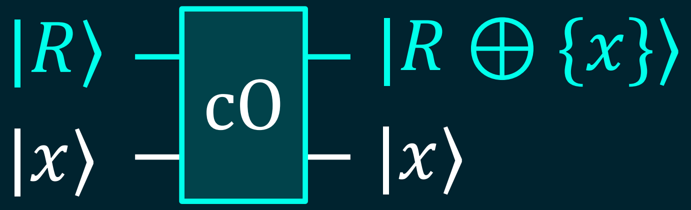
Compressed oracle [Z18]:



- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

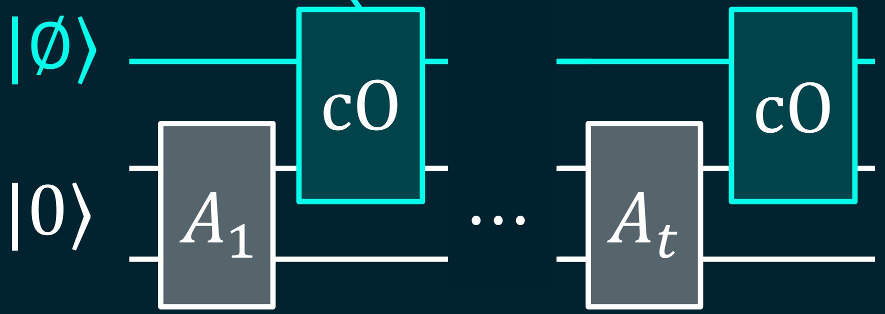
Intuition: think of cO as "recording" x onto the $|R\rangle$ register

Compressed oracle [Z18]:



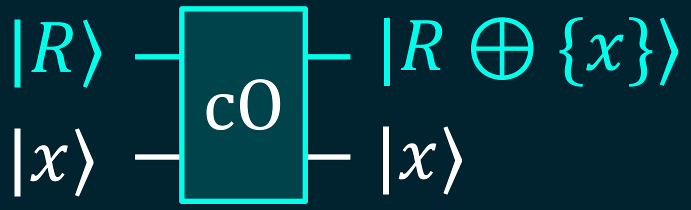
- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

Intuition: think of cO as "recording" x onto the $|R\rangle$ register

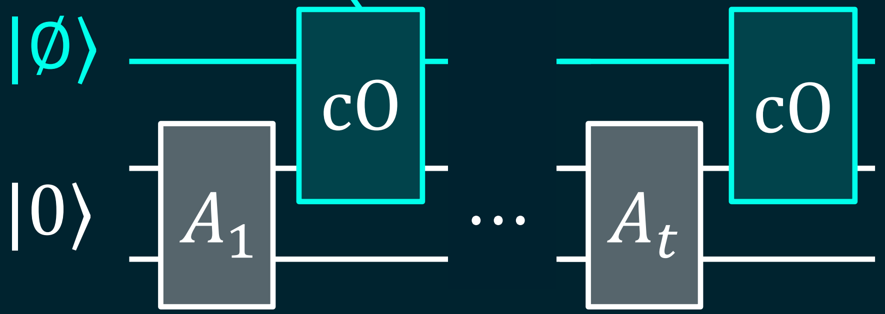


Efficient simulation

Compressed oracle [Z18]:

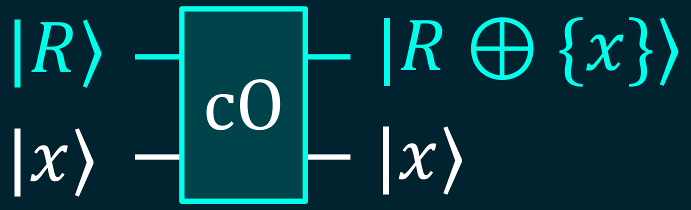


- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference



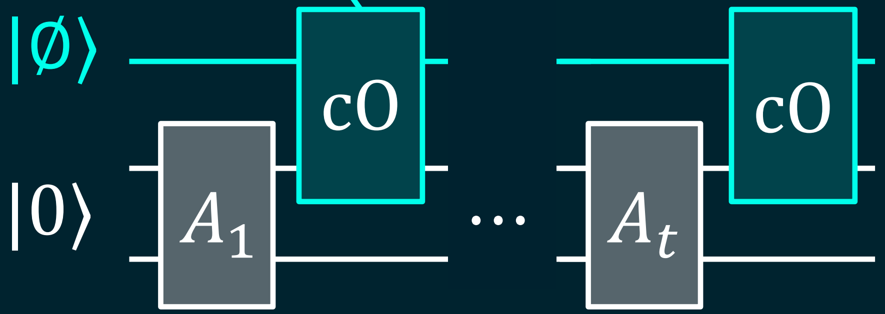
Efficient simulation

Compressed oracle [Z18]:



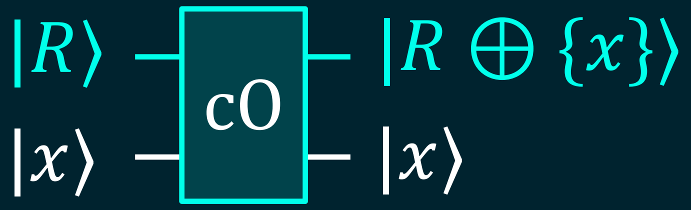
- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

Standard (exp-time)

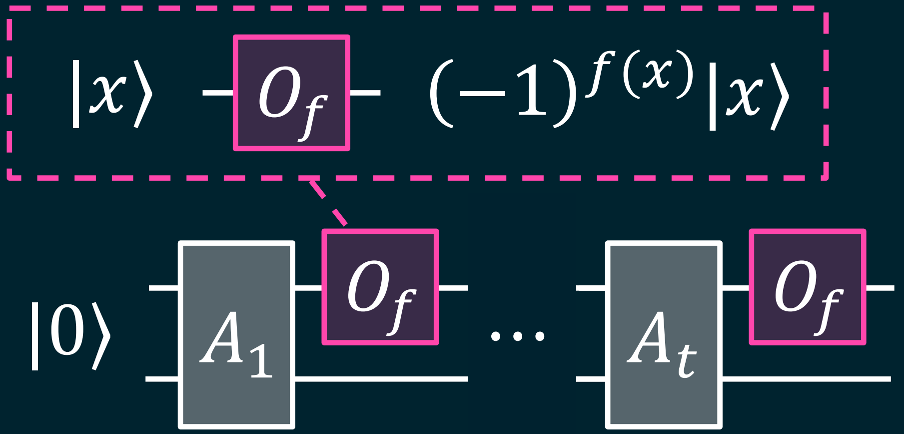


Efficient simulation

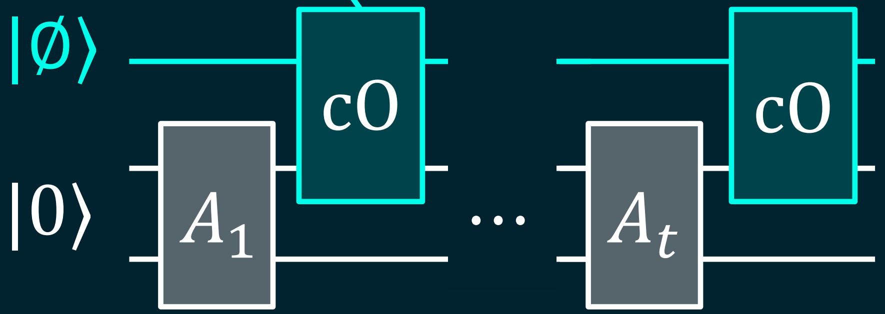
Compressed oracle [Z18]:



- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

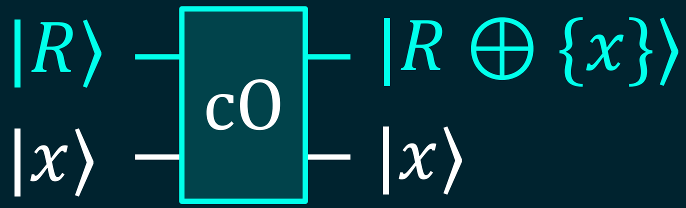


Standard (exp-time)

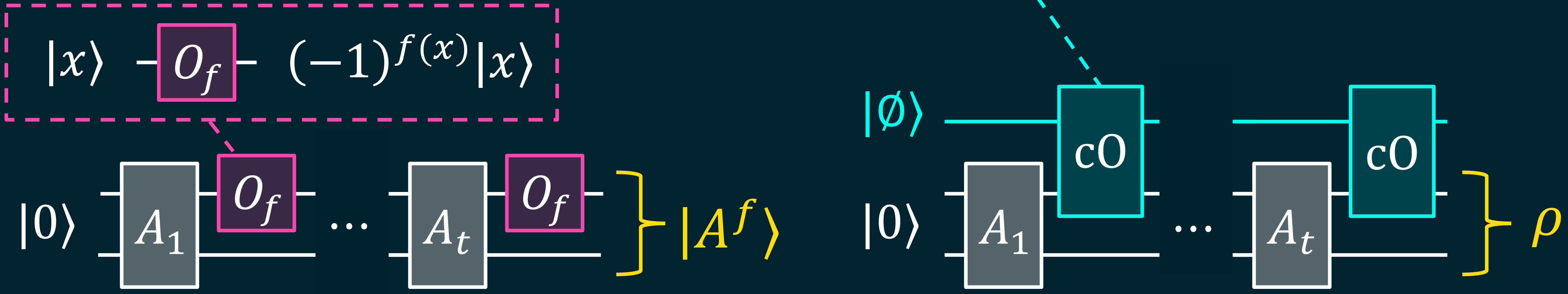


Efficient simulation

Compressed oracle [Z18]:



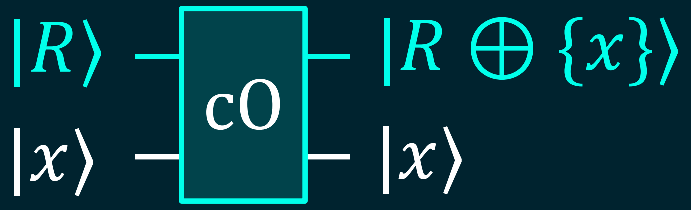
- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference



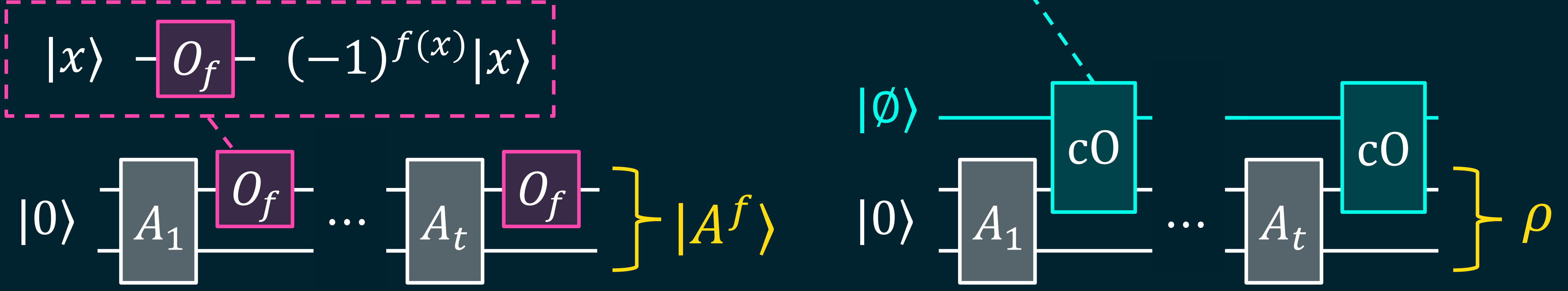
Standard (exp-time)

Efficient simulation

Compressed oracle [Z18]:



- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference



Standard (exp-time) | Efficient simulation

[Z18] proves cO is a perfect simulation: $\mathbb{E}_f \, |A^f\rangle\langle A^f| = \rho$

Compressed oracle [Z18]:

$$|R\rangle \;-\!\boxed{\mathrm{cO}}\!-\; |R \oplus \{x\}\rangle$$
$$|x\rangle \;-\!\boxed{\phantom{\mathrm{cO}}}\!-\; |x\rangle$$

- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

How [Z18] proves it:

[Z18] proves cO is a perfect simulation: $\mathbb{E}_f\, |A^f\rangle\langle A^f| = \rho$

Compressed oracle [Z18]:

$$\begin{array}{ccc} |R\rangle & \multirow{2}{*}{\boxed{\mathrm{cO}}} & |R \oplus \{x\}\rangle \\ |x\rangle & & |x\rangle \end{array}$$

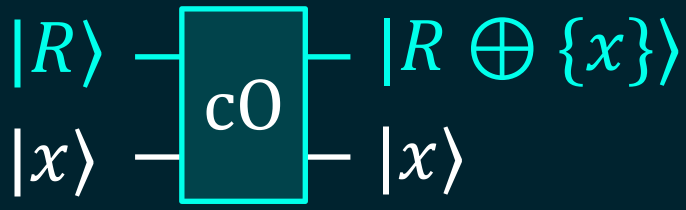
- $R = \{x_1, \dots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

How [Z18] proves it: Replace **random** function f with **purification**

$$\sum_f |A^f\rangle \otimes |f\rangle,$$

[Z18] proves cO is a perfect simulation: $\mathbb{E}_f\, |A^f\rangle\langle A^f| = \rho$

Compressed oracle [Z18]:



- $R = \{x_1, \ldots, x_t\}$ is a **set**
- $|R\rangle$ is a unit vector labeled by R
- $\oplus$ is symmetric difference

How [Z18] proves it: Replace **random** function f with **purification**

$$\sum_f |A^f\rangle \otimes |f\rangle,$$

and view $|f\rangle$ in the Fourier basis.

[Z18] proves cO is a perfect simulation: $\mathbb{E}_f \, |A^f\rangle\langle A^f| = \rho$

Another perspective:

[Z18] simulates queries to random **diagonal unitaries**

Another perspective:

[Z18] simulates queries to random **diagonal unitaries**

$$O_f = \begin{pmatrix} +1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix}$$

Another perspective:

[Z18] simulates queries to random **diagonal unitaries**

$$O_f = \begin{pmatrix} +1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix}$$

This work:

Simulate queries to **Haar-random unitaries**

Another perspective:

[Z18] simulates queries to random **diagonal unitaries**

$$O_f = \begin{pmatrix} +1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix}$$

This work:

Simulate queries to **Haar-random unitaries**

$$U = \begin{pmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ U_{21} & U_{22} & & \\ \vdots & & \ddots & \\ U_{N1} & & & U_{NN} \end{pmatrix}$$

Another perspective:

[Z18] simulates queries to random **diagonal unitaries**

$$O_f = \begin{pmatrix} +1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix}$$

independent

This work:

Simulate queries to **Haar-random unitaries**

$$U = \begin{pmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ U_{21} & U_{22} & & \\ \vdots & & \ddots & \\ U_{N1} & & & U_{NN} \end{pmatrix}$$

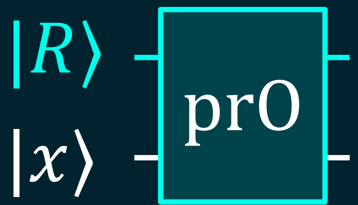
not independent!

Up next: the path-recording oracle

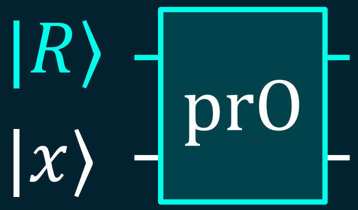
(our simulator for Haar-random unitaries)

The path-recording oracle prO

The path-recording oracle prO



The path-recording oracle prO



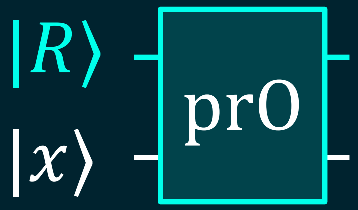
- $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ is a set of ordered pairs

The path-recording oracle prO

$$\mathrm{prO}\,|R\rangle|x\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ is a set of ordered pairs

The path-recording oracle prO



$$\Big\} \sum_{y \notin R} |y\rangle |R \cup \{(x, y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

The path-recording oracle prO

$$|R\rangle, |x\rangle \xrightarrow{\mathrm{prO}} \sum_{y \notin R} |y\rangle |R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

(actually, we should have a $\frac{1}{\sqrt{N-|R|}}$ in front)

The path-recording oracle prO

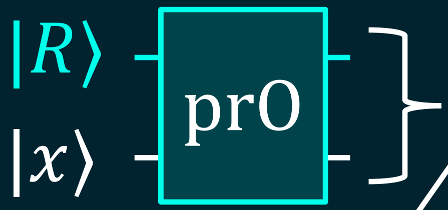
$$\mathrm{prO}\,|R\rangle|x\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

(actually, we should have a $\frac{1}{\sqrt{N-|R|}}$ in front)

Note: prO is an isometry.

The path-recording oracle prO



$$\sum_{y \notin R} |y\rangle |R \cup \{(x, y)\}\rangle$$

- $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \ldots, y_t\}$

(actually, we should have a $\frac{1}{\sqrt{N - |R|}}$ in front)

Note: prO is an isometry.

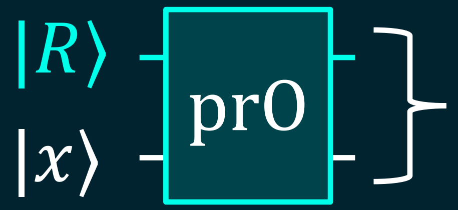
Intuition: $|y\rangle |R \cup \{(x, y)\}\rangle$ uniquely determines $|x\rangle |R\rangle$.

The path-recording oracle prO

$$\sum_{y\notin R} |y\rangle |R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

The path-recording oracle prO

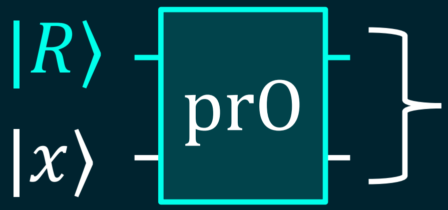


$$\sum_{y \notin R} |y\rangle |R \cup \{(x, y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

Standard (exp-time)

The path-recording oracle prO



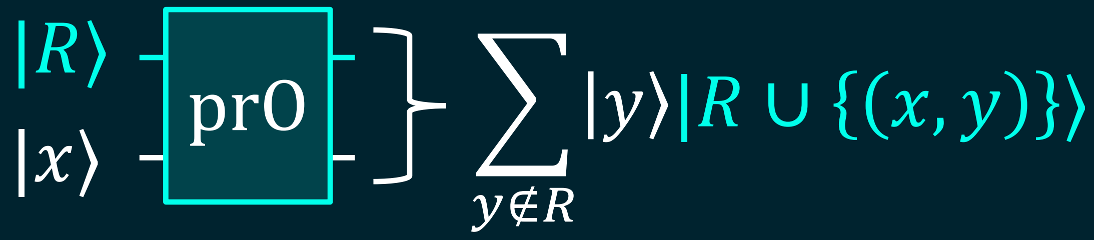
$$\sum_{y \notin R} |y\rangle |R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \ldots, y_t\}$

Standard (exp-time)

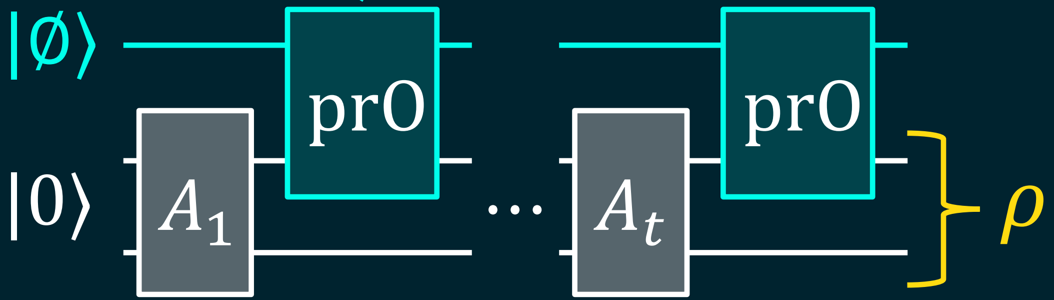
Efficient simulation

The path-recording oracle prO



- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

Standard (exp-time)



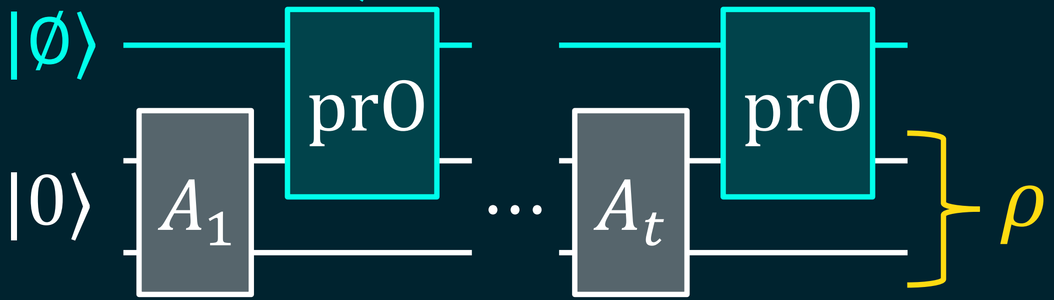
Efficient simulation

The path-recording oracle prO

$$|R\rangle|x\rangle \xrightarrow{\mathrm{prO}} \sum_{y \notin R} |y\rangle |R \cup \{(x,y)\}\rangle$$

- $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$ is a set of ordered pairs
- sum over $y \notin \{y_1, \dots, y_t\}$

Standard (exp-time)



Efficient simulation

We show: $\mathbb{E}_{U \leftarrow \mathrm{Haar}} |A^U\rangle\langle A^U|$ and ρ have trace distance $\leq t^2/2^n$.

Up next: a few examples

Example 1: one query on $|0\rangle$

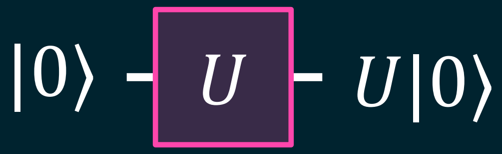
$$|0\rangle \;-\!\boxed{U}\!-\; U|0\rangle$$

Example 1: one query on $|0\rangle$

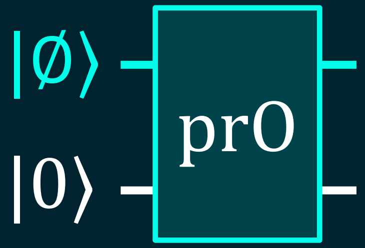
$$|0\rangle \;-\boxed{U}-\; U|0\rangle$$

Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*

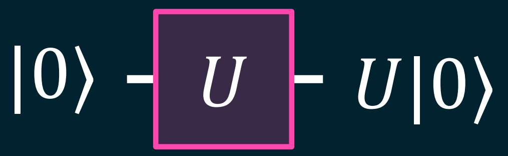
Example 1: one query on $|0\rangle$



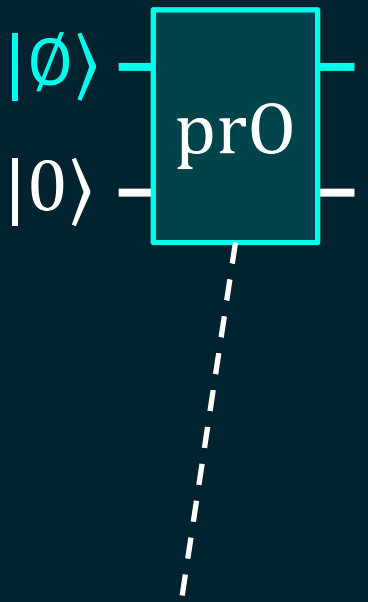
Average over $U \leftarrow \text{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*



Example 1: one query on $|0\rangle$

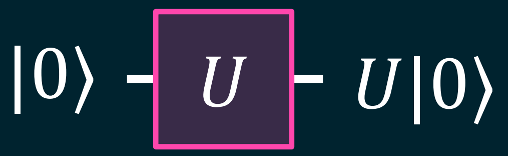


Average over $U \leftarrow \text{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*

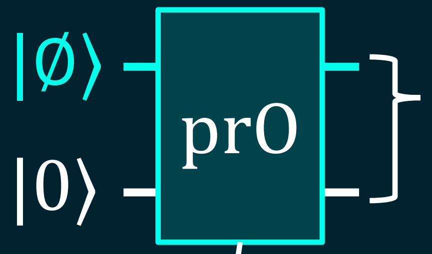


$$\mathrm{prO}\,|x\rangle|R\rangle = \sum_{y\notin R}|y\rangle|R \cup \{(x,y)\}\rangle$$

Example 1: one query on $|0\rangle$



Average over $U \leftarrow \text{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*



$\sum_y |y\rangle \otimes |\{(0, y)\}\rangle$

$$\mathrm{prO}\,|x\rangle|R\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Example 1: one query on $|0\rangle$

$|0\rangle$ — $\boxed{U}$ — $U|0\rangle$

Average over $U \leftarrow \text{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*

$$\begin{array}{l} |\emptyset\rangle \\ |0\rangle \end{array} \boxed{\mathrm{prO}} \; \Sigma_y |y\rangle \otimes \underbrace{|\{(0,y)\}\rangle}_{\text{trace out/measure}}$$

$$\mathrm{prO}\ |x\rangle|R\rangle = \Sigma_{y \notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

Example 1: one query on $|0\rangle$

$$|0\rangle \;-\boxed{U}-\; U|0\rangle$$

Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle$ becomes the *maximally mixed state.*

$$\left.\begin{array}{l} |\emptyset\rangle \;-\!\!\!\!\\ |0\rangle \;-\!\!\!\! \end{array}\boxed{\mathrm{prO}}\right\} \sum_y |y\rangle \otimes \underbrace{|\{(0,y)\}\rangle}_{\text{trace out/measure}}$$

After tracing out: uniform mixture over $|y\rangle$, which is the *maximally mixed state.*

$$\mathrm{prO}\ |x\rangle|R\rangle = \sum_{y\notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

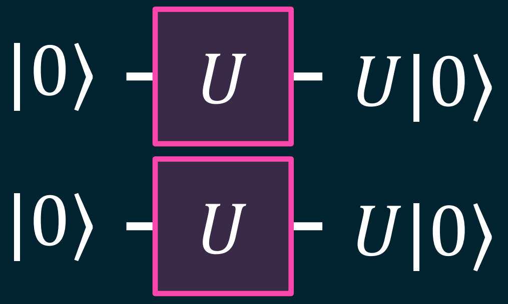
Example 2: two queries on $|0\rangle$

Example 2: two queries on $|0\rangle$

$$|0\rangle \;-\boxed{U}-\; U|0\rangle$$

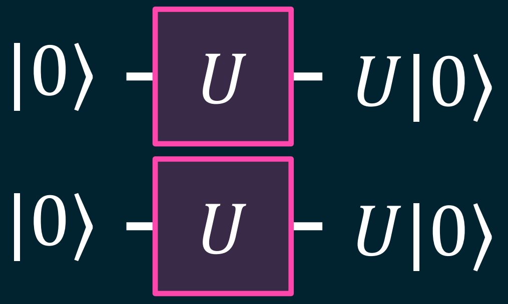
$$|0\rangle \;-\boxed{U}-\; U|0\rangle$$

Example 2: two queries on $|0\rangle$

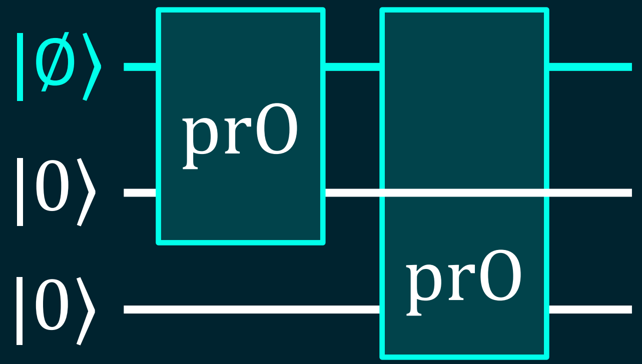


Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.

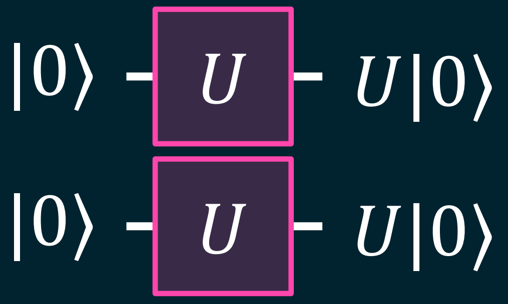
Example 2: two queries on $|0\rangle$



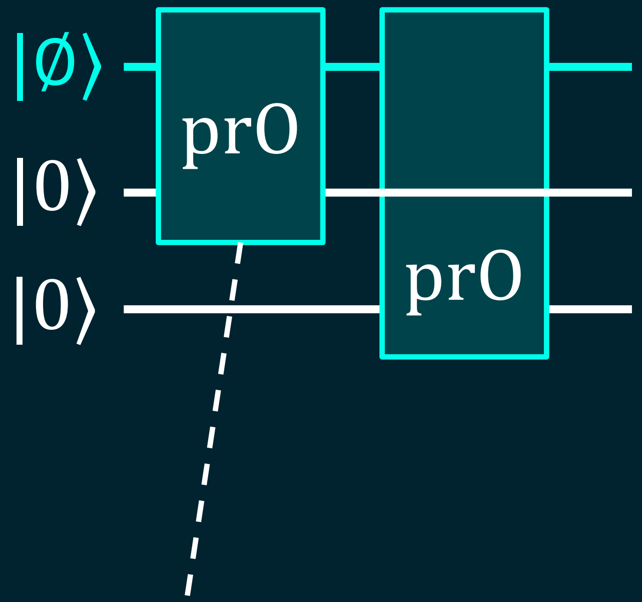
Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.



Example 2: two queries on $|0\rangle$

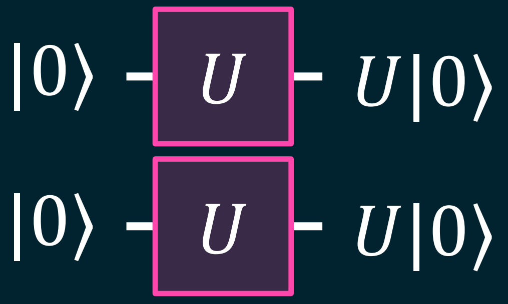


Average over $U \leftarrow \text{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.

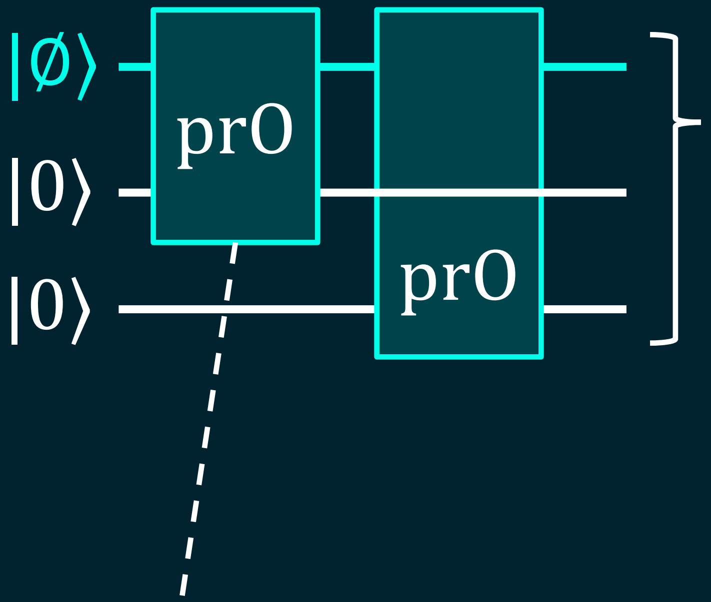


$$\mathrm{prO}\, |x\rangle|R\rangle = \textstyle\sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Example 2: two queries on $|0\rangle$



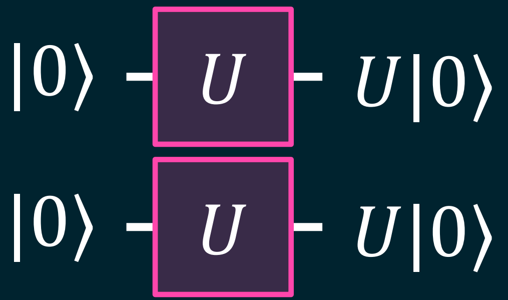
Average over $U \leftarrow$ Haar: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.



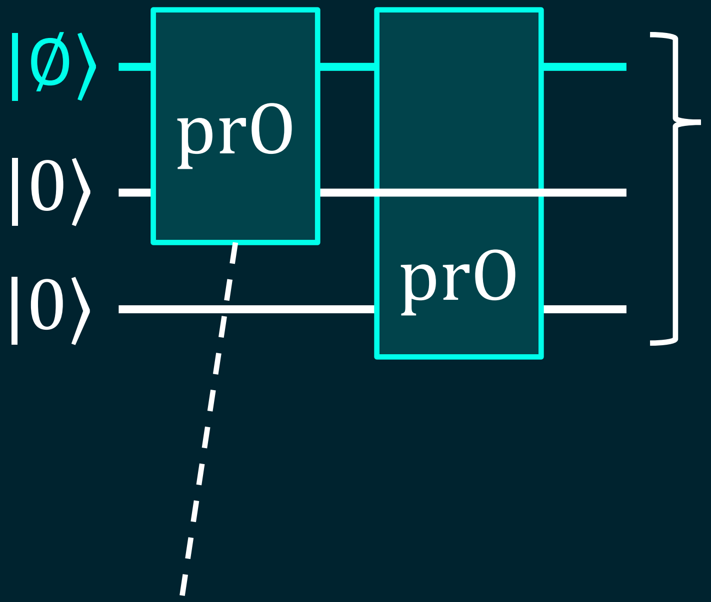
$\sum_{y_1 \neq y_2} |y_1, y_2\rangle \otimes |\{(0, y_1), (0, y_2)\}\rangle$

$$\mathrm{prO}\ |x\rangle|R\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Example 2: two queries on $|0\rangle$



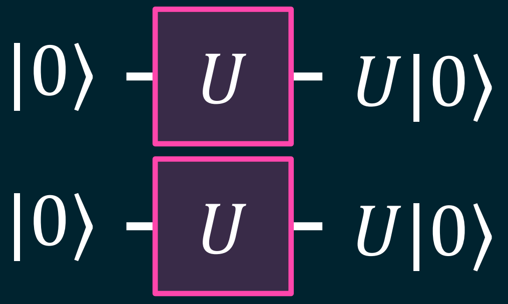
Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.



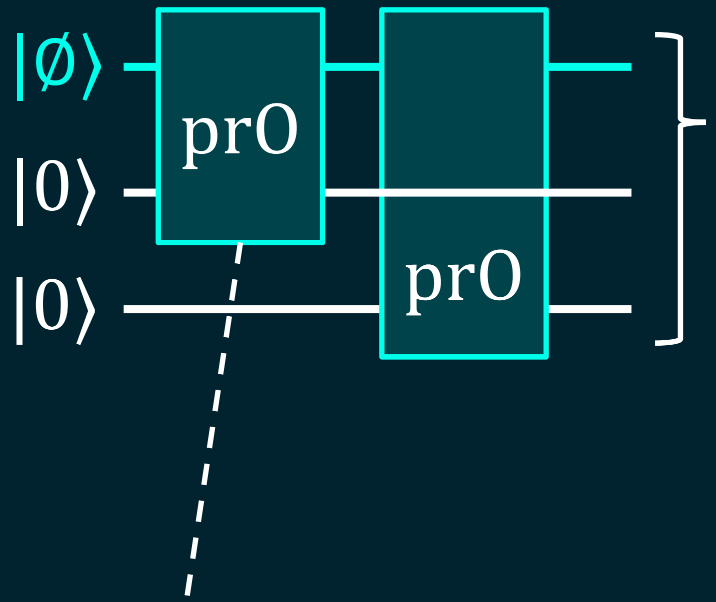
$$\sum_{y_1 \neq y_2} |y_1, y_2\rangle \otimes \underbrace{|\{(0, y_1), (0, y_2)\}\rangle}_{\text{trace out/measure}}$$

$$\mathrm{prO}\, |x\rangle|R\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Example 2: two queries on $|0\rangle$



Average over $U \leftarrow \mathrm{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.

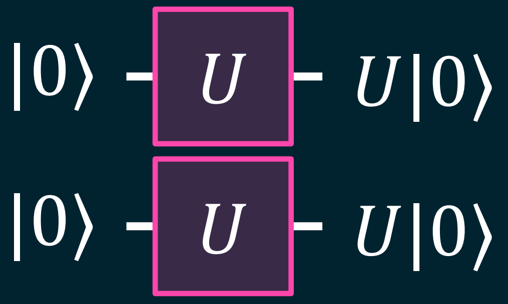


$\sum_{y_1 \neq y_2} |y_1, y_2\rangle \otimes \underbrace{|\{(0, y_1), (0, y_2)\}\rangle}_{\text{trace out/measure}}$

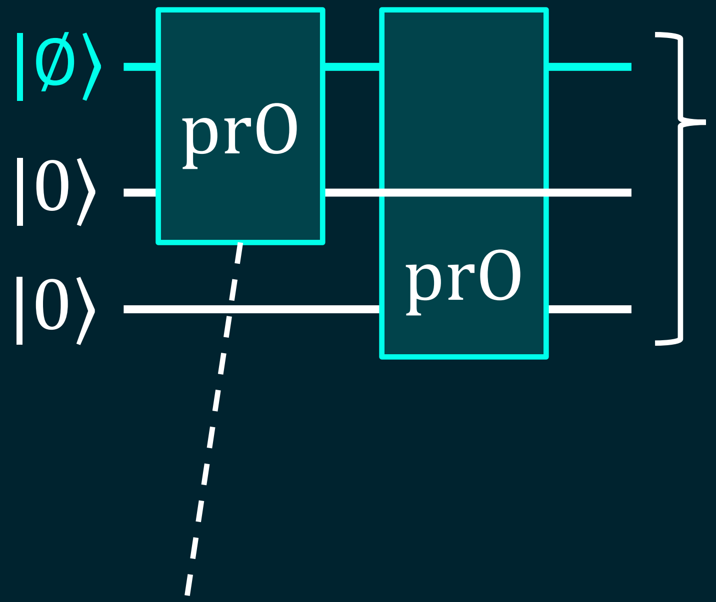
After tracing out: mixture of $|y_1, y_2\rangle + |y_2, y_1\rangle$ for random distinct y_1, y_2.

$$\mathrm{prO}\ |x\rangle|R\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Example 2: two queries on $|0\rangle$



Average over $U \leftarrow \text{Haar}$: $U|0\rangle \otimes U|0\rangle$ is *maximally mixed on the symmetric subspace*.



$\sum_{y_1 \neq y_2} |y_1, y_2\rangle \otimes \underbrace{|\{(0, y_1), (0, y_2)\}\rangle}_{\text{trace out/measure}}$

After tracing out: mixture of $|y_1, y_2\rangle + |y_2, y_1\rangle$ for random distinct y_1, y_2. This is almost *maximally mixed on the symmetric subspace*.

$$\mathrm{prO}\,|x\rangle|R\rangle = \sum_{y \notin R} |y\rangle|R \cup \{(x, y)\}\rangle$$

Up next:

Prove two claims simultaneously

Up next:

Prove two claims simultaneously

1) Our simulator works

Up next:

Prove two claims simultaneously

1) Our simulator works

2) “PFC” ensemble is a secure PRU.

Definition: $P \cdot F \cdot C$ ensemble
[MPSY24]

Definition: $P \cdot F \cdot C$ ensemble

[MPSY24]

$$P : |x\rangle \mapsto |\pi(x)\rangle$$

for random **permutation**

$$\pi : [N] \mapsto [N]$$

Definition: $P \cdot F \cdot C$ ensemble

[MPSY24]

$P: |x\rangle \mapsto |\pi(x)\rangle$

for random **permutation**

$\pi: [N] \mapsto [N]$

$F: |x\rangle \mapsto (-1)^{f(x)}|x\rangle$

for random **function**

$f: [N] \to \{0,1\}$

C sampled from a unitary 2-design $\mathfrak{D}$.

Definition: $P \cdot F \cdot C$ ensemble

[MPSY24]

$P: |x\rangle \mapsto |\pi(x)\rangle$

for random **permutation** $\pi: [N] \mapsto [N]$

$F: |x\rangle \mapsto (-1)^{f(x)}|x\rangle$

for random **function** $f: [N] \to \{0,1\}$

C sampled from a unitary 2-design $\mathfrak{D}$.

Example: $\mathfrak{D}$ = random Clifford

Definition: $P \cdot F \cdot C$ ensemble

[MPSY24]

$P: |x\rangle \mapsto |\pi(x)\rangle$

for random **permutation**

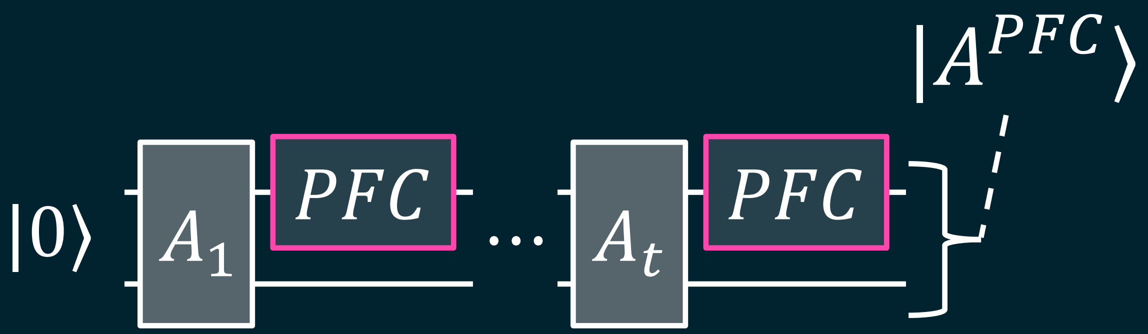
$\pi: [N] \mapsto [N]$

$F: |x\rangle \mapsto (-1)^{f(x)}|x\rangle$

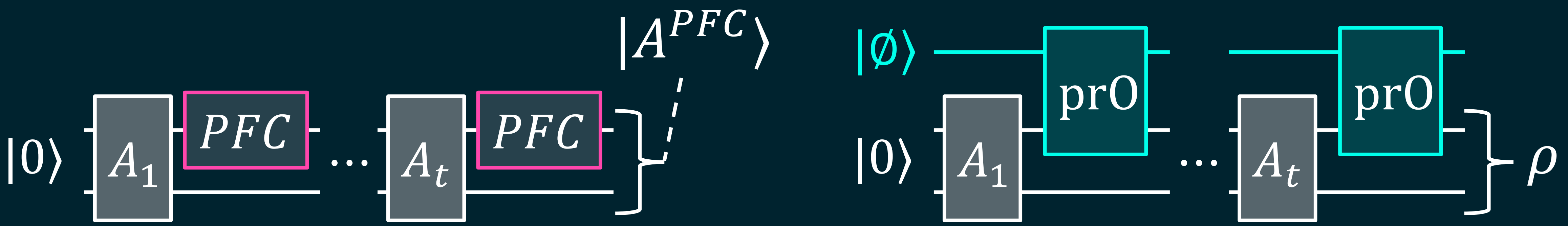
for random **function**

$f: [N] \to \{0,1\}$

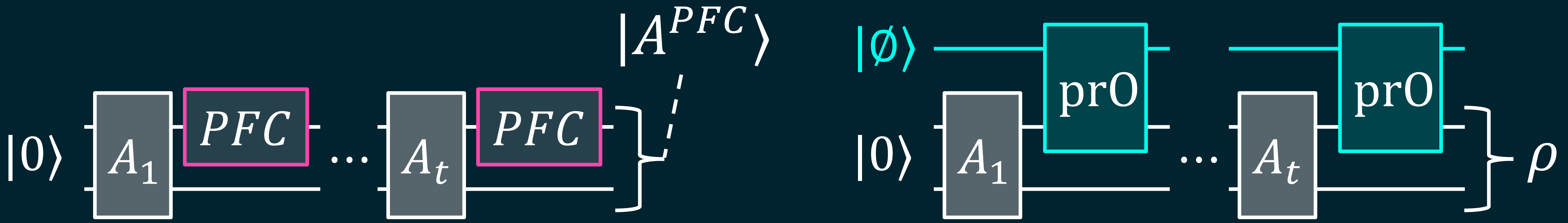
Claim: $\mathrm{pr0}$ simulates PFC.



Claim: pr0 simulates PFC.

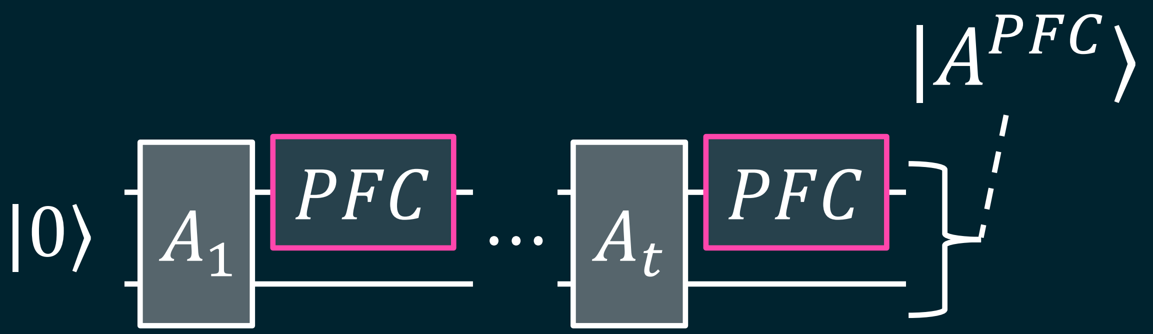


Claim: pr0 simulates PFC.



Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

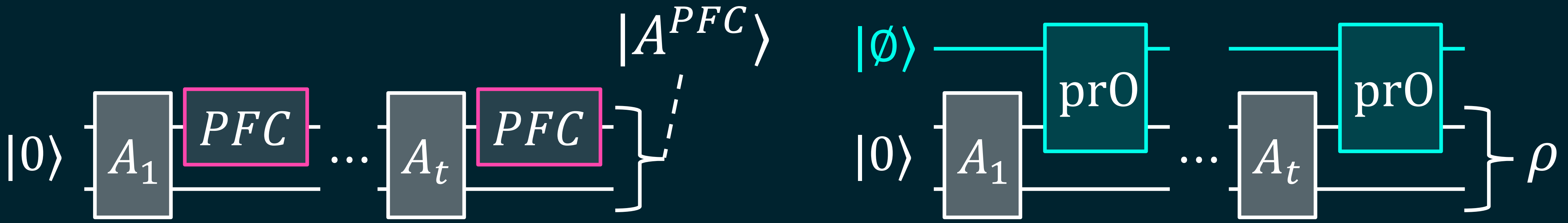
$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}} |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$



Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}} |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

ρ is independent of the choice of $\mathfrak{D}$!

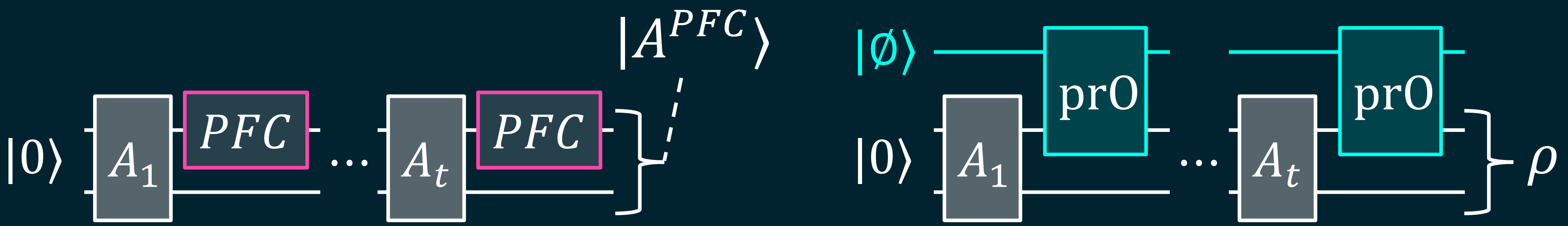


Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}}\ |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

ρ is independent of the choice of $\mathfrak{D}$!

Implies:

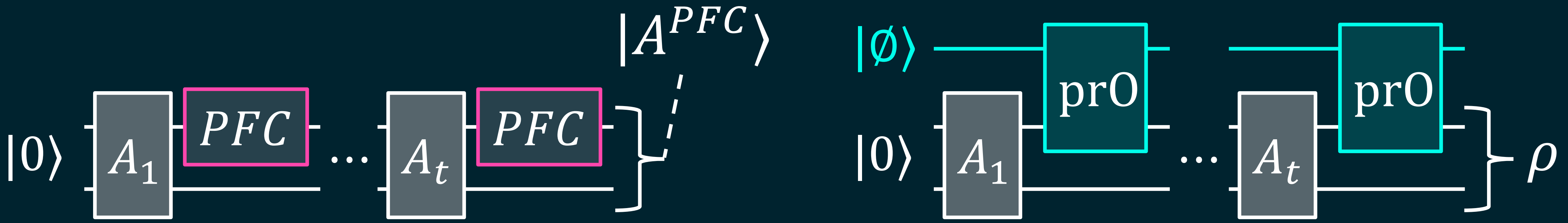


Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}}\, |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

ρ is independent of the choice of $\mathfrak{D}$!

Implies: 1) prO simulates Haar-random U ($\mathfrak{D} = \mathrm{Haar}$)



Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}}\,|A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

ρ is independent of the choice of $\mathfrak{D}$!

Implies: 1) prO simulates Haar-random U ($\mathfrak{D} = \mathrm{Haar}$)

2) PRUs exist ($\mathfrak{D} = \mathrm{Clifford}$, pseudorandom P, F)

Claim: prO simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C \leftarrow \mathfrak{D}} \, |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

Up next: proof of this claim

Claim: $\mathrm{pr0}$ simulates PFC. For ***any*** 2-design $\mathfrak{D}$,

$$\mathbf{TD}\left(\mathbb{E}_{P,F,C\leftarrow\mathfrak{D}} \, |A^{PFC}\rangle\langle A^{PFC}|, \rho\right) \leq \frac{t^2}{2^n}$$

Proof overview

Proof overview

Part 1: prO simulates $P \cdot F$ (random permutation and function) assuming A doesn't query on "bad" inputs.

Proof overview

Part 1: prO simulates $P \cdot F$ (random permutation and function) assuming A doesn't query on "bad" inputs.

For restricted algorithms A:

Proof overview

Part 1: prO simulates $P \cdot F$ (random permutation and function) assuming A doesn't query on "bad" inputs.

For restricted algorithms A:

Part 2: Insert a random C (sampled from any 2 design) to prevent A from querying on "bad" inputs.

Analyzing queries to PF

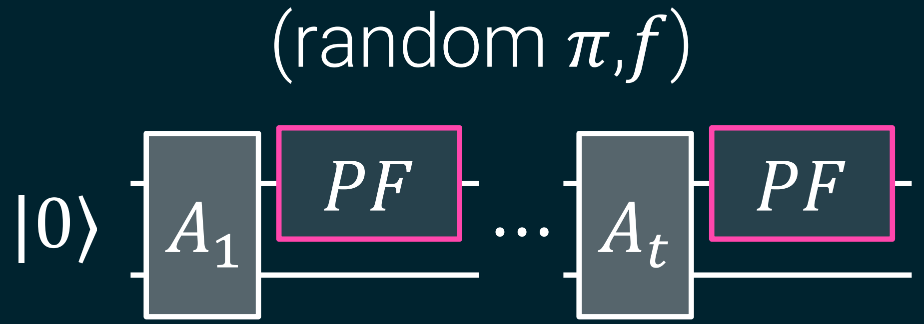
Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

standard implementation



Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

standard implementation

(random π, f)

$$|x\rangle \;-\boxed{PF}-\; (-1)^{f(x)}|\pi(x)\rangle$$

Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

standard implementation

(random π, f)

purify

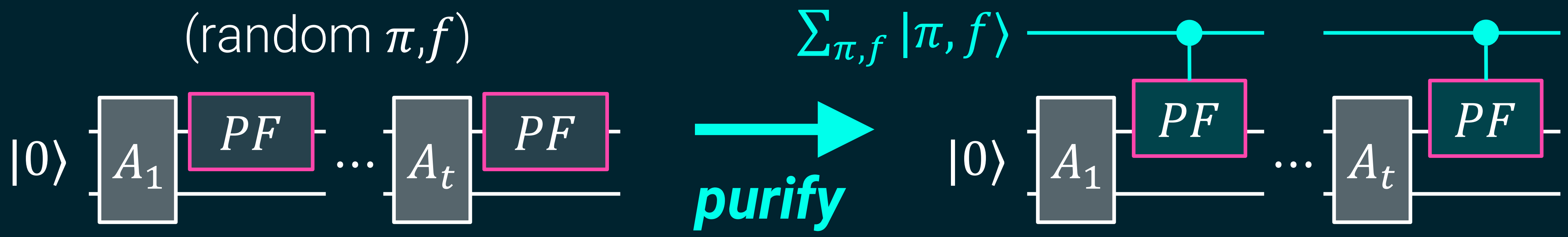
$$|x\rangle - \boxed{PF} - (-1)^{f(x)}|\pi(x)\rangle$$

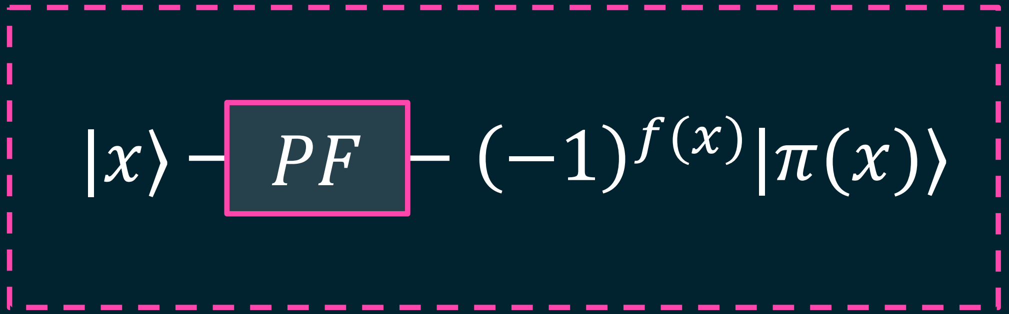
Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

standard implementation

purification

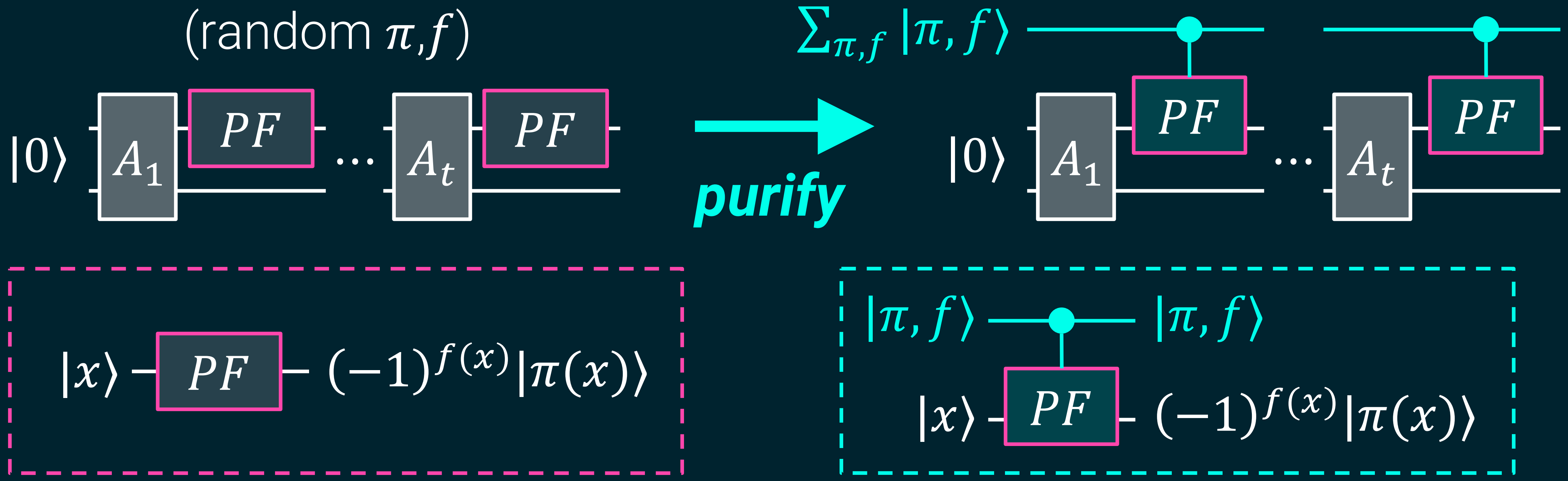




Analyzing queries to PF

Idea: ***purify*** the randomness of the permutation + function.

standard implementation | purification



In the PF-purification, each query corresponds to

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto (-1)^{f(x)} |\pi(x)\rangle \otimes |\pi, f\rangle$

In the PF-purification, each query corresponds to

$$\text{controlled-}PF\text{:}\quad |x\rangle \otimes |\pi, f\rangle \mapsto \underbrace{(-1)^{f(x)}|\pi(x)\rangle \otimes |\pi, f\rangle}$$

Rewrite the right-hand side:

In the PF-purification, each query corresponds to

$$\text{controlled-}PF\text{:} \quad |x\rangle \otimes |\pi, f\rangle \mapsto \underbrace{(-1)^{f(x)}|\pi(x)\rangle \otimes |\pi, f\rangle}$$

Rewrite the right-hand side:

(1) Plug in
$|\pi(x)\rangle = \sum_y \delta_{\pi(x)=y} |y\rangle$.

In the PF-purification, each query corresponds to

$$\text{controlled-}PF: \quad |x\rangle \otimes |\pi, f\rangle \mapsto (-1)^{f(x)}|\pi(x)\rangle \otimes |\pi, f\rangle$$

Rewrite the right-hand side:

(1) Plug in
$|\pi(x)\rangle = \sum_y \delta_{\pi(x)=y}|y\rangle$.

$$(-1)^{f(x)} \sum_y \delta_{\pi(x)=y}|y\rangle \otimes |\pi, f\rangle$$

In the PF-purification, each query corresponds to

$$\text{controlled-}PF\text{:} \quad |x\rangle \otimes |\pi, f\rangle \mapsto (-1)^{f(x)}|\pi(x)\rangle \otimes |\pi, f\rangle$$

Rewrite the right-hand side:

$$(-1)^{f(x)} \sum_y \delta_{\pi(x)=y} |y\rangle \otimes |\pi, f\rangle$$

(1) Plug in
$|\pi(x)\rangle = \sum_y \delta_{\pi(x)=y} |y\rangle$.

(2) Rearrange coefficients:

In the PF-purification, each query corresponds to

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto (-1)^{f(x)}|\pi(x)\rangle \otimes |\pi, f\rangle$

Rewrite the right-hand side:

(1) Plug in $|\pi(x)\rangle = \sum_y \delta_{\pi(x)=y}|y\rangle$.

$$(-1)^{f(x)} \sum_y \delta_{\pi(x)=y}|y\rangle \otimes |\pi, f\rangle$$

(2) Rearrange coefficients:

$$\sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y}\,|\pi, f\rangle$$

$$\boxed{\sum_{y} |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y}\, |\pi, f\rangle}$$

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle$

$$\boxed{\sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle}$$

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle$

path-recording oracle $\mathbf{prO}$: $|x\rangle \otimes |R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |R \cup \{(x, y)\}\rangle$

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} |\pi, f\rangle$

path-recording oracle prO: $|x\rangle \otimes |R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |R \cup \{(x,y)\}\rangle$

Intuition: prO creates a superposition over y and simultaneously "records" (x, y)

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} |\pi, f\rangle$

path-recording oracle $\mathbf{prO}$: $|x\rangle \otimes |R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |R \cup \{(x, y)\}\rangle$

Intuition: $\mathbf{prO}$ creates a superposition over y and simultaneously "records" (x, y)

We'll show that controlled-PF does (almost) the same thing.

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_{y} |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle$

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} |\pi, f\rangle$

Initial state: purifying register begins as $\sum_{\pi,f} |\pi, f\rangle$.

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle$

Initial state: purifying register begins as $\sum_{\pi,f} |\pi, f\rangle$.

After 1 query: purifying register is a superposition of

$$\sum_{\pi,f} (-1)^{f(x)} \cdot \delta_{\pi(x)=y} \, |\pi, f\rangle$$

controlled-PF: $|x\rangle \otimes |\pi, f\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{f(x)} \cdot \delta_{\pi(x)=y} |\pi, f\rangle$

Initial state: purifying register begins as $\sum_{\pi,f} |\pi, f\rangle$.

After 1 query: purifying register is a superposition of

$$\sum_{\pi,f} (-1)^{f(x)} \cdot \delta_{\pi(x)=y} |\pi, f\rangle$$

After t queries: purifying register is a superposition of

$$\sum_{\pi,f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} |\pi, f\rangle$$

Definition: for $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$, let

$$|\mathrm{pf}_R\rangle := \sum_{\pi, f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} |\pi, f\rangle$$

After t queries: purifying register is a superposition of

$$\sum_{\pi, f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} |\pi, f\rangle$$

Definition: for $R = \{(x_1, y_1), \dots, (x_t, y_t)\}$, let

$$|\mathrm{pf}_R\rangle := \sum_{\pi, f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} \, |\pi, f\rangle$$

controlled-PF: $|x\rangle \otimes |\mathrm{pf}_R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes \left|\mathrm{pf}_{R\cup\{(x,y)\}}\right\rangle$

Definition: for $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$, let

$$|\mathrm{pf}_R\rangle := \sum_{\pi,f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} \, |\pi, f\rangle$$

controlled-PF: $\quad |x\rangle \otimes |\mathrm{pf}_R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes \left|\mathrm{pf}_{R \cup \{(x,y)\}}\right\rangle$

prO: $\quad |x\rangle \otimes |R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |R \cup \{(x, y)\}\rangle$

Definition: for $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$, let

$$|\mathrm{pf}_R\rangle \coloneqq \sum_{\pi,f} (-1)^{f(x_1)+\cdots+f(x_t)} \cdot \delta_{\pi(x_1)=y_1} \cdots \delta_{\pi(x_t)=y_t} \, |\pi, f\rangle$$

controlled-PF: $\quad |x\rangle \otimes |\mathrm{pf}_R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |\mathrm{pf}_{R \cup \{(x,y)\}}\rangle$

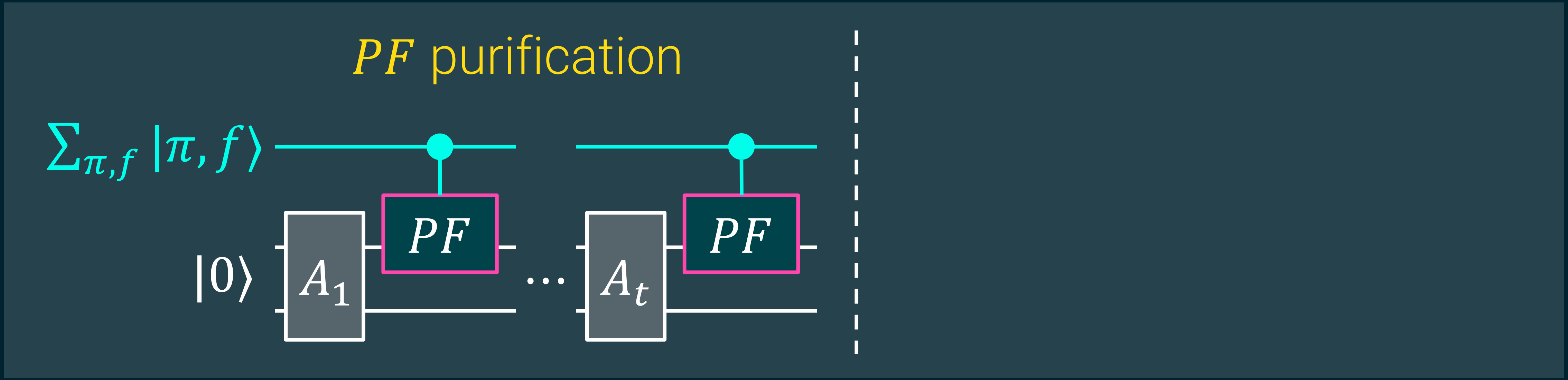
prO: $\quad |x\rangle \otimes |R\rangle \mapsto \sum_{y \notin R} |y\rangle \otimes |R \cup \{(x, y)\}\rangle$

Claim: $|\mathrm{pf}_R\rangle$ are orthogonal for R s.t. $x_1, \ldots, x_t$ are distinct.

So these are equivalent from the algorithm's point of view:

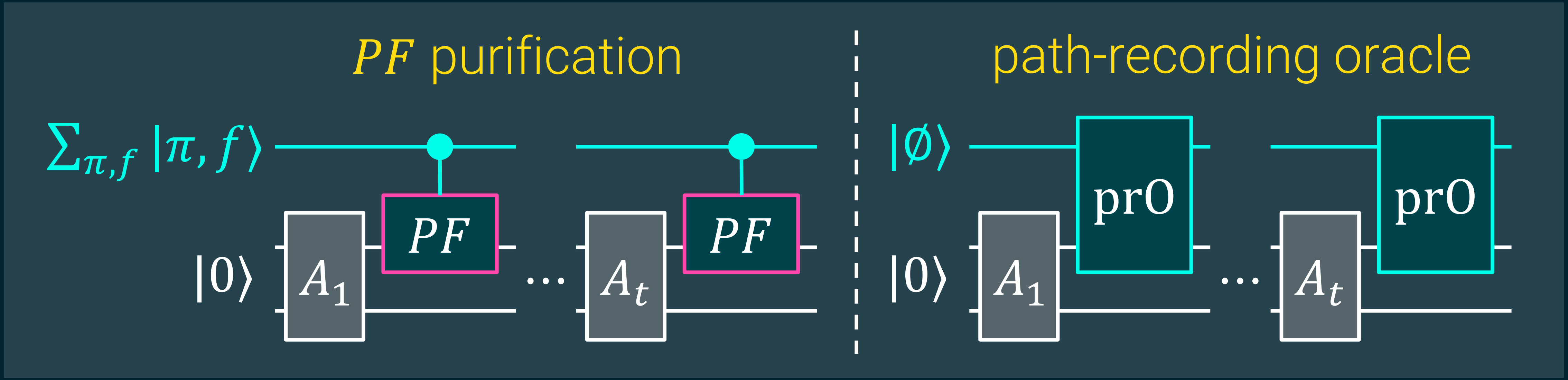
Claim: $|\mathrm{pf}_R\rangle$ are orthogonal for R s.t. $x_1, \ldots, x_t$ are distinct.

So these are equivalent from the algorithm's point of view:



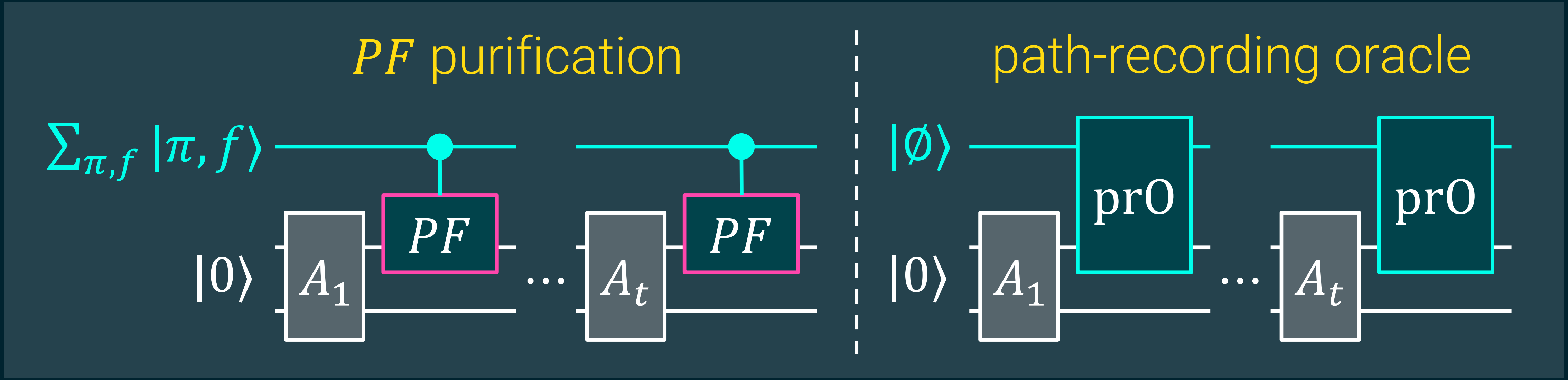
Claim: $|\mathrm{pf}_R\rangle$ are orthogonal for R s.t. $x_1, \dots, x_t$ are distinct.

So these are equivalent from the algorithm's point of view:



Claim: $|\mathrm{pf}_R\rangle$ are orthogonal for R s.t. $x_1, \ldots, x_t$ are distinct.

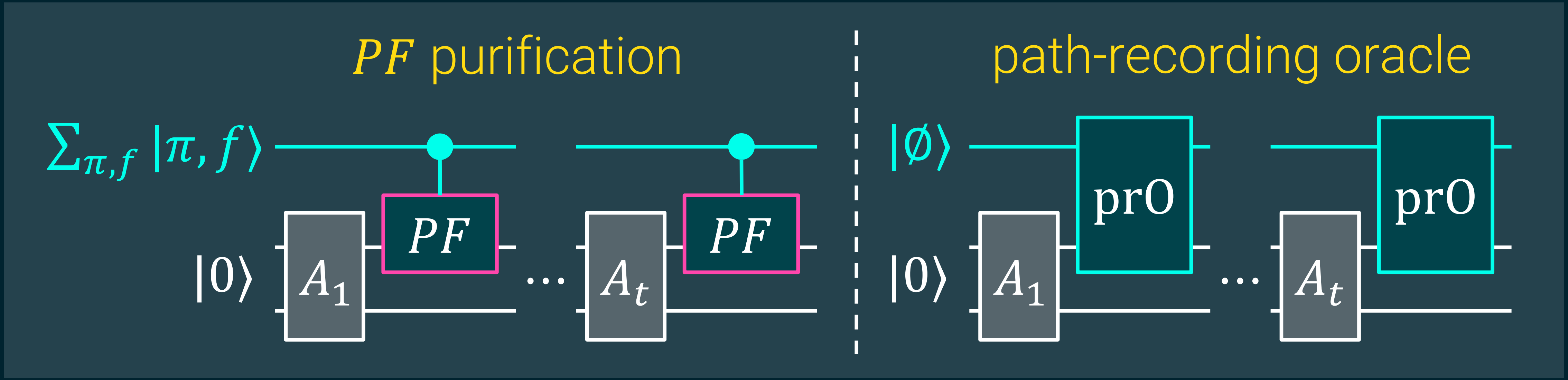
So these are equivalent from the algorithm's point of view:



...unless $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ has **colliding** x_i's.

Claim: $|\mathrm{pf}_R\rangle$ are orthogonal for R s.t. $x_1, \ldots, x_t$ are distinct.

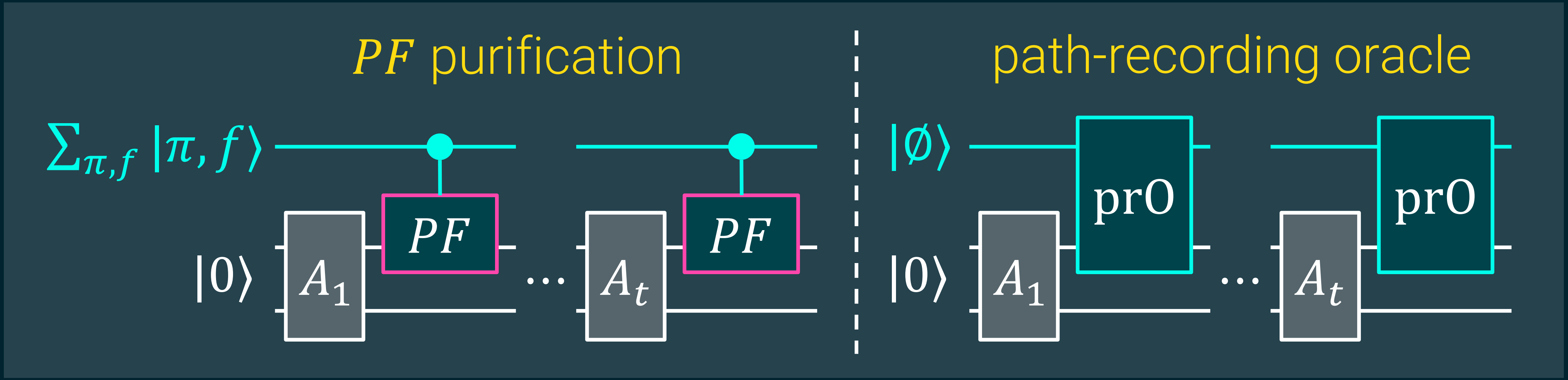
So these are equivalent from the algorithm's point of view:



...unless $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ has **colliding** x_i's.

This is where the 2-design comes in!

So these are equivalent from the algorithm's point of view:



...unless $R = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ has **colliding** x_i's.

This is where the 2-design comes in!

Claim: inserting C **before** each query prevents collisions.

Recap

Recap

1) We defined the path-recording oracle:

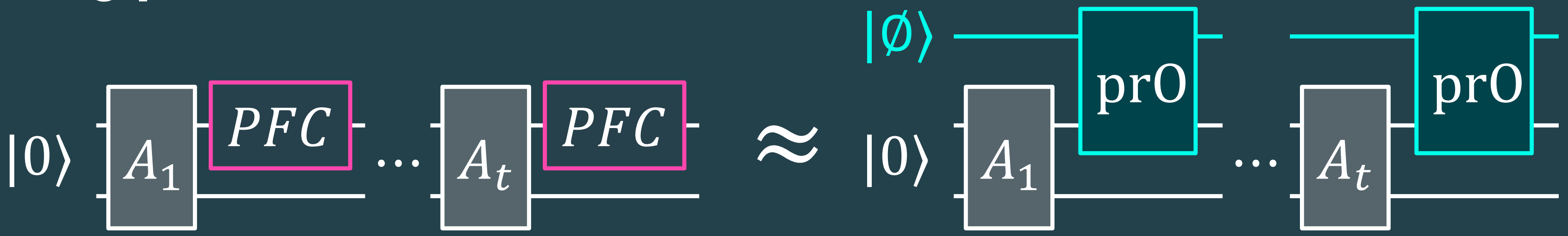
$$\mathrm{prO}\colon |x\rangle|R\rangle \mapsto \sum_{y\notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

Recap

1) We defined the path-recording oracle:

$$\mathrm{prO}\colon |x\rangle|R\rangle \mapsto \sum_{y \notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

2) Using purification, we showed:

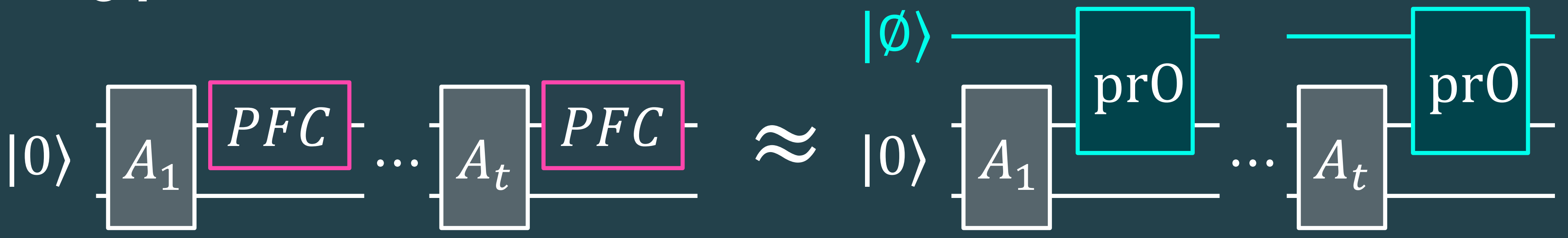


Recap

1) We defined the path-recording oracle:

$$\mathrm{prO}\colon |x\rangle|R\rangle \mapsto \sum_{y \notin R} |y\rangle|R \cup \{(x,y)\}\rangle$$

2) Using purification, we showed:



Consequence: prO simulates Haar-random unitaries + PRU exist

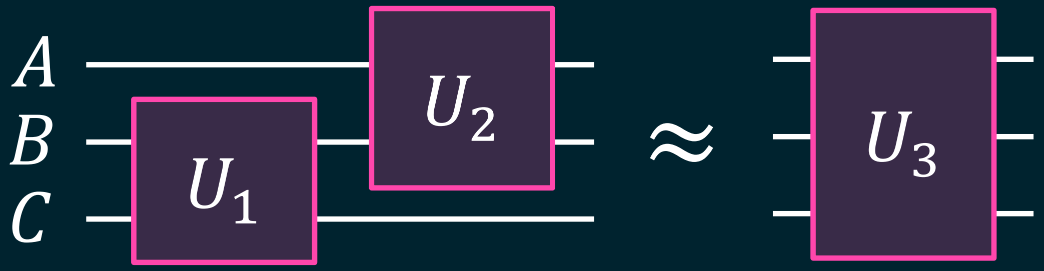
The path-recording oracle gives us a new way to study random unitaries.

The path-recording oracle gives us a new way to study random unitaries.

Let’s see an example.

Application: a simpler proof of the “gluing” lemma

Gluing lemma [SSH24]:

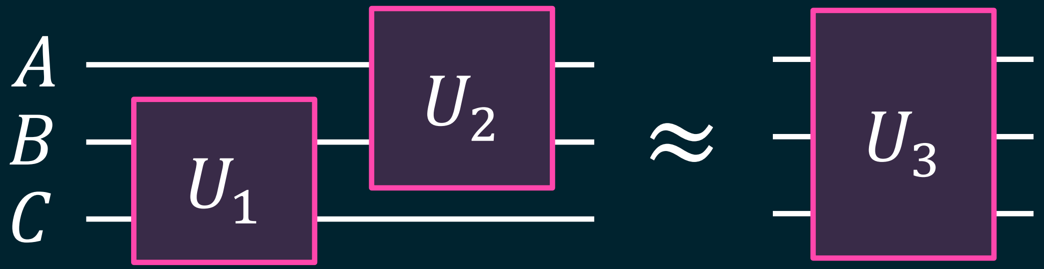


Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

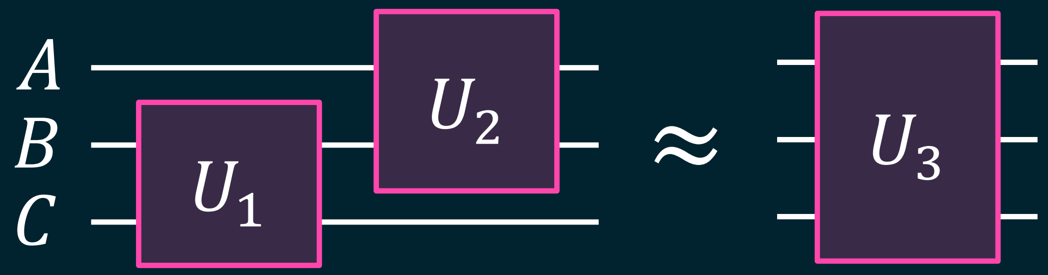


Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$



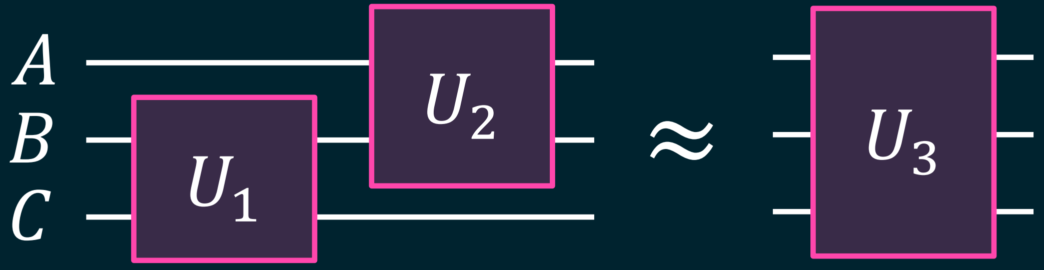
Proof via path-recording:

Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$



Proof via path-recording:

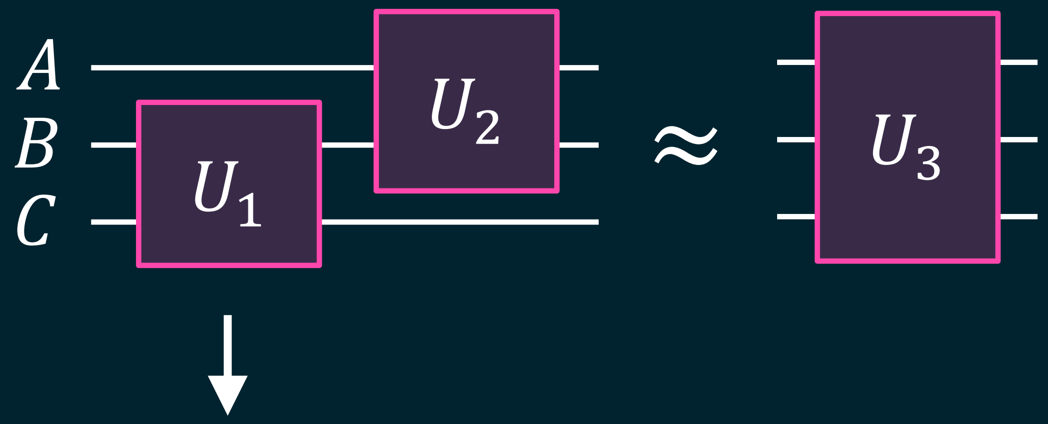
(1) replace each U_i with prO_i

Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

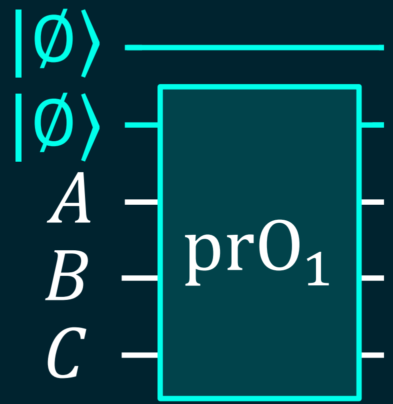
If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$



Proof via path-recording:

(1) replace each U_i with prO_i



Application: a simpler proof of the "gluing" lemma

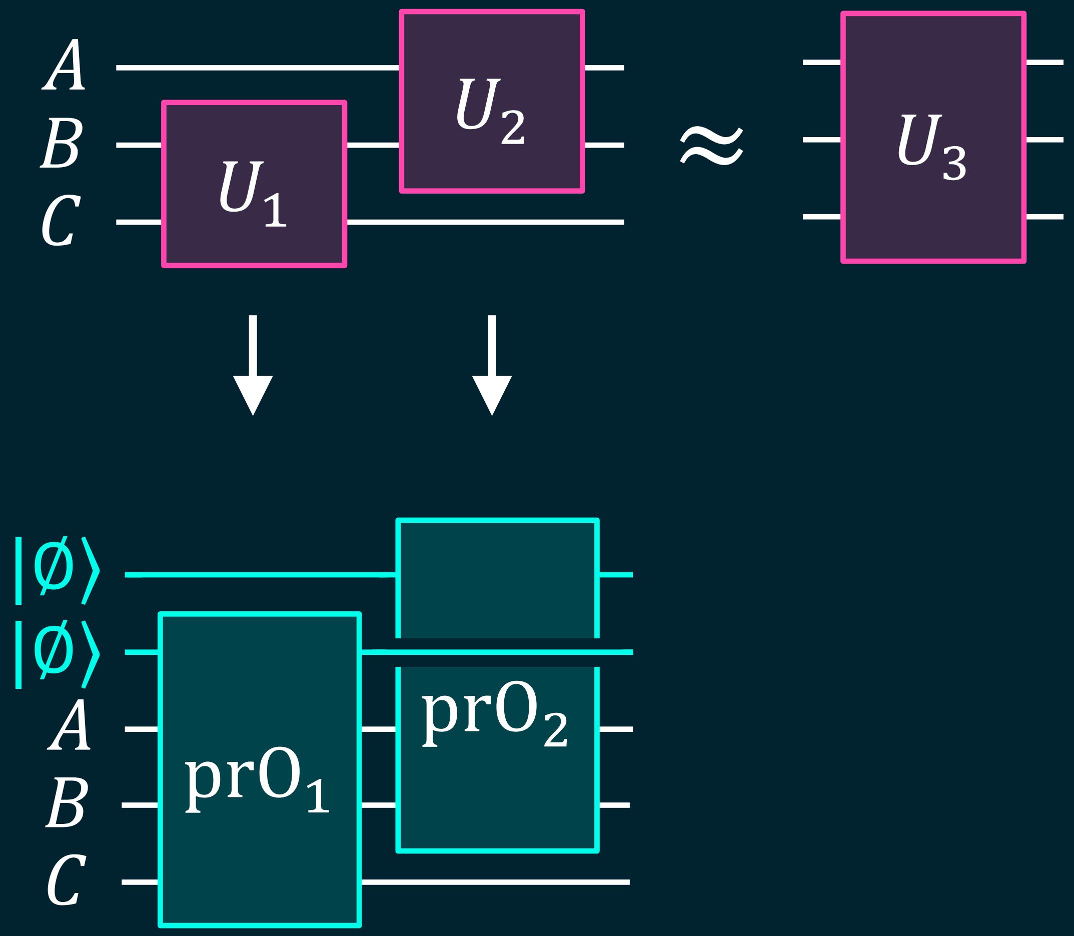
Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

Proof via path-recording:

(1) replace each U_i with prO_i



Application: a simpler proof of the “gluing” lemma

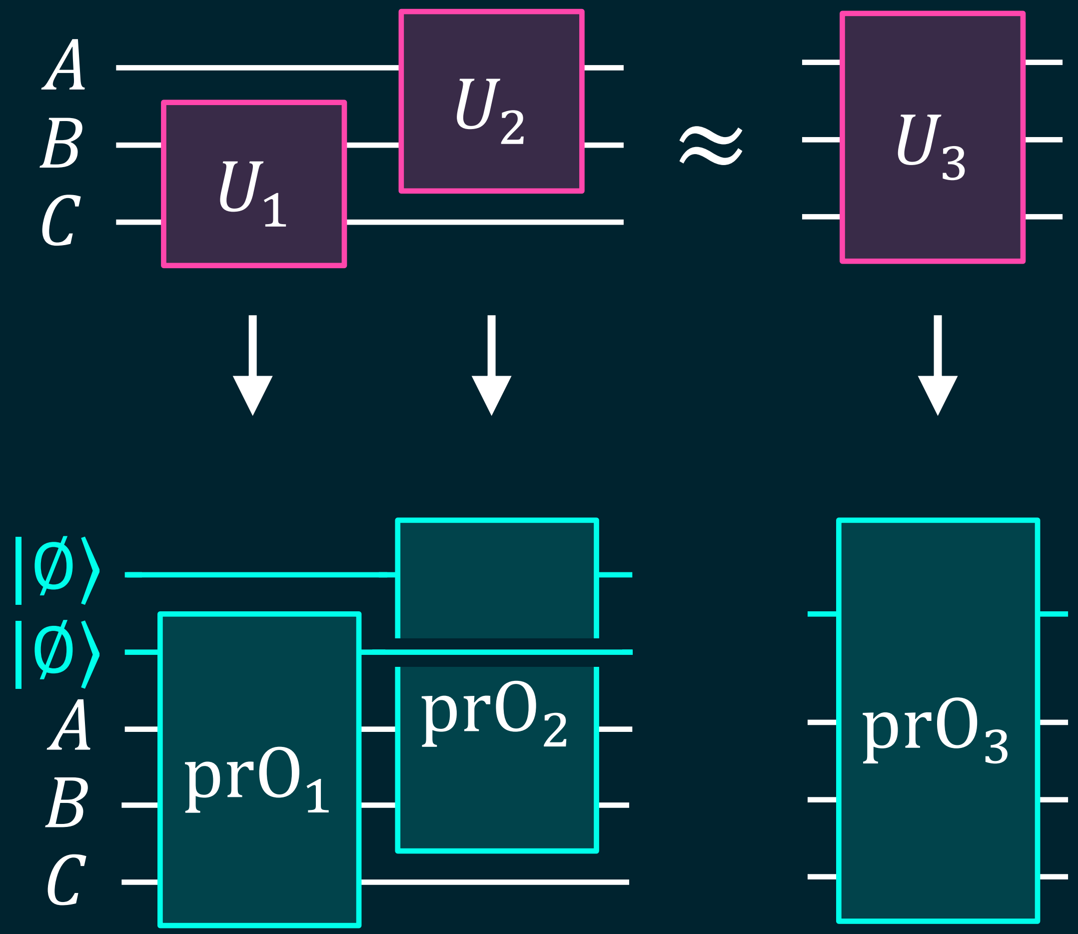
Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

Proof via path-recording:

(1) replace each U_i with prO_i



Application: a simpler proof of the "gluing" lemma

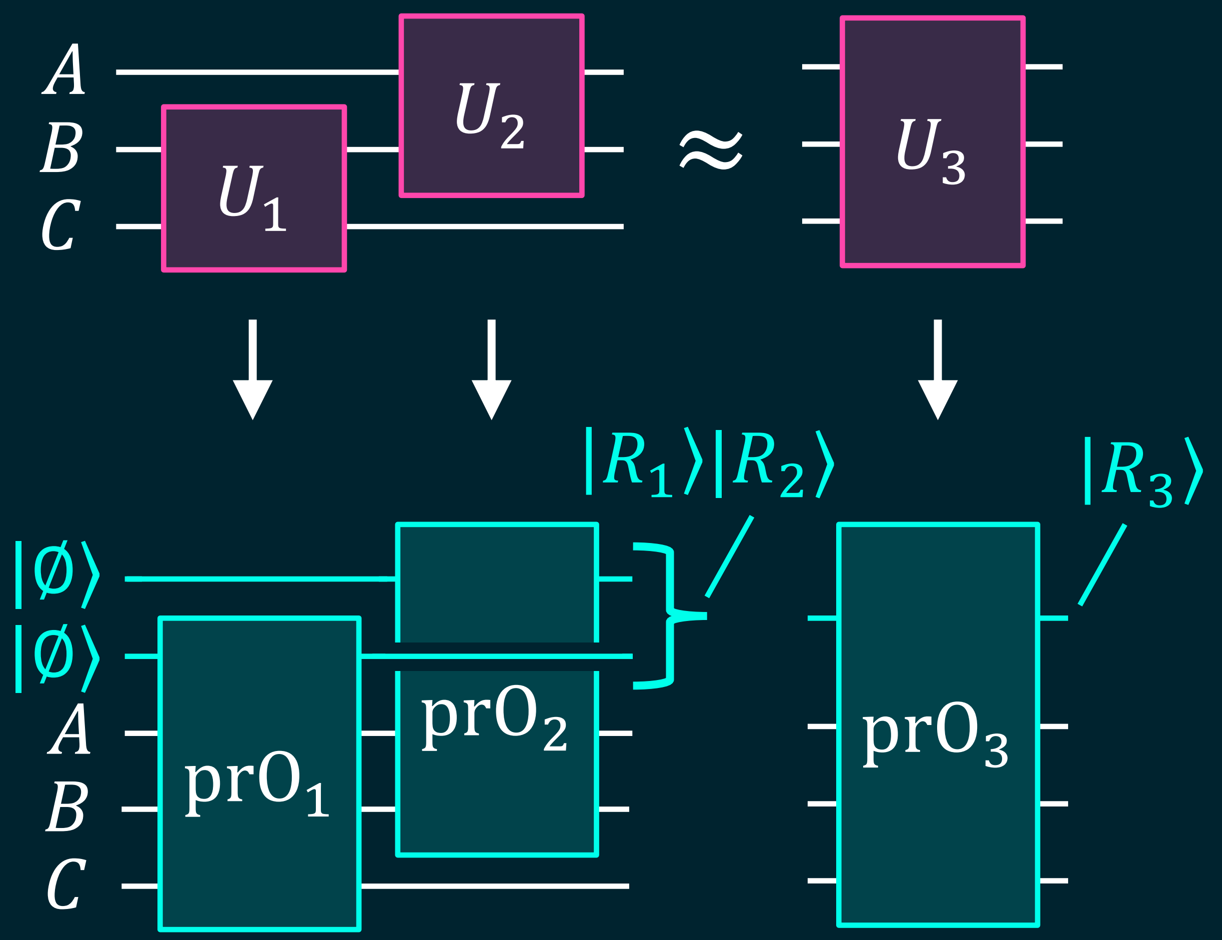
Gluing lemma [SSH24]:

If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

Proof via path-recording:

(1) replace each U_i with prO_i



Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

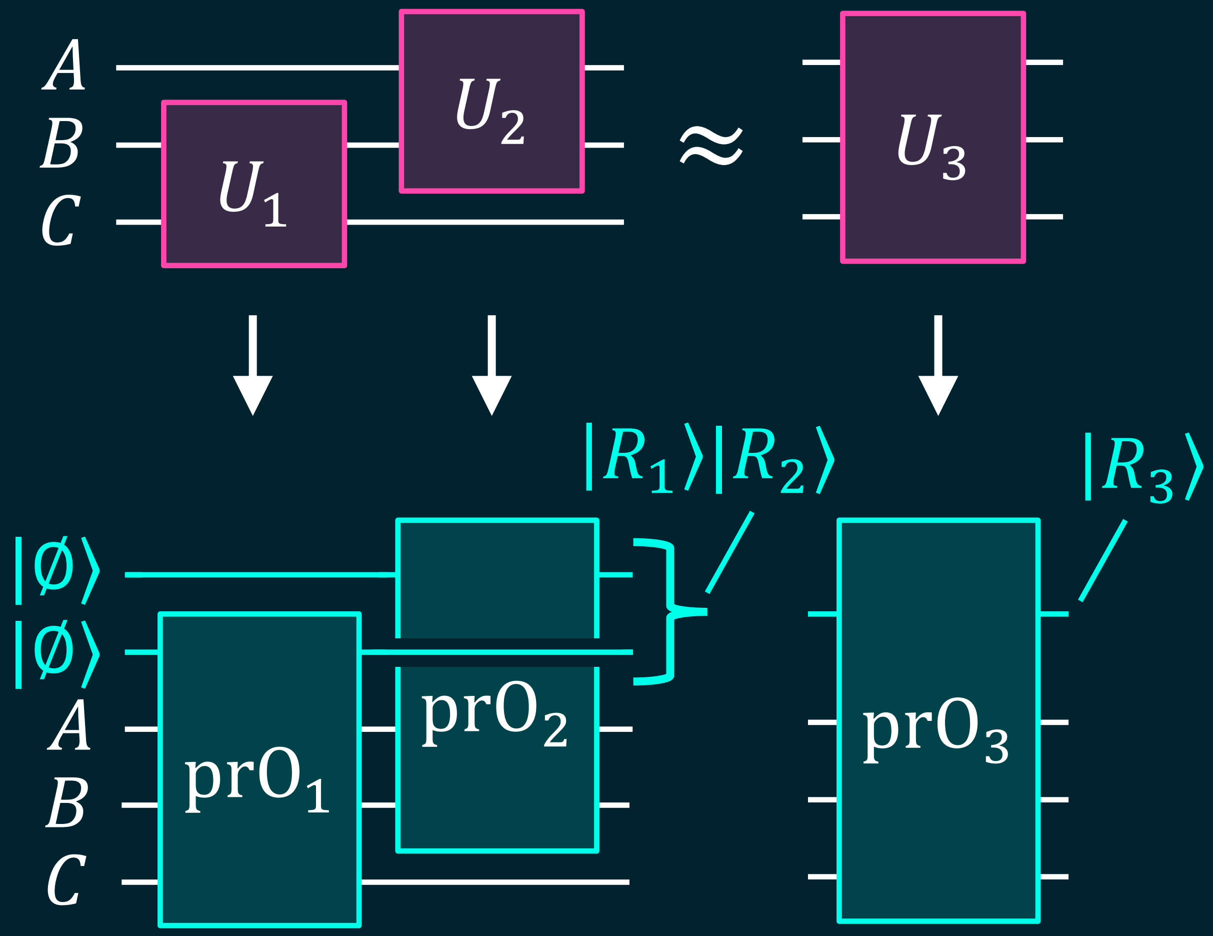
If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

Proof via path-recording:

(1) replace each U_i with prO_i

(2) construct isometry Glue that maps $|R_1\rangle|R_2\rangle$ to $|R_3\rangle$



Application: a simpler proof of the "gluing" lemma

Gluing lemma [SSH24]:

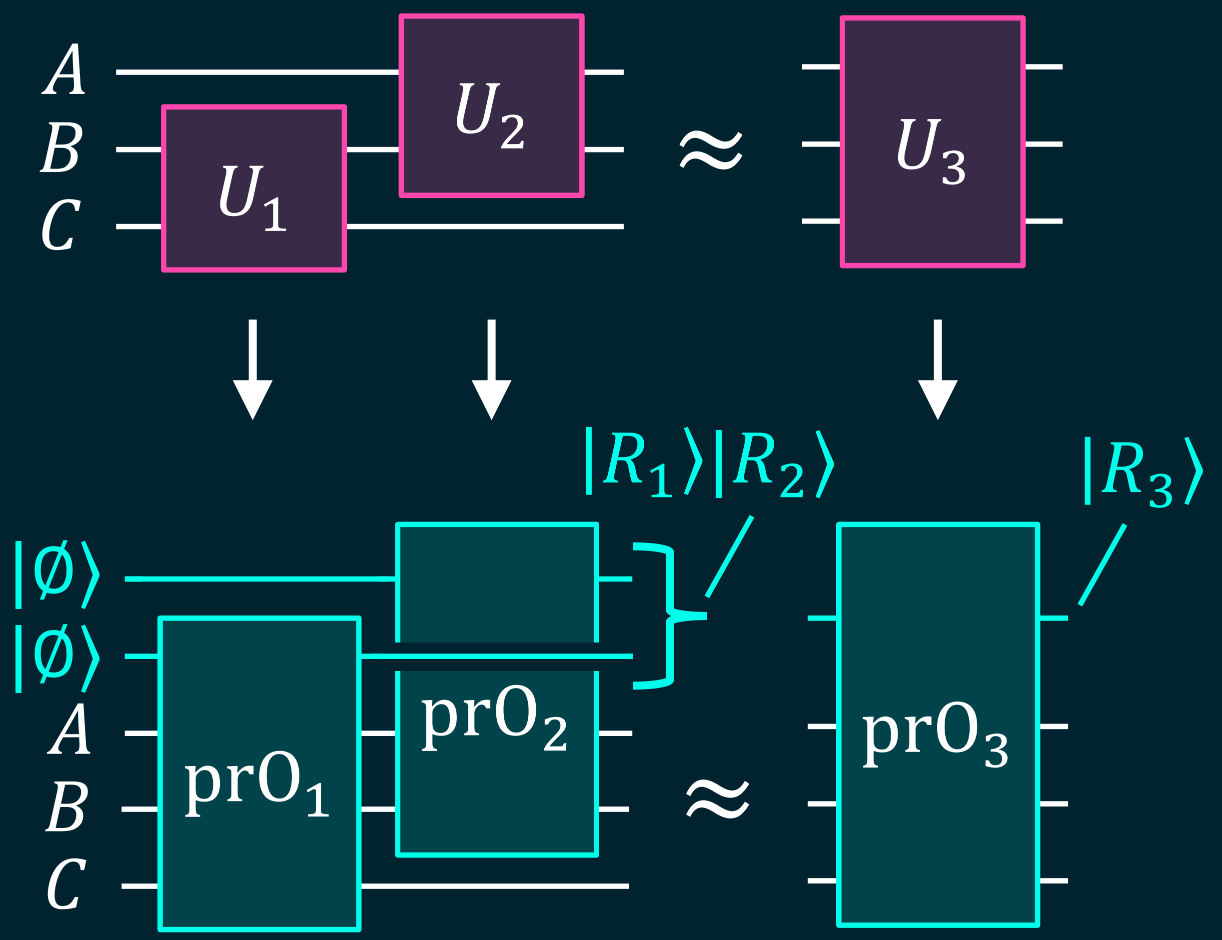
If U_1 and U_2 overlap on $|B| = \omega(\log n)$ qubits, then

$$U_2 \cdot U_1 \approx U_3.$$

Proof via path-recording:

(1) replace each U_i with prO_i

(2) construct isometry Glue that maps $|R_1\rangle|R_2\rangle$ to $|R_3\rangle$



What about inverse queries?

What about inverse queries?

Today, we considered algorithms that query U **but not $U^\dagger$**.

What about inverse queries?

Today, we considered algorithms that query U **but not** $U^\dagger$.

In the paper:

- "symmetrized" $\mathbf{prO}$ that simulates forward **+ inverse** queries

What about inverse queries?

Today, we considered algorithms that query U **but not $U^\dagger$**.

In the paper:

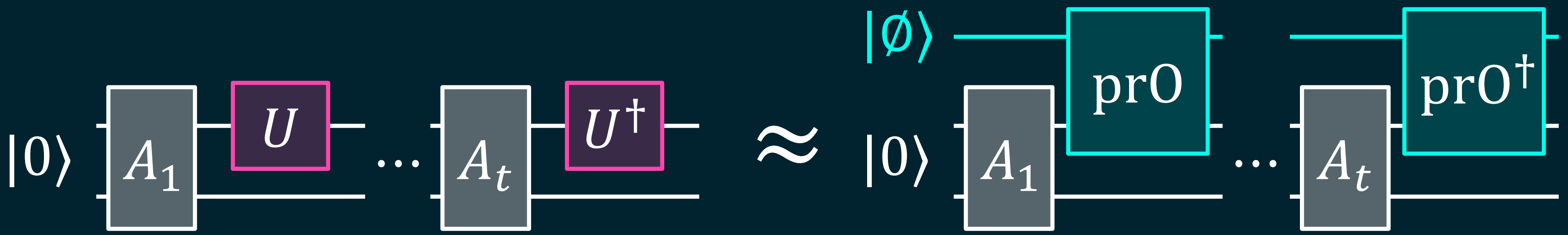
- “symmetrized” $\mathbf{prO}$ that simulates forward **+ inverse** queries
- PRUs secure against forward + inverse queries

What about inverse queries?

Today, we considered algorithms that query U **but not $U^\dagger$**.

In the paper:

- "symmetrized" $\mathbf{prO}$ that simulates forward **+ inverse** queries
- PRUs secure against forward + inverse queries

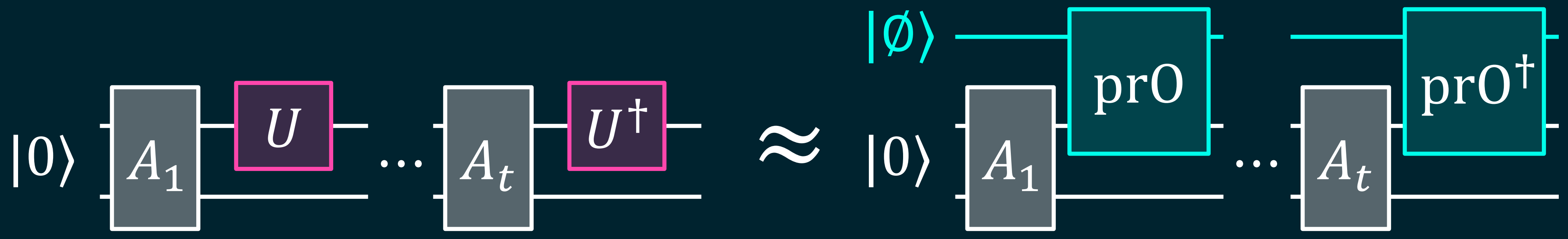


What about inverse queries?

Today, we considered algorithms that query U **but not $U^\dagger$**.

In the paper:

- "symmetrized" prO that simulates forward **+ inverse** queries
- PRUs secure against forward + inverse queries



Challenge: recording the path isn't enough; also need to **erase**!

Future directions

Future directions

(physics) do we have the right notions of pseudorandomness?

Future directions

(physics) do we have the right notions of pseudorandomness?

(physics) physical interpretation of the path-recording oracle prO?

Future directions

(physics) do we have the right notions of pseudorandomness?

(physics) physical interpretation of the path-recording oracle $\mathbf{prO}$?

(math) consequences for random matrix theory or rep theory?

Future directions

(physics) do we have the right notions of pseudorandomness?

(physics) physical interpretation of the path-recording oracle $\mathbf{prO}$?

(math) consequences for random matrix theory or rep theory?

(quantum computing) use path-recording to study quantum algs?

Future directions

(physics) do we have the right notions of pseudorandomness?

(physics) physical interpretation of the path-recording oracle $\mathbf{prO}$?

(math) consequences for random matrix theory or rep theory?

(quantum computing) use path-recording to study quantum algs?

(crypto) PRUs without one-way functions? other applications?

Future directions

(physics) do we have the right notions of pseudorandomness?

(physics) physical interpretation of the path-recording oracle prO?

(math) consequences for random matrix theory or rep theory?

(quantum computing) use path-recording to study quantum algs?

(crypto) PRUs without one-way functions? other applications?

Thanks for listening!

arXiv: 2410.10116