How to Construct Random Unitaries

Fermi Ma
Berkeley → NYU

joint work with Hsin-Yuan Huang

Haar measure: uniform distribution on unitaries

Haar measure: uniform distribution on unitaries

Property: for any unitary W, if $U \sim \text{Haar}$, $W \cdot U \sim \text{Haar}$

Haar measure: uniform distribution on unitaries

Property: for any unitary W, if $U \sim$ Haar, $W \cdot U \sim$ Haar

Haar-random unitaries show up everywhere:

Haar measure: uniform distribution on unitaries

Property: for any unitary W, if $U \sim$ Haar, $W \cdot U \sim$ Haar

Haar-random unitaries show up everywhere:

- black hole information scrambling
- entanglement
- quantum learning algorithms
- quantum crypto
- random quantum circuits
- unitary complexity
- quantum error correction

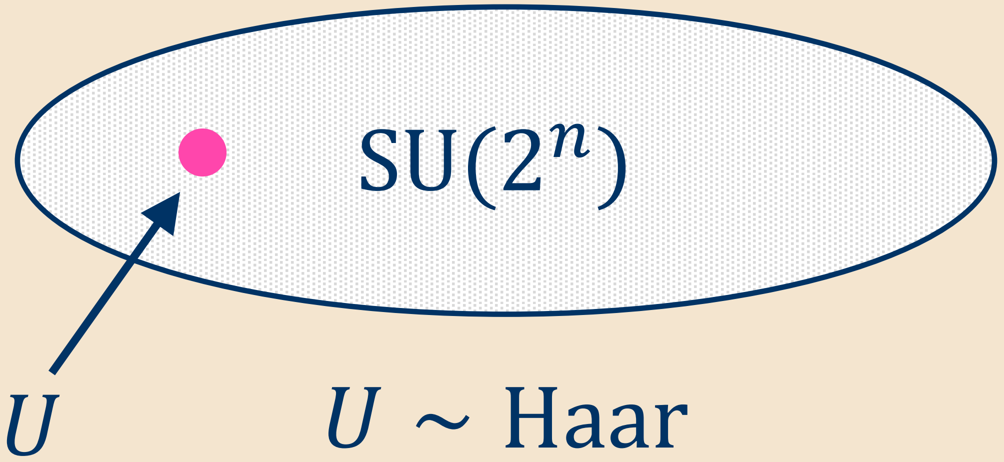
…

Challenge:

Haar-random unitaries are exponentially complex

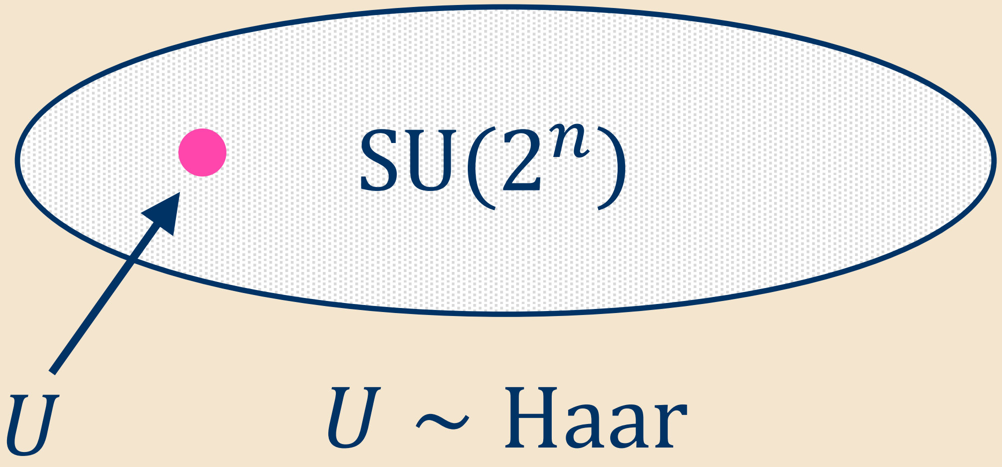
Challenge:

Haar-random unitaries are exponentially complex



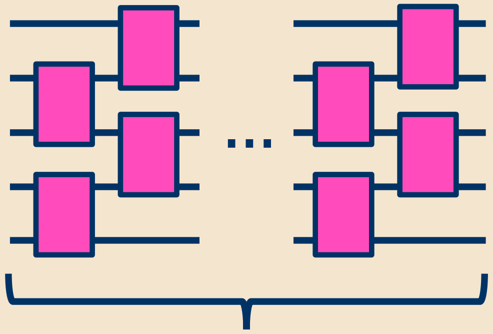
Challenge:

Haar-random unitaries are exponentially complex



$U \sim \text{Haar}$

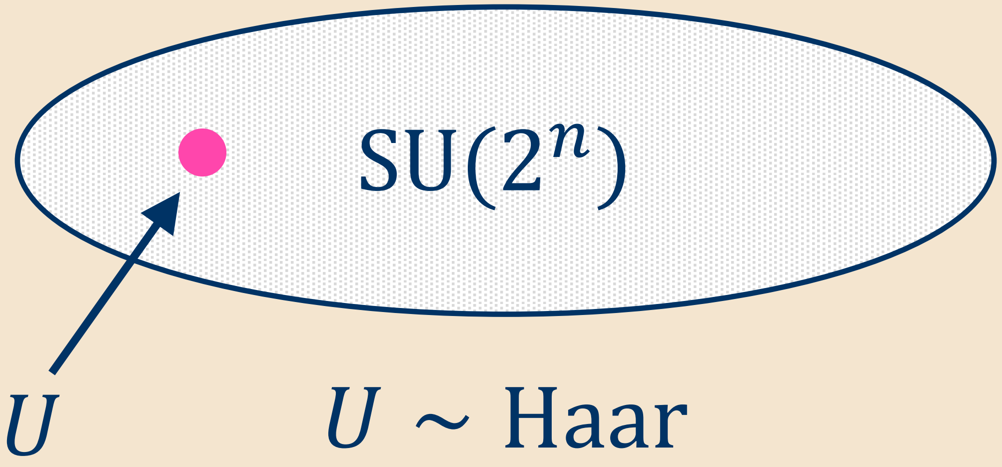
minimal circuit for U



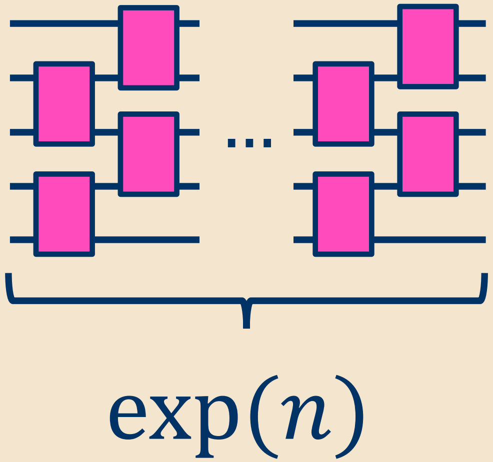
$\exp(n)$

Challenge:

Haar-random unitaries are exponentially complex



minimal circuit for U



This makes them impractical for most applications!

There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

1	$f(1)$
2	$f(2)$
$\vdots$	$\vdots$
2^n	$f(2^n)$

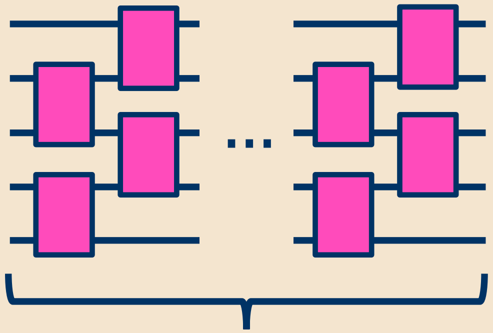
random f

There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

1	$f(1)$
2	$f(2)$
⋮	⋮
2^n	$f(2^n)$

random f

minimal circuit for f



$\exp(n)$

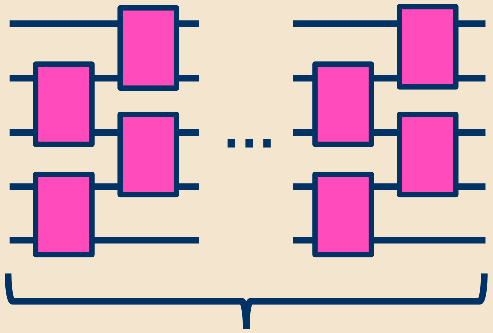
There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

So in practice, we use **pseudorandom functions (PRFs)**.

1	$f(1)$
2	$f(2)$
$\vdots$	$\vdots$
2^n	$f(2^n)$

random f

minimal circuit for f



$\exp(n)$

There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

So in practice, we use **pseudorandom functions (PRFs)**.

1	$f(1)$
2	$f(2)$
$\vdots$	$\vdots$
2^n	$f(2^n)$

random f

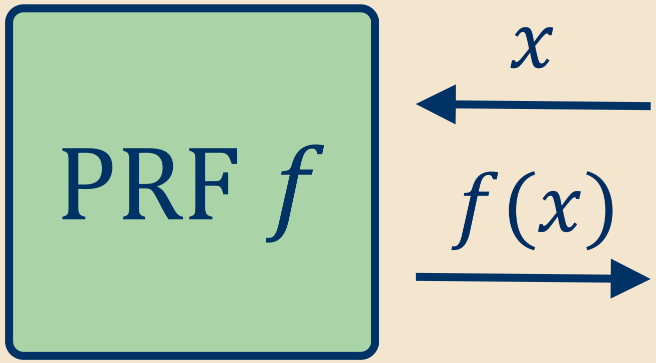
PRF f

There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

So in practice, we use **pseudorandom functions (PRFs)**.

1	$f(1)$
2	$f(2)$
$\vdots$	$\vdots$
2^n	$f(2^n)$

random f

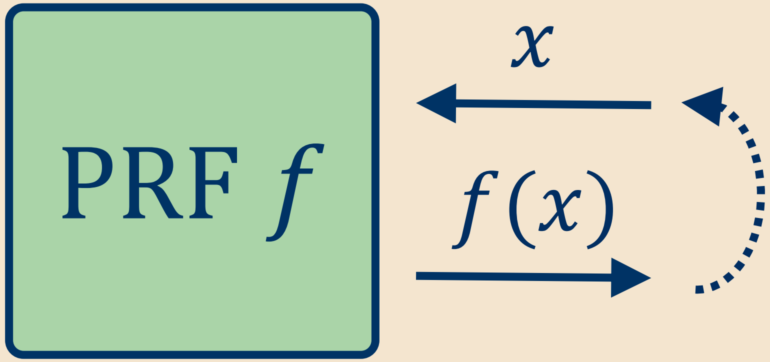


There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

So in practice, we use **pseudorandom functions (PRFs)**.

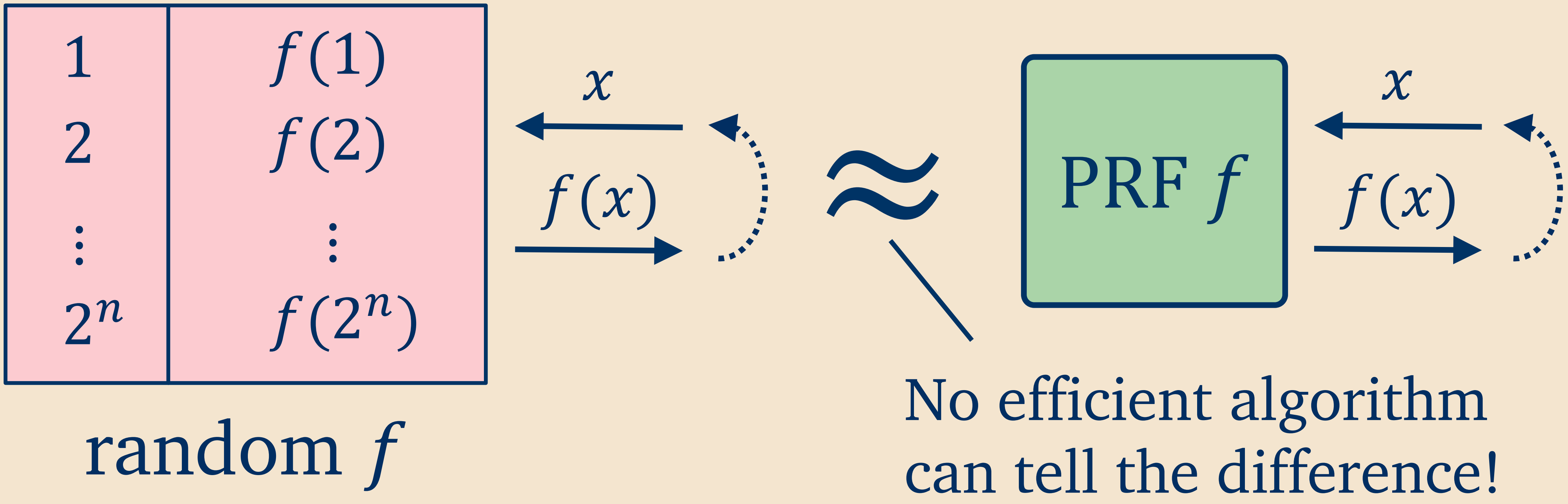
1	$f(1)$
2	$f(2)$
$\vdots$	$\vdots$
2^n	$f(2^n)$

random f



There's an analogous "problem" for functions: a random function on n bits is exponentially complex!

So in practice, we use **pseudorandom functions (PRFs)**.

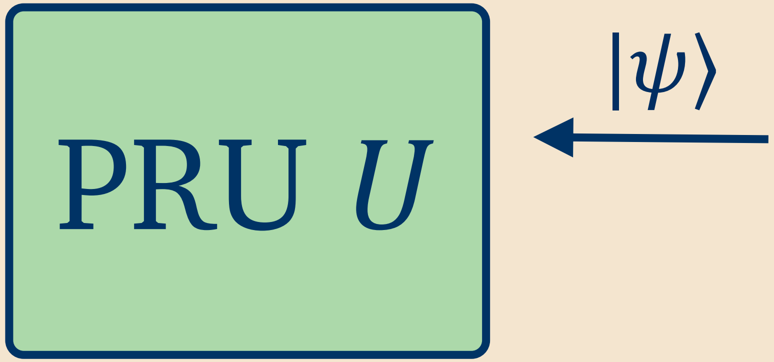


Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random

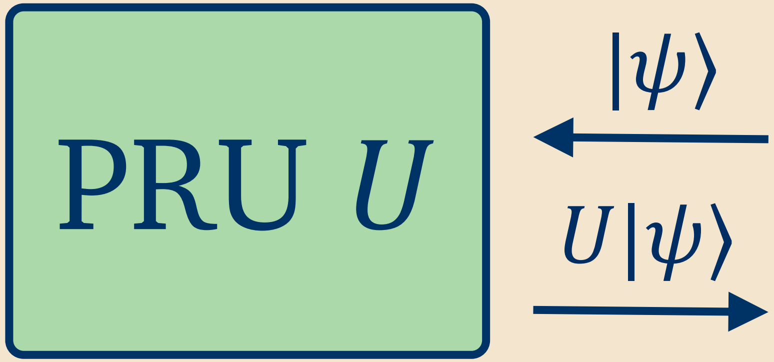
Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random



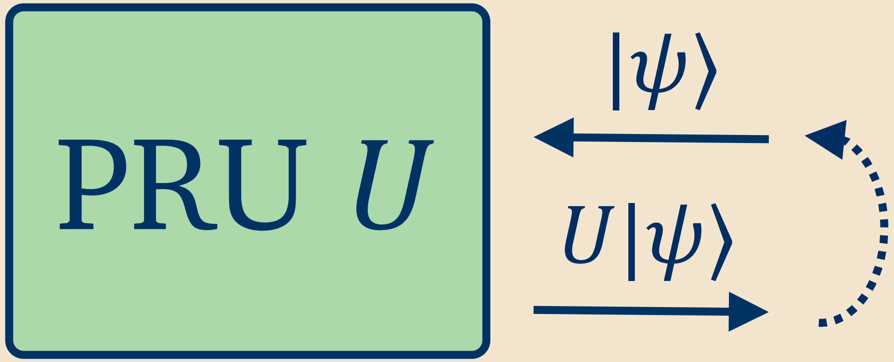
Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random



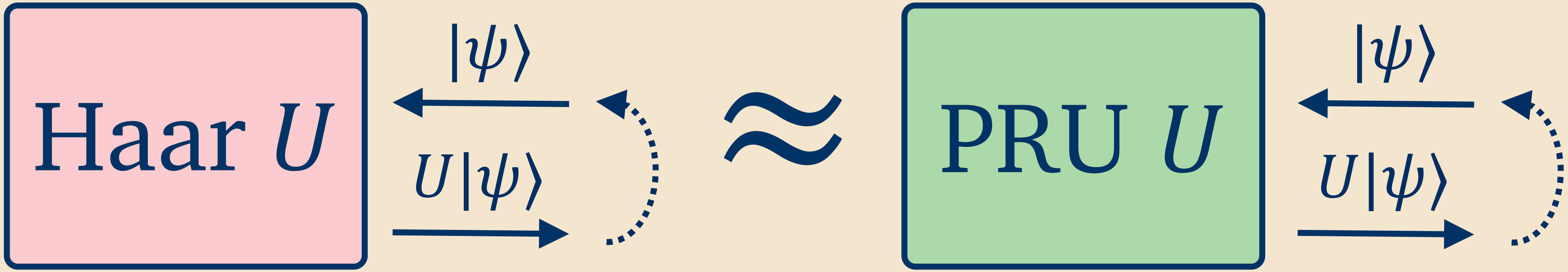
Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random



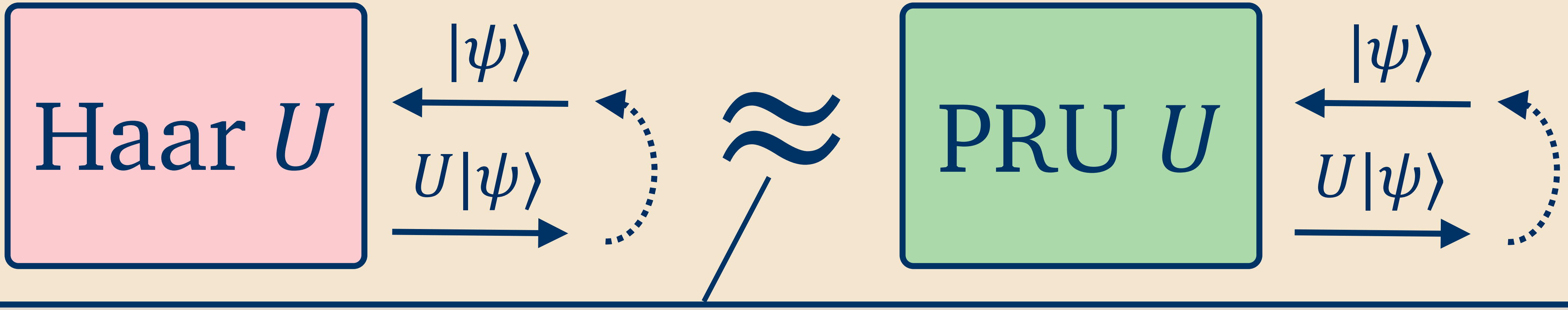
Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random



Pseudorandom unitaries (PRUs) [JLS18]

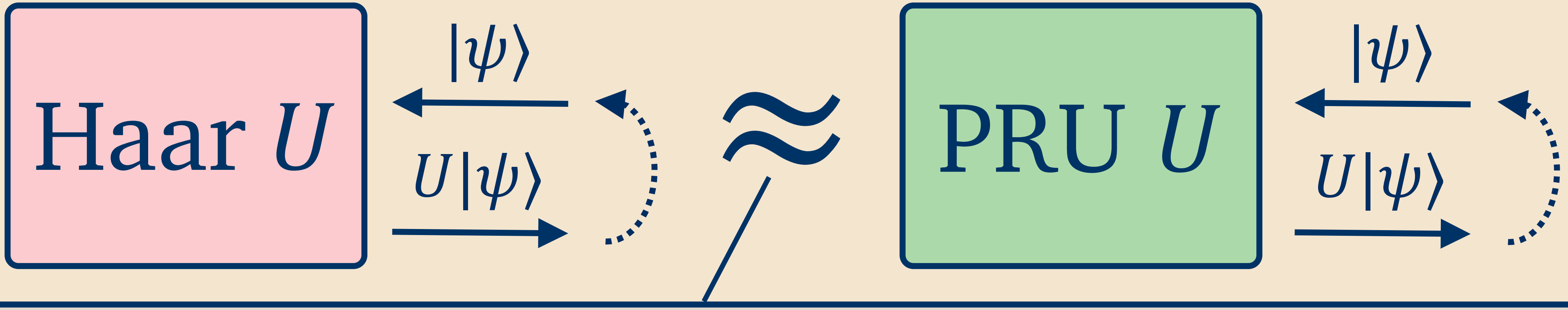
efficiently-computable unitaries that appear Haar-random



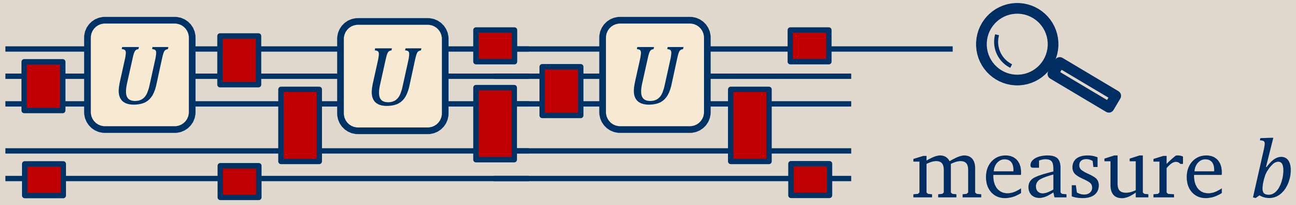
For any efficient algorithm A:

Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random

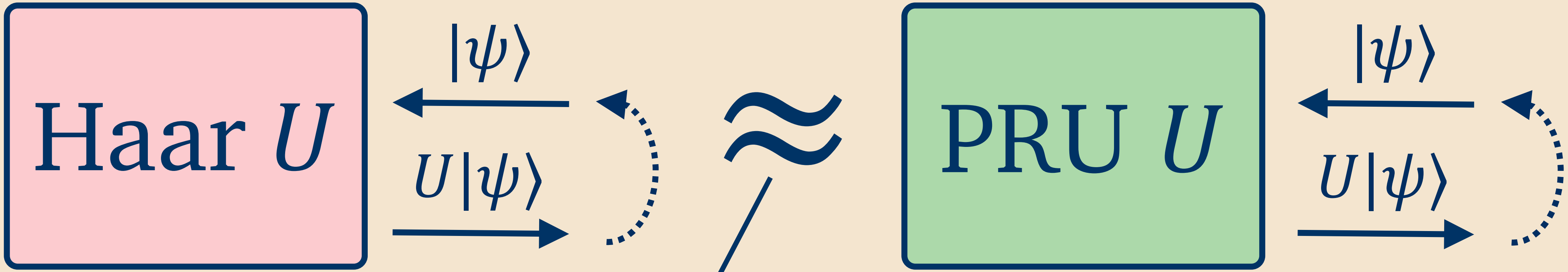


For any efficient algorithm A:

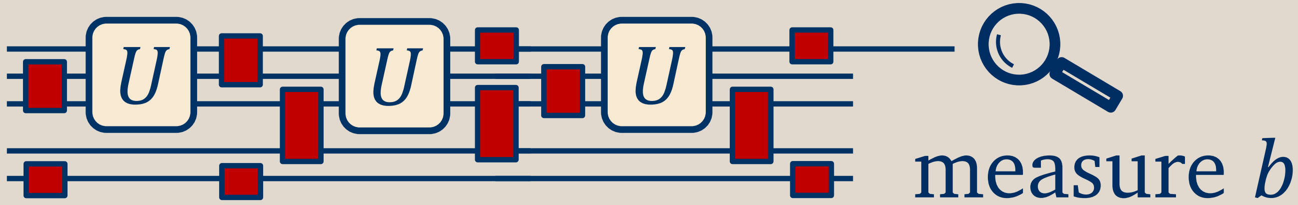


Pseudorandom unitaries (PRUs) [JLS18]

efficiently-computable unitaries that appear Haar-random



For any efficient algorithm A:



$$\Pr[b = 1 \mid U \leftarrow \text{Haar}] \approx \Pr[b = 1 \mid U \leftarrow \text{PRU}]$$

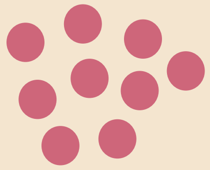
Application: modeling black hole dynamics

Application: modeling black hole dynamics

Physicists often study the dynamics of chaotic systems. Key example:

Application: modeling black hole dynamics

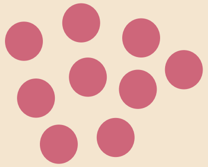
Physicists often study the dynamics of chaotic systems. Key example:



particles in state $|\psi\rangle$

Application: modeling black hole dynamics

Physicists often study the dynamics of chaotic systems. Key example:

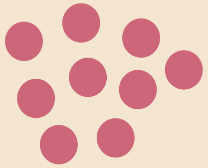


particles in state $|\psi\rangle$

collapse into black hole →

Application: modeling black hole dynamics

Physicists often study the dynamics of chaotic systems. Key example:



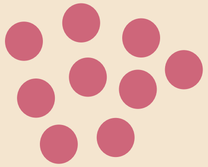
particles in state $|\psi\rangle$

collapse into black hole →

black hole evaporates →

Application: modeling black hole dynamics

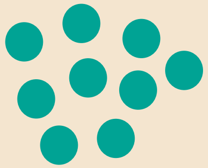
Physicists often study the dynamics of chaotic systems. Key example:



particles in state $|\psi\rangle$

collapse into black hole →

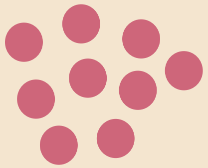
black hole evaporates →



particles in state $|\psi'\rangle$

Application: modeling black hole dynamics

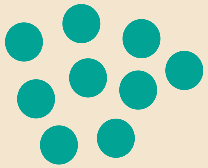
Common practice: pretend this Haar-random



particles in state $|\psi\rangle$

collapse into black hole →

black hole evaporates →



particles in state $|\psi'\rangle$

Application: modeling black hole dynamics

Common practice: pretend this Haar-random

particles in state $|\psi\rangle$

collapse into black hole →

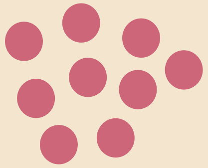
black hole evaporates →

particles in state $|\psi'\rangle$

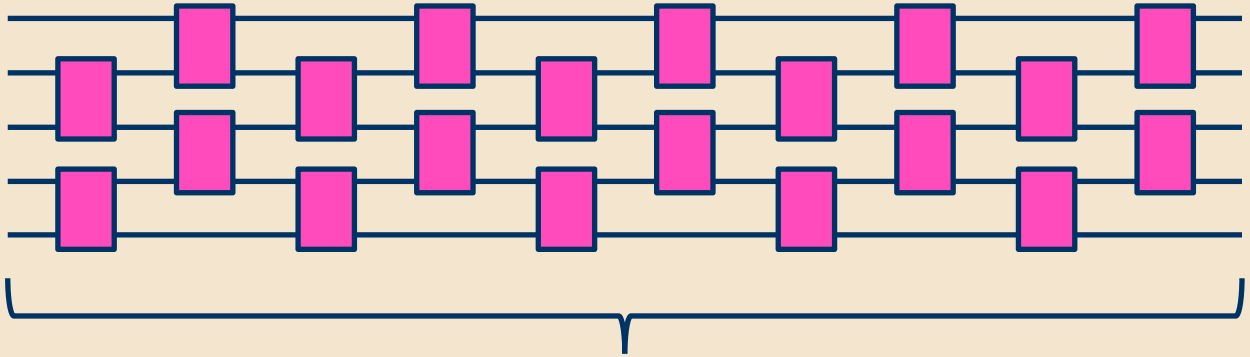
However, Quantum Church-Turing says this has to be efficient!

Application: modeling black hole dynamics

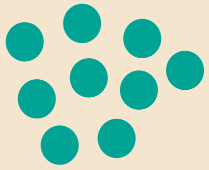
Common practice: pretend this Haar-random



particles in state $|\psi\rangle$



size = poly(# of qubits)

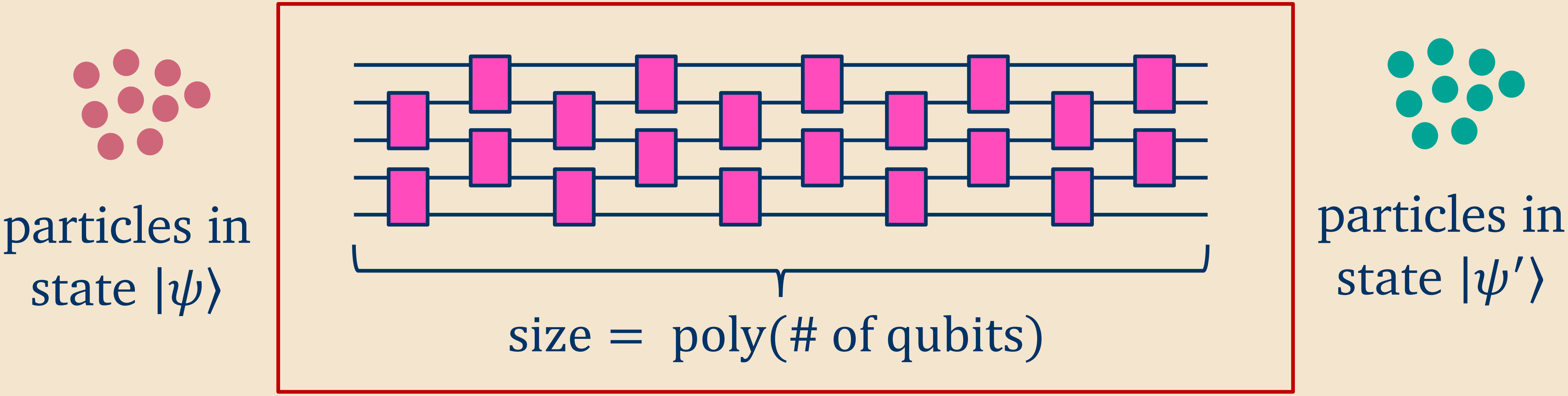


particles in state $|\psi'\rangle$

However, Quantum Church-Turing says this has to be efficient!

Application: modeling black hole dynamics

Common practice: pretend this Haar-random



However, Quantum Church-Turing says this has to be efficient!

Consequence: many physics results now rely on the assumption that various physical processes are PRUs [KP23,YE23,EFLVY24]

But do PRUs exist?

But do PRUs exist?
(under crypto assumptions)

But do PRUs exist?
(under crypto assumptions)

This was left as an open problem by [JLS18].

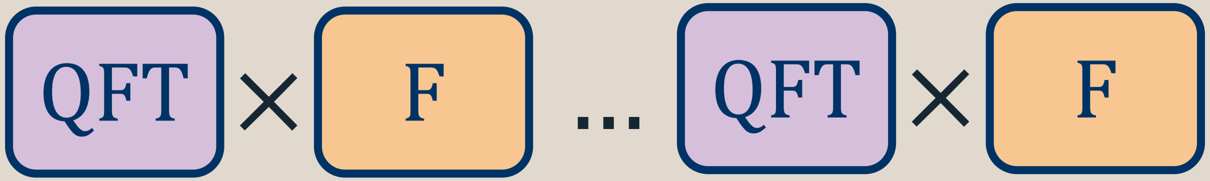
Prior work

Prior work

1) Many proposed constructions:

Prior work

1) Many proposed constructions:



[JLS18]

Prior work

1) Many proposed constructions:

QFT × F … QFT × F

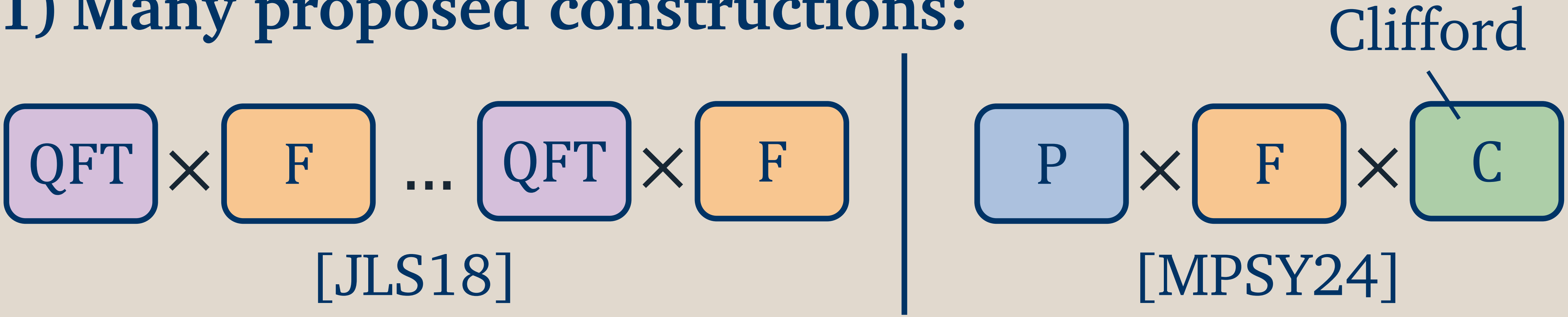
[JLS18]

P × F × C (C = Clifford)

[MPSY24]

Prior work

1) Many proposed constructions:



2) Proofs of non-adaptive security [MPSY24, CBBDHX24]

Prior work

1) Many proposed constructions:

QFT × F … QFT × F

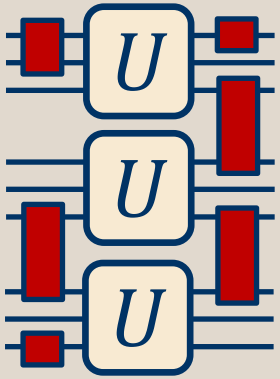
[JLS18]

P × F × C (C = Clifford)

[MPSY24]

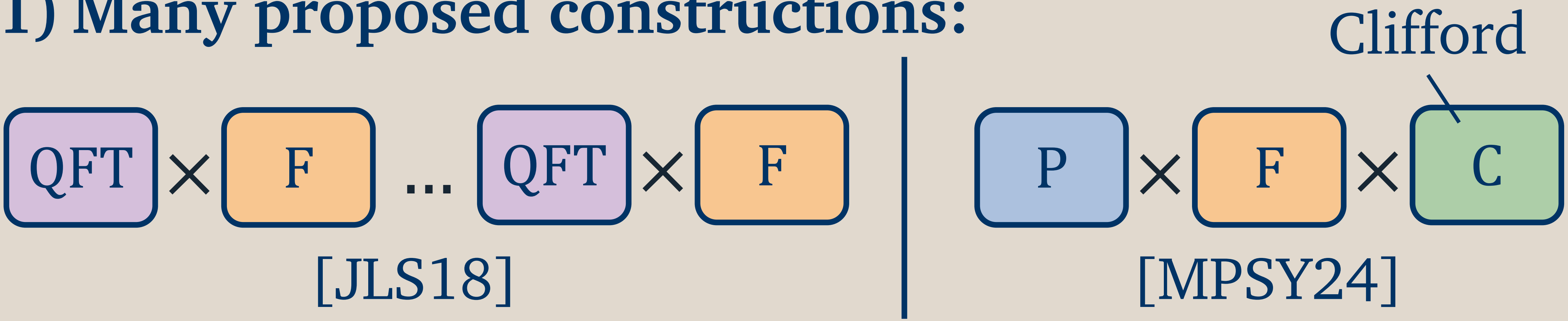
2) Proofs of non-adaptive security [MPSY24, CBBDHX24]

can analyze this:



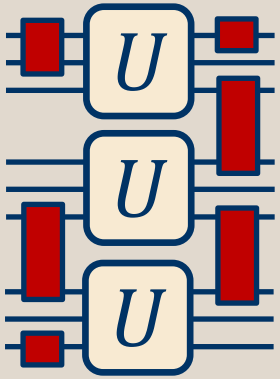
Prior work

1) Many proposed constructions:

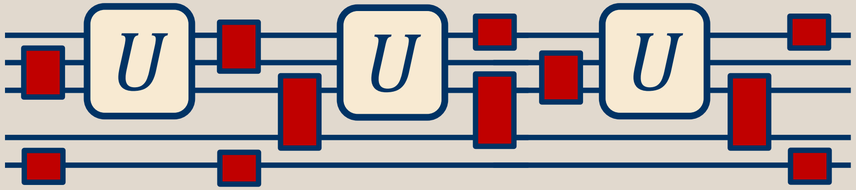


2) Proofs of non-adaptive security [MPSY24, CBBDHX24]

can analyze this:



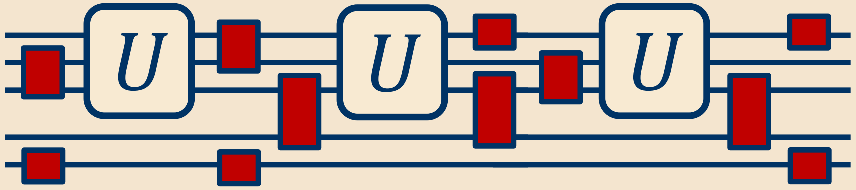
but not this:



Why has it been hard to prove PRUs exist?

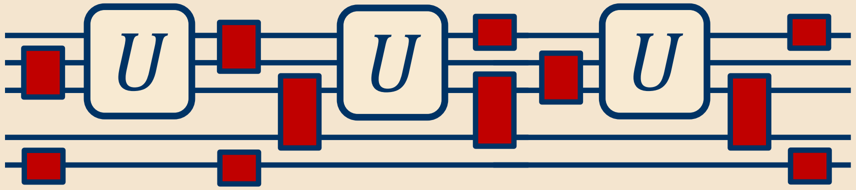
Why has it been hard to prove PRUs exist?

1) Need to understand behavior of an arbitrary algorithm:



Why has it been hard to prove PRUs exist?

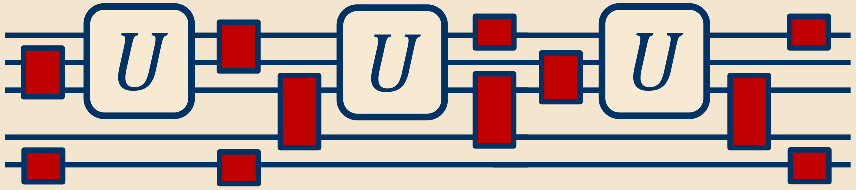
1) Need to understand behavior of an arbitrary algorithm:



2) Mathematics of random unitaries is complicated.

Why has it been hard to prove PRUs exist?

1) Need to understand behavior of an arbitrary algorithm:



2) Mathematics of random unitaries is complicated.

- Weingarten calculus
- free probability
- ???

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 $d \geq k$.

[MH24]: PRUs exist

(if one-way functions exist)

[MH24]: PRUs exist

(if one-way functions exist)

Same construction as [MPSY24]:

P × F × C

[MH24]: PRUs exist

(if one-way functions exist)

Same construction
as [MPSY24]:

P × F × C

New technique: the path-recording oracle

[MH24]: PRUs exist

(if one-way functions exist)

Same construction as [MPSY24]:

P × F × C

New technique: the path-recording oracle

- efficient simulation of Haar-random unitaries

[MH24]: PRUs exist

(if one-way functions exist)

Same construction as [MPSY24]:

P × F × C

New technique: the path-recording oracle

- efficient simulation of Haar-random unitaries
- only uses basic quantum info (purification)

Our second result

Our second result

In the [JLS18] PRU definition, the distinguisher only queries U. **What if it queries $\boldsymbol{U}$ and $\boldsymbol{U^\dagger}$?**

Our second result

In the [JLS18] PRU definition, the distinguisher only queries U. **What if it queries U and $U^\dagger$?**

Result #2: "Strong" PRUs exist (assuming OWFs).

Construction: $C_1 \times \mathrm{P} \times \mathrm{F} \times C_2$

Our second result

In the [JLS18] PRU definition, the distinguisher only queries U. **What if it queries $\boldsymbol{U}$ and $\boldsymbol{U^\dagger}$?**

Result #2: "Strong" PRUs exist (assuming OWFs).

Construction: $C_1 \times \mathrm{P} \times \mathrm{F} \times C_2$

[SMLBH25]: same proof extends to U^T and U^*.

Our second result

In the [JLS18] PRU definition, the distinguisher only queries U. **What if it queries U and $U^\dagger$?**

Result #2: "Strong" PRUs exist (assuming OWFs).

Construction: $C_1 \times \mathrm{P} \times \mathrm{F} \times C_2$

[SMLBH25]: same proof extends to U^T and U^*.

But for this talk, I'll focus on the weakest notion.

Cartoon overview of our proof

Cartoon overview of our proof

Want to show:

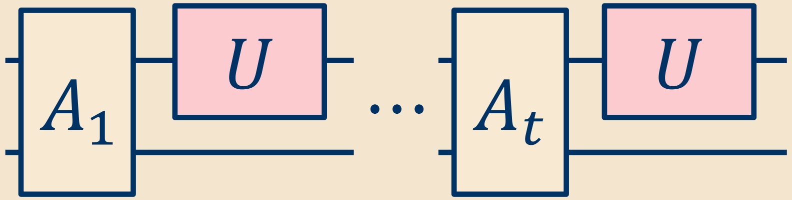
For all efficient algorithms A,

Cartoon overview of our proof

Want to show:

For all efficient algorithms A,

$U \leftarrow$ **Haar**

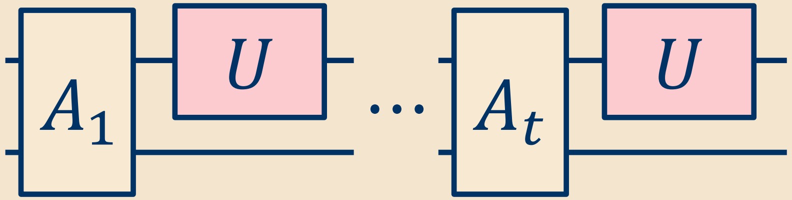


Cartoon overview of our proof

Want to show:

For all efficient algorithms A,

$U \leftarrow$ **Haar**



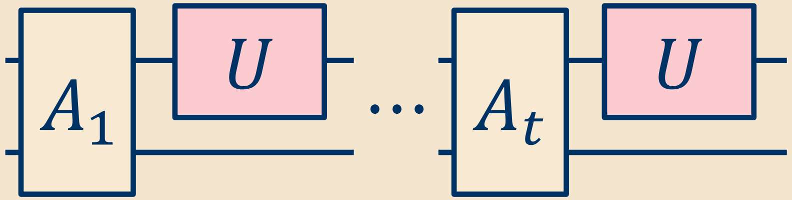
$\approx$

Cartoon overview of our proof

Want to show:

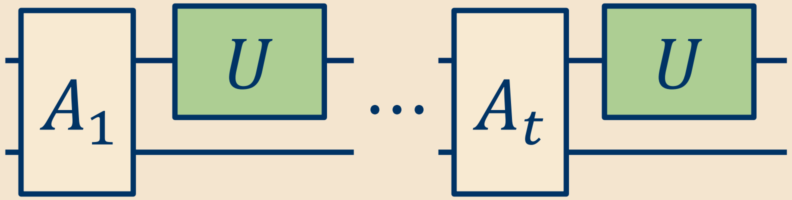
For all efficient algorithms A,

$U \leftarrow$ **Haar**



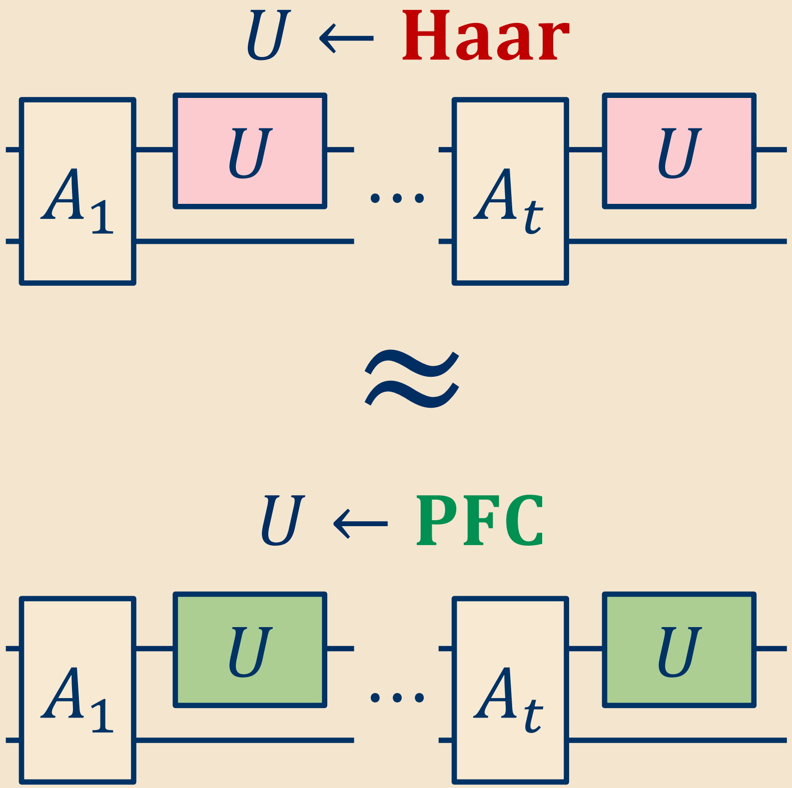
$$\approx$$

$U \leftarrow$ **PFC**



Cartoon overview of our proof

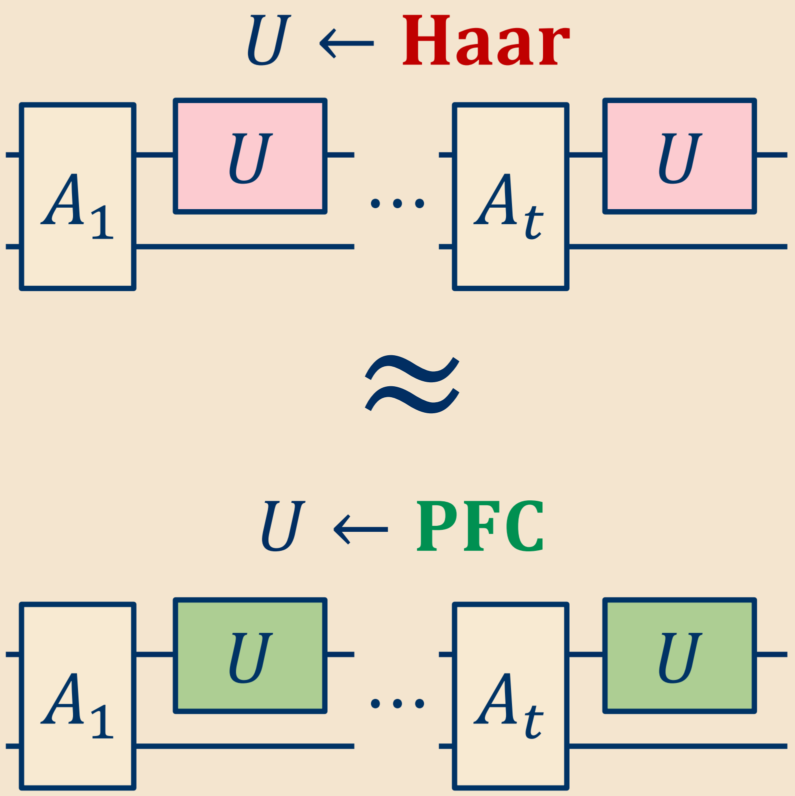
Want to show:
For all efficient algorithms A,



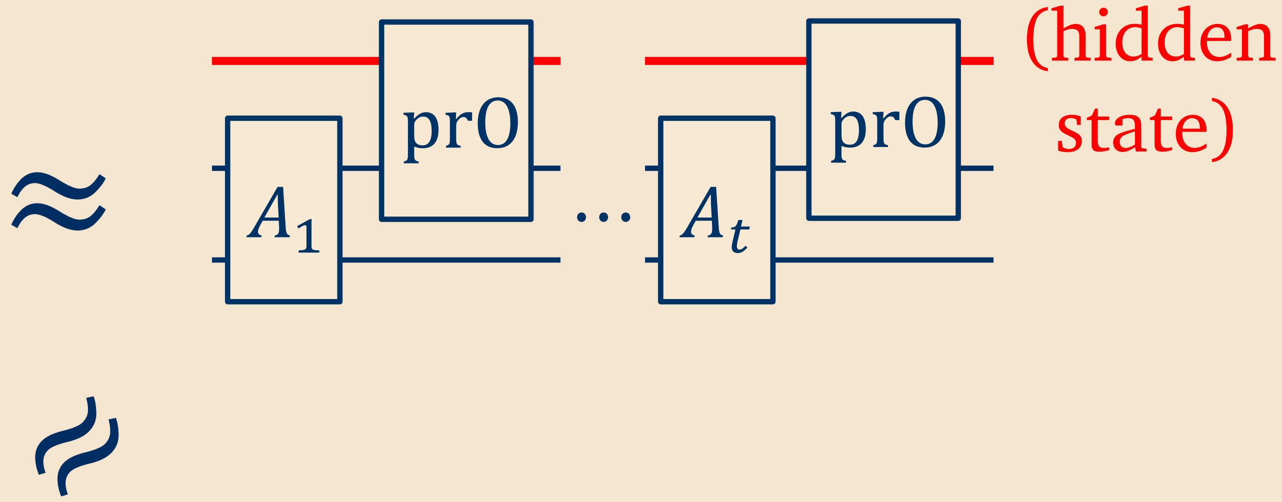
Proof strategy: show that both are indistinguishable from

Cartoon overview of our proof

Want to show:
For all efficient algorithms A,



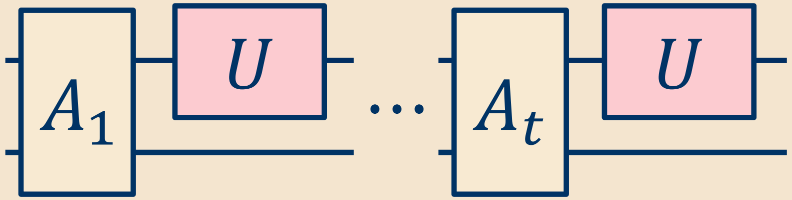
Proof strategy: show that both are indistinguishable from



Cartoon overview of our proof

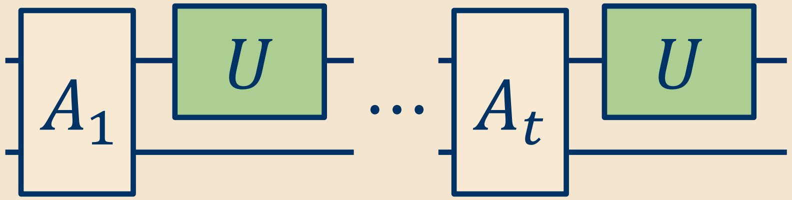
Want to show:
For all efficient algorithms A,

$U \leftarrow$ **Haar**



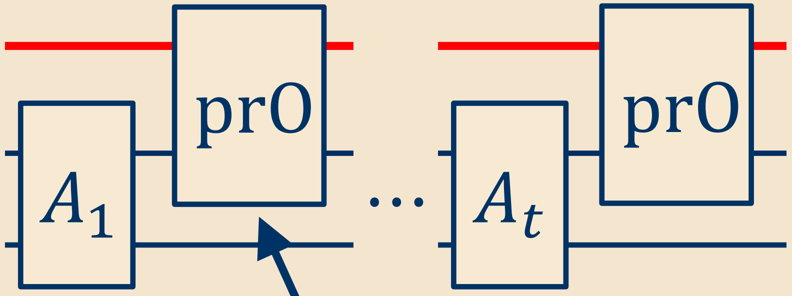
$\approx$

$U \leftarrow$ **PFC**



Proof strategy: show that both are indistinguishable from

$\approx$



(hidden state)

$\approx$

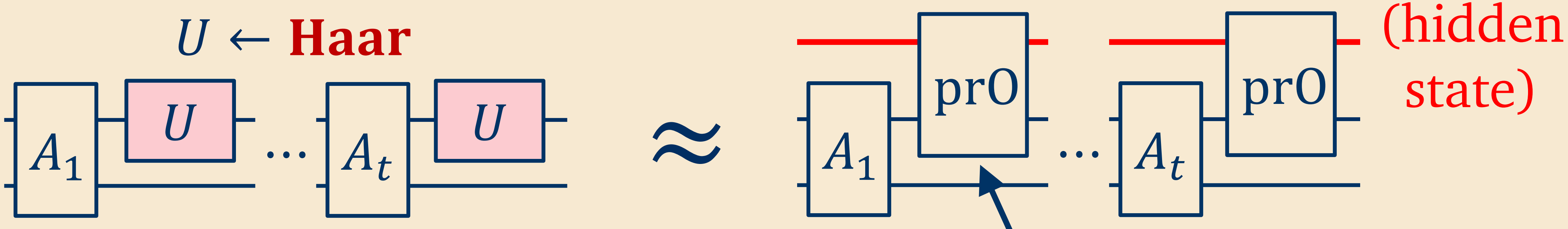
(path-recording oracle) maintains a data structure that "lazily samples" a Haar-random unitary

Cartoon overview of our proof

Want to show:
For all efficient algorithms A,

Proof strategy: show that both
are indistinguishable from

most of the proof



(path-recording oracle)
maintains a data structure that "lazily samples" a Haar-random unitary

Rest of this talk

- **Lazy sampling of a random function**
- Lazy sampling of a random unitary
- Proving correctness + PRUs exist
- Applications

Lazy sampling of a random **function**

Lazy sampling of a random function

Goal: efficiently implement an algorithm that queries a random **function** f.

Lazy sampling of a random function

Goal: efficiently implement an algorithm that queries a random **function** f.

Solution: • only sample $f(x)$ when needed, "on the fly"

Lazy sampling of a random **function**

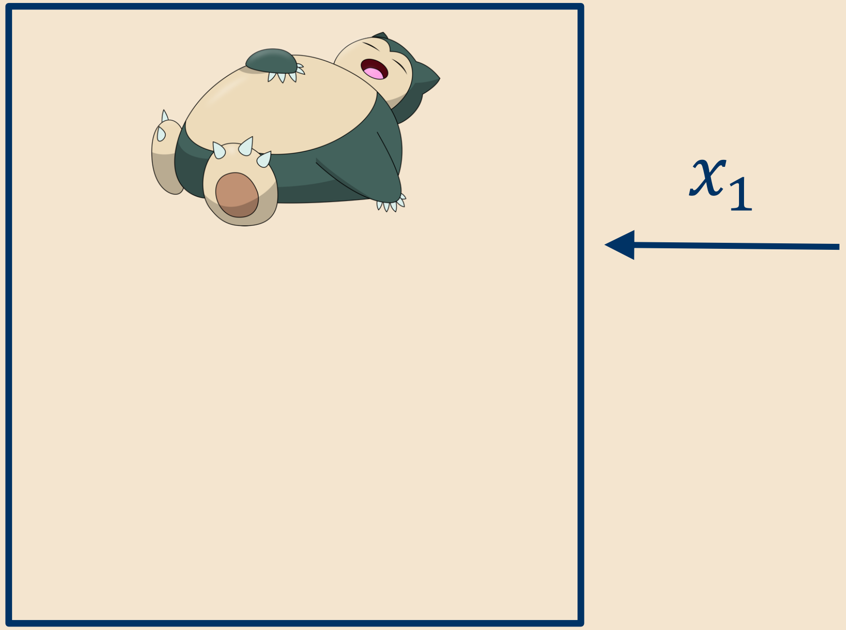
Goal: efficiently implement an algorithm that queries a random **function** f.

Solution:
- only sample $f(x)$ when needed, “on the fly”
- remember what you sampled (for consistency)

Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

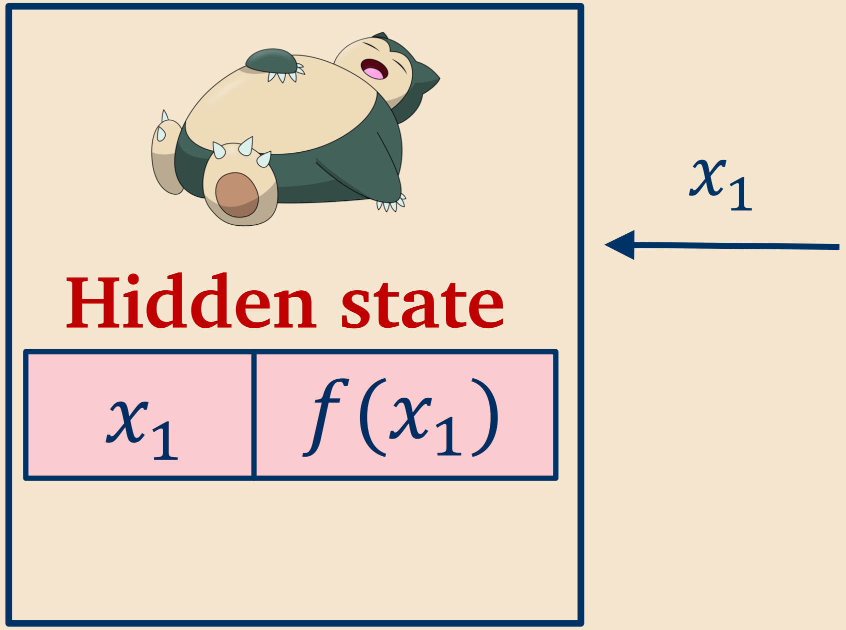
Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

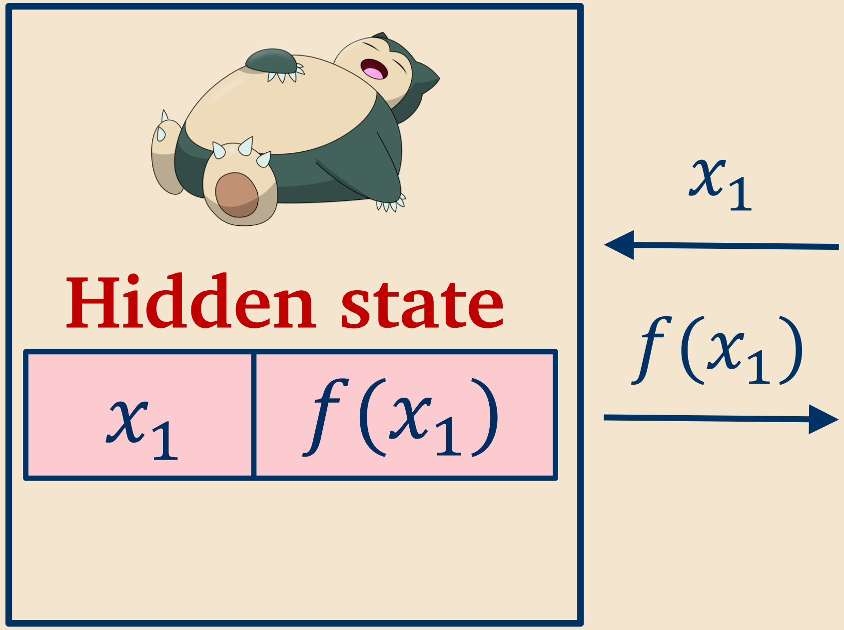
Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

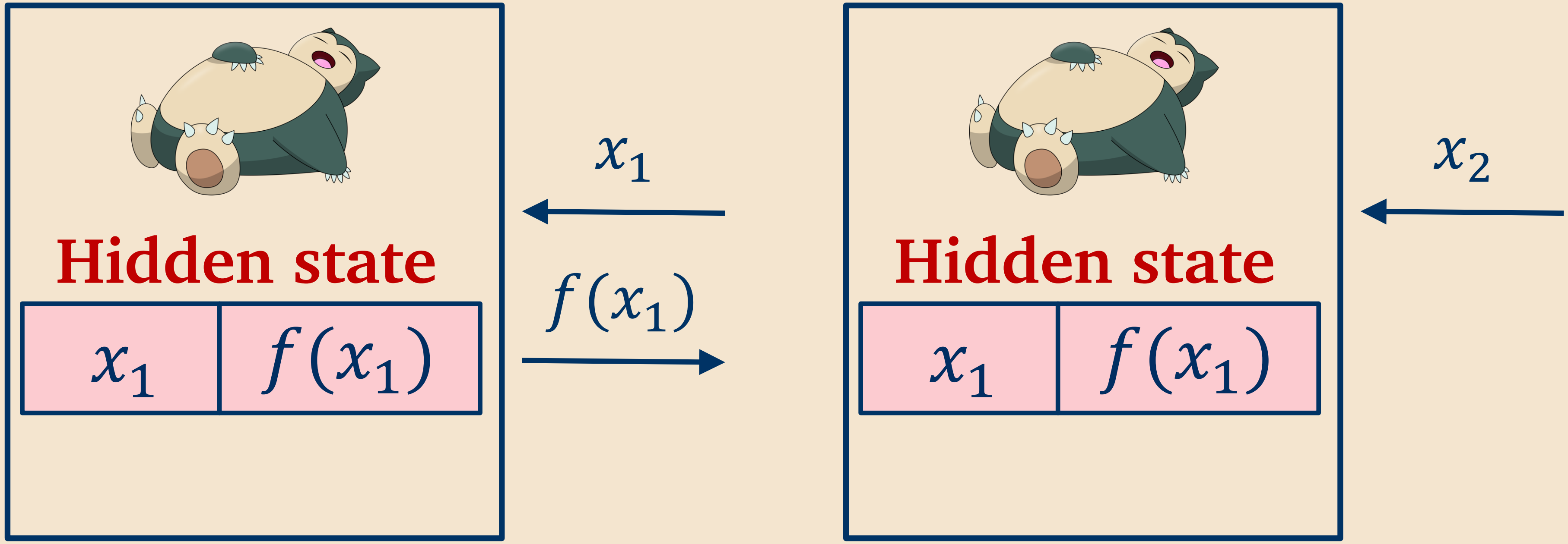
Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

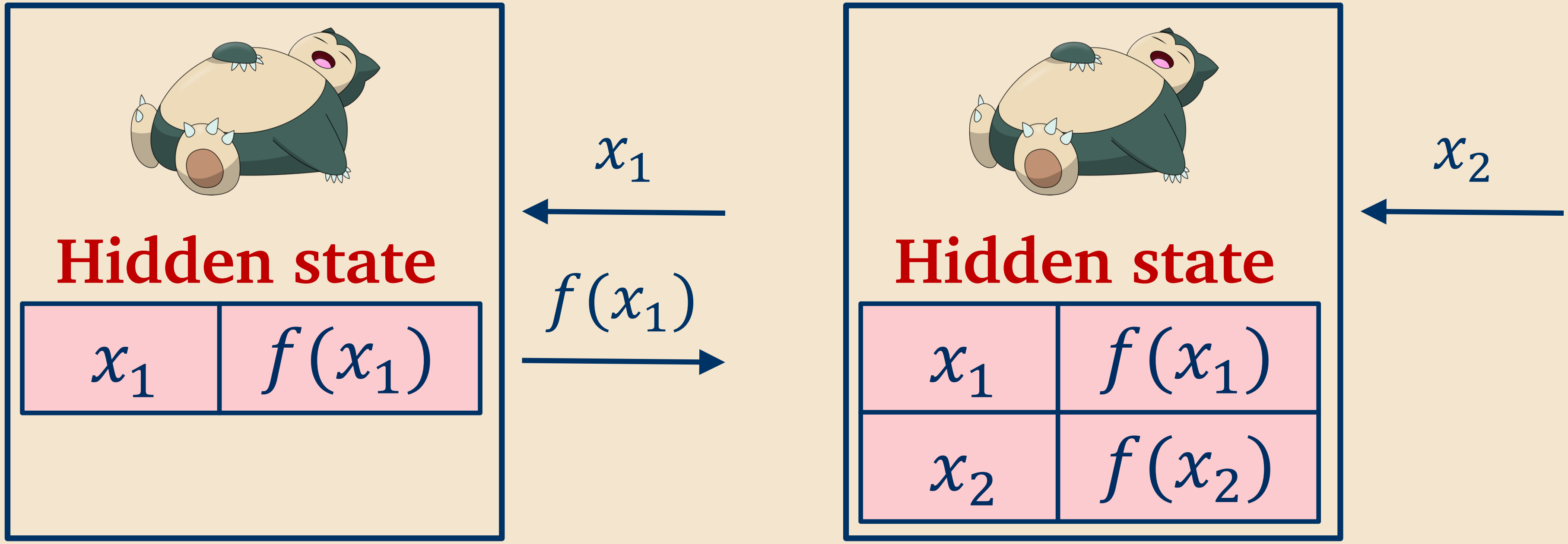
Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

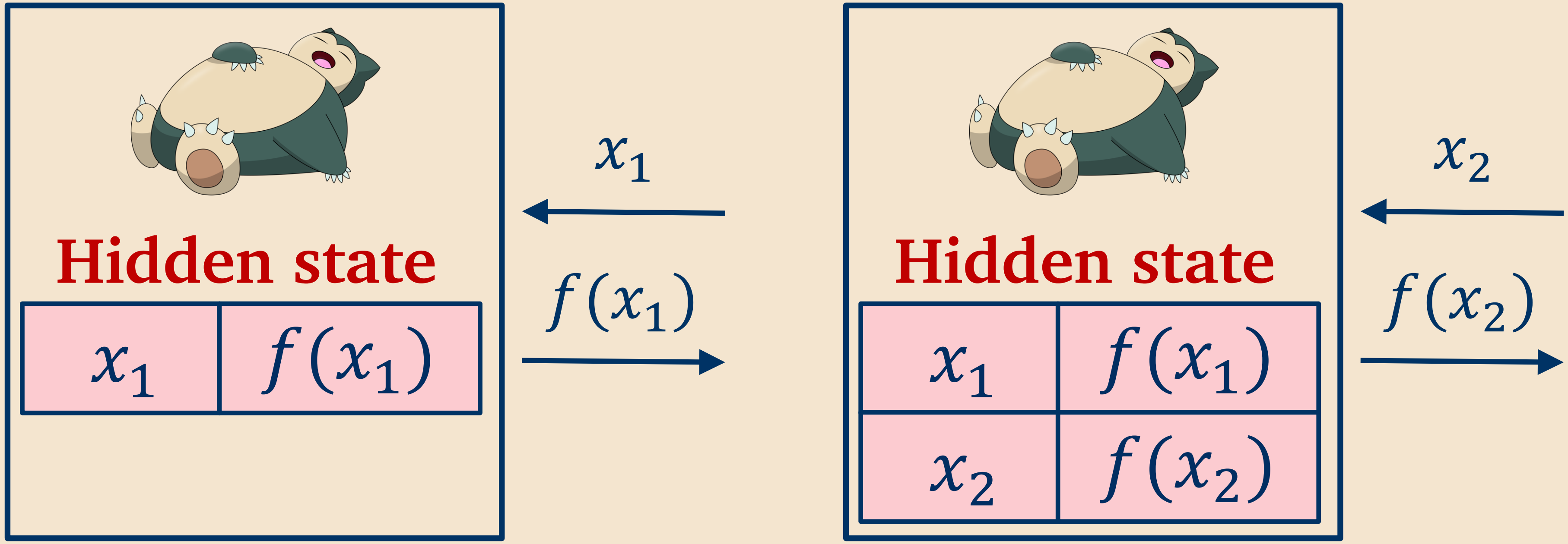
Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Lazy sampling of a random **function**

Goal: efficiently implement an algorithm that queries a random **function** f.

Solution:
- only sample $f(x)$ when needed, "on the fly"
- remember what you sampled (for consistency)



Rest of this talk

- Lazy sampling of a random function
- **Lazy sampling of a random unitary**
- Proving correctness + PRUs exist
- Applications

Lazy sampling of a random **unitary**

Lazy sampling of a random **unitary**

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

Lazy sampling of a random **unitary**

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

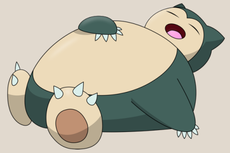
A priori, not clear how to do this!

Lazy sampling of a random **unitary**

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

A priori, not clear how to do this!

Our solution: the path-recording oracle



Lazy sampling of a random unitary

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

A priori, not clear how to do this!

Our solution: the path-recording oracle

We use **entanglement** with a **hidden data structure** that succinctly "remembers" enough information to spoof a Haar-random U.

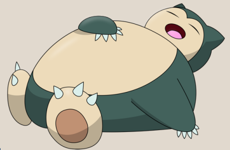
Lazy sampling of a random **unitary**

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

A priori, not clear how to do this!

Classically, the data structure is the set of $(x, f(x))$ tuples.

Our solution: the path-recording oracle



We use **entanglement** with a **hidden data structure** that succinctly "remembers" enough information to spoof a Haar-random U.

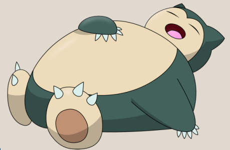
Lazy sampling of a random **unitary**

Goal: efficiently implement a quantum algorithm that queries a Haar-random unitary U.

Classically, the data structure is the set of $(x, f(x))$ tuples.

A priori, not clear how to do this!

Our solution: the path-recording oracle



We use **entanglement** with a **hidden data structure** that succinctly “remembers” enough information to spoof a Haar-random U.

Inspiration: compressed oracle technique [Zhandry19]

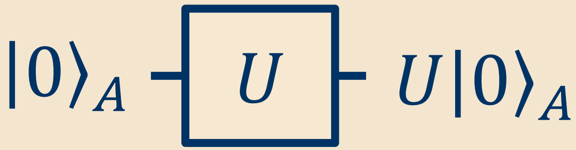
Up next:

"Derive" the path-recording oracle through simple examples

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

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

The algorithm:



$(U \leftarrow \text{Haar})$

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

The algorithm: $|0\rangle_A$ —[U]— $U|0\rangle_A$ ← Fact: this is the "maximally mixed" state

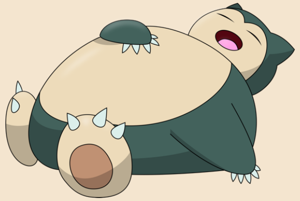
$(U \leftarrow \text{Haar})$

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

The algorithm: $|0\rangle_A$ —[U]— $U|0\rangle_A$ ← Fact: this is the "maximally mixed" state

$(U \leftarrow \text{Haar})$

How to "spoof" it:



$$\sum_y |y\rangle_A {\color{red}|y\rangle_S}$$

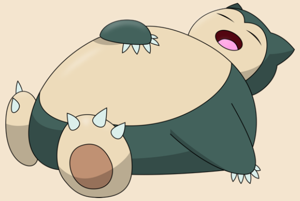
(S register is hidden)

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

The algorithm: $|0\rangle_A$ —[U]— $U|0\rangle_A$ ← Fact: this is the "maximally mixed" state

$(U \leftarrow \text{Haar})$

How to "spoof" it:



$$\sum_y |y\rangle_A |y\rangle_S$$ ← Fact: this is *also* the maximally mixed state.

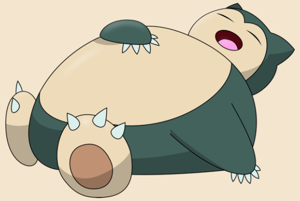
(S register is hidden)

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

The algorithm: $|0\rangle_A$ —[U]— $U|0\rangle_A$ ← Fact: this is the "maximally mixed" state

$(U \leftarrow \text{Haar})$

How to "spoof" it:



$$\sum_y |y\rangle_A |y\rangle_S$$

← Fact: this is ***also*** the maximally mixed state.

(S register is hidden)

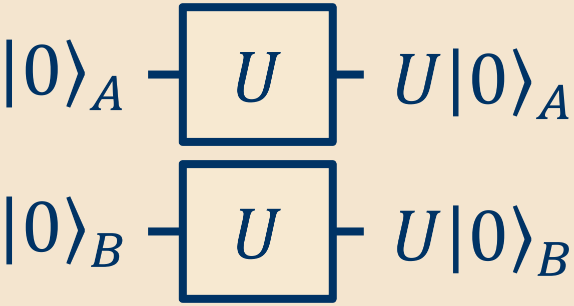
Idea 1: entanglement with a hidden register S can simulate one query to U.

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

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

The algorithm:

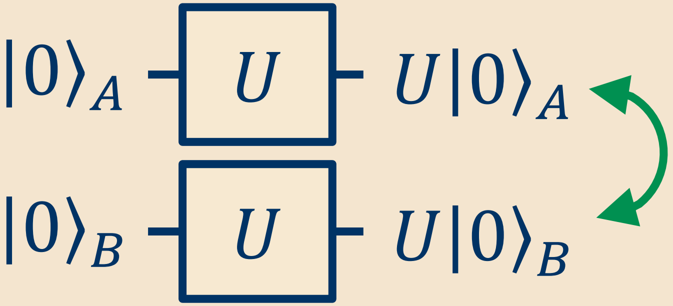
$(U \leftarrow \text{Haar})$



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

The algorithm:

$(U \leftarrow \text{Haar})$



Fact: this is the maximally mixed "symmetric" state

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

The algorithm:

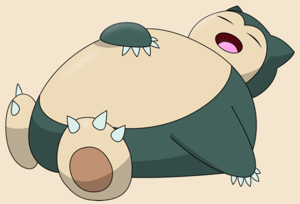
$(U \leftarrow \text{Haar})$

$|0\rangle_A$ — U — $U|0\rangle_A$

$|0\rangle_B$ — U — $U|0\rangle_B$

Fact: this is the maximally mixed "symmetric" state

How to "spoof" it:



$$\sum_{y_1, y_2} |y_1\rangle_A |y_2\rangle_B |\{y_1, y_2\}\rangle_S$$

(S register is hidden)

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

The algorithm:

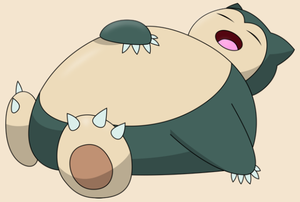
$(U \leftarrow \text{Haar})$

$|0\rangle_A$ — U — $U|0\rangle_A$

$|0\rangle_B$ — U — $U|0\rangle_B$

Fact: this is the maximally mixed "symmetric" state

How to "spoof" it:



also symmetric!

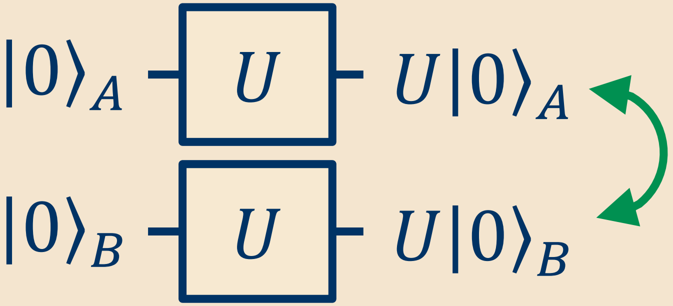
$$\sum_{y_1,y_2} |y_1\rangle_A |y_2\rangle_B |\{y_1, y_2\}\rangle_S$$

(S register is hidden)

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

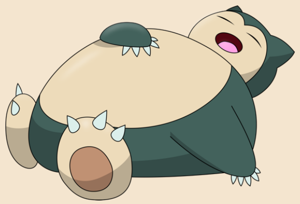
The algorithm:

$(U \leftarrow \text{Haar})$



Fact: this is the maximally mixed "symmetric" state

How to "spoof" it:



also symmetric!

$$\sum_{y_1,y_2} |y_1\rangle_A |y_2\rangle_B |\{y_1, y_2\}\rangle_S$$

(S register is hidden)

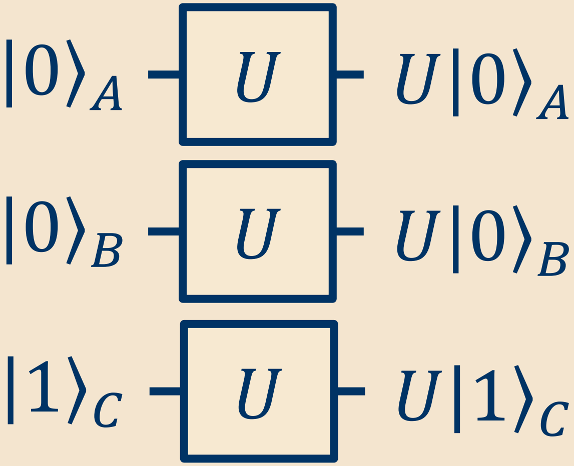
Idea 2: use an **unordered set** to spoof "swap-symmetry".

Example 3: mixed queries

Example 3: mixed queries

The algorithm:

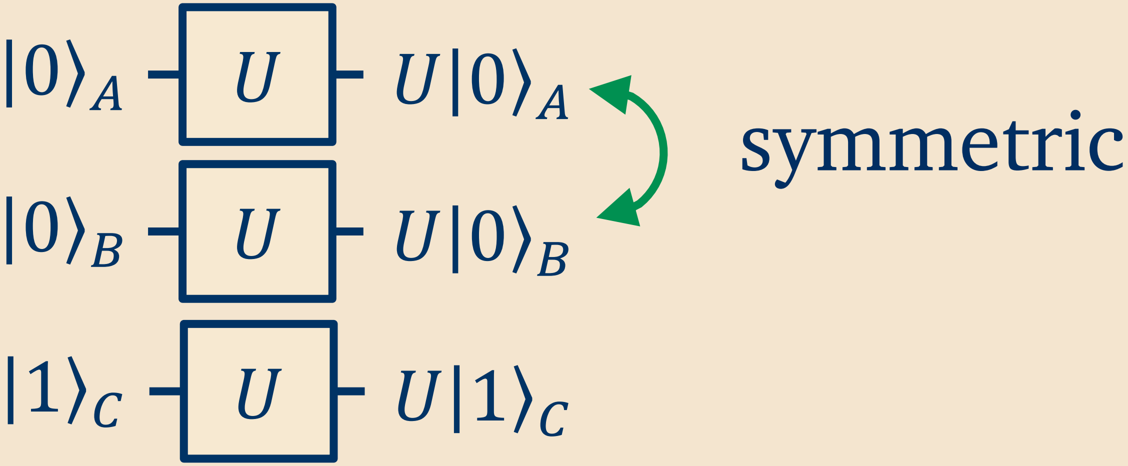
$(U \leftarrow \text{Haar})$



Example 3: mixed queries

The algorithm:

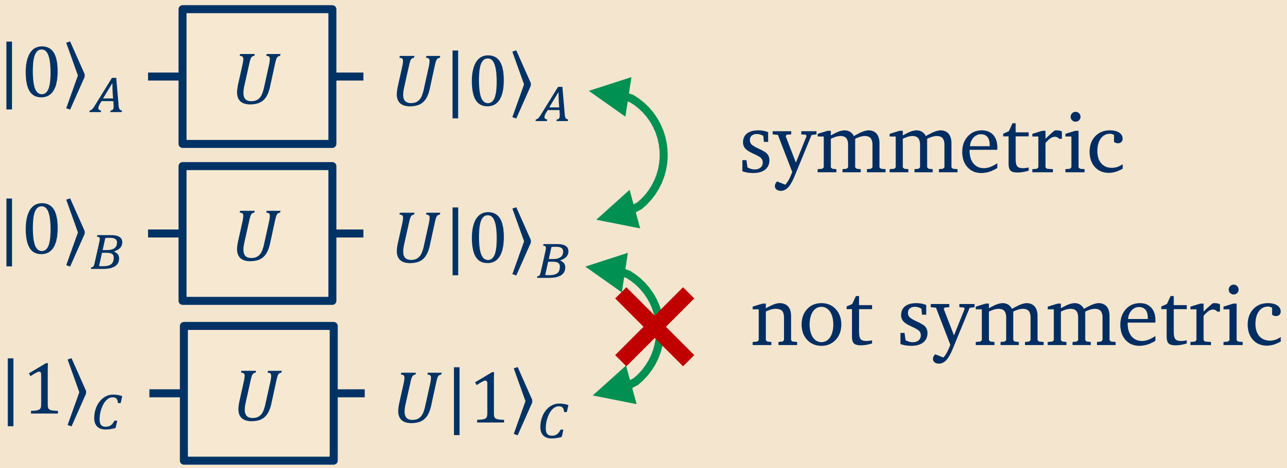
$(U \leftarrow \text{Haar})$



Example 3: mixed queries

The algorithm:

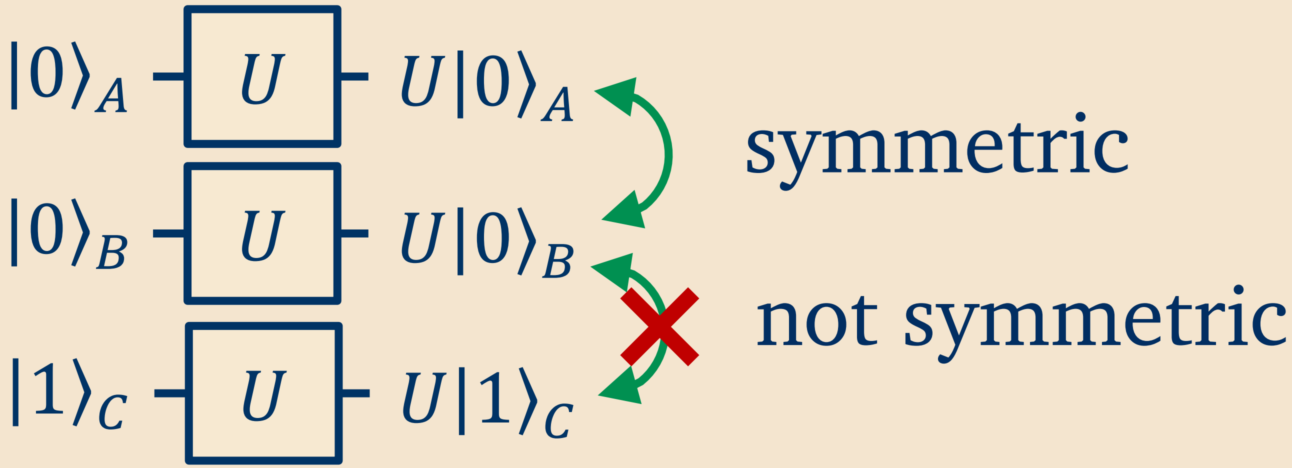
$(U \leftarrow \text{Haar})$



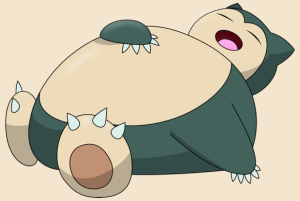
Example 3: mixed queries

The algorithm:

$(U \leftarrow \text{Haar})$



How to "spoof" it:

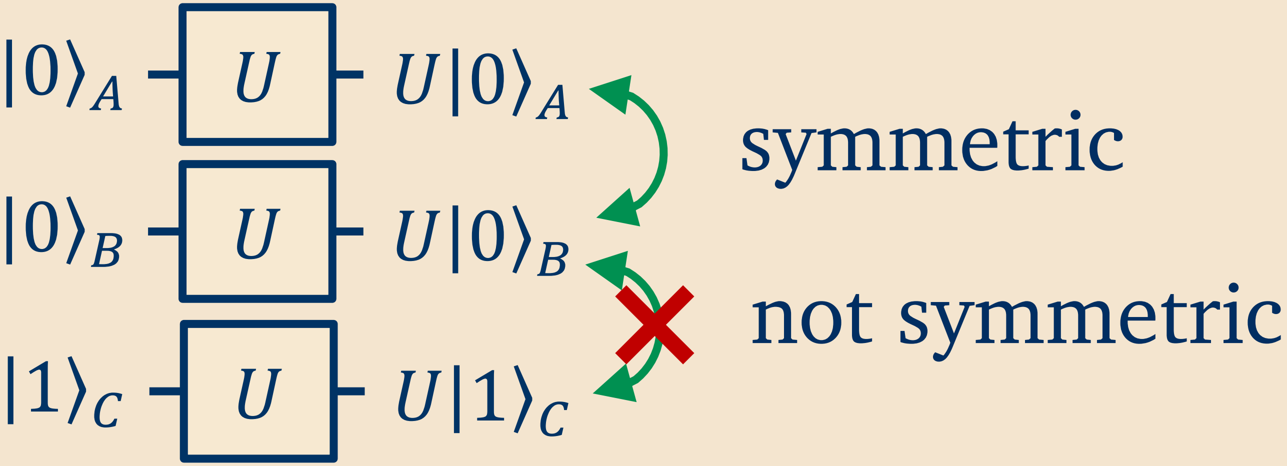


$$\sum_{y_1,y_2,y_3} |y_1\rangle_A |y_2\rangle_B |y_3\rangle_C |\{(0,y_1),(0,y_2),(1,y_3)\}\rangle_S$$

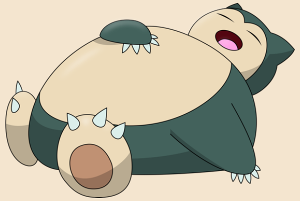
Example 3: mixed queries

The algorithm:

$(U \leftarrow \text{Haar})$



How to "spoof" it:



$$\sum_{y_1,y_2,y_3} |y_1\rangle_A |y_2\rangle_B |y_3\rangle_C |\{(0,y_1),(0,y_2),(1,y_3)\}\rangle_S$$

Example 3: mixed queries

The algorithm:

$(U \leftarrow \text{Haar})$

$|0\rangle_A$ — U — $U|0\rangle_A$

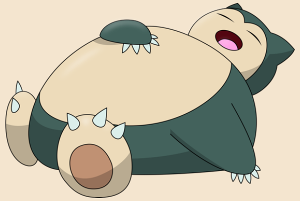
$|0\rangle_B$ — U — $U|0\rangle_B$

$|1\rangle_C$ — U — $U|1\rangle_C$

$U|0\rangle_A$ ↔ $U|0\rangle_B$: symmetric

$U|0\rangle_B$ ↔ $U|1\rangle_C$: not symmetric

How to "spoof" it:

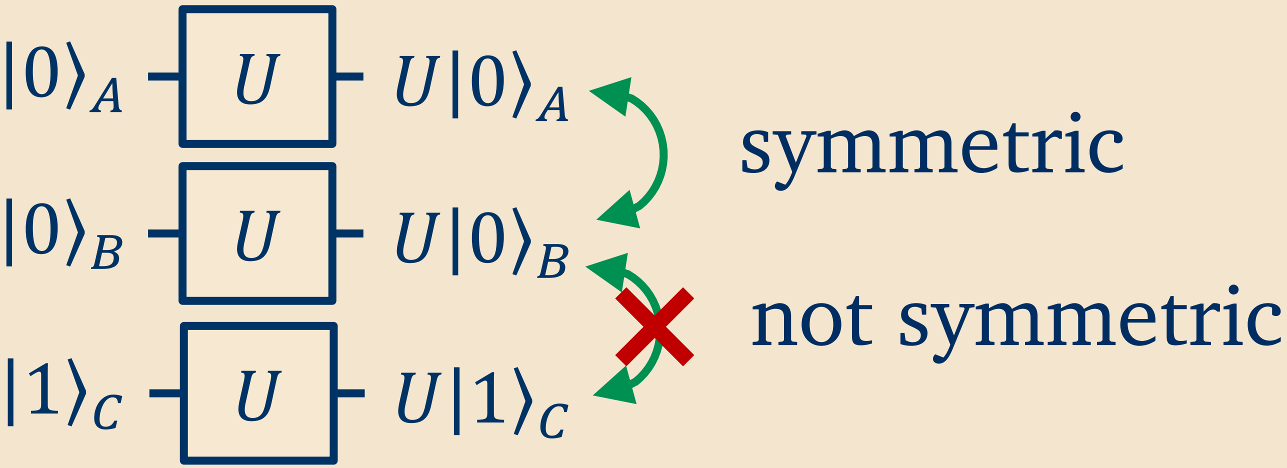


$$\sum_{y_1,y_2,y_3} |y_1\rangle_A |y_2\rangle_B |y_3\rangle_C |\{(0,y_1),(0,y_2),(1,y_3)\}\rangle_S$$

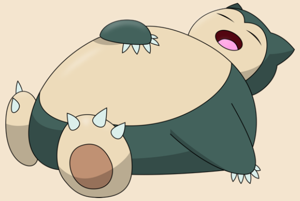
Example 3: mixed queries

The algorithm:

$(U \leftarrow \text{Haar})$



How to "spoof" it:

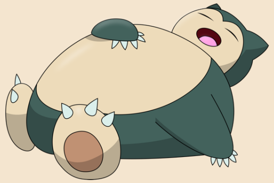


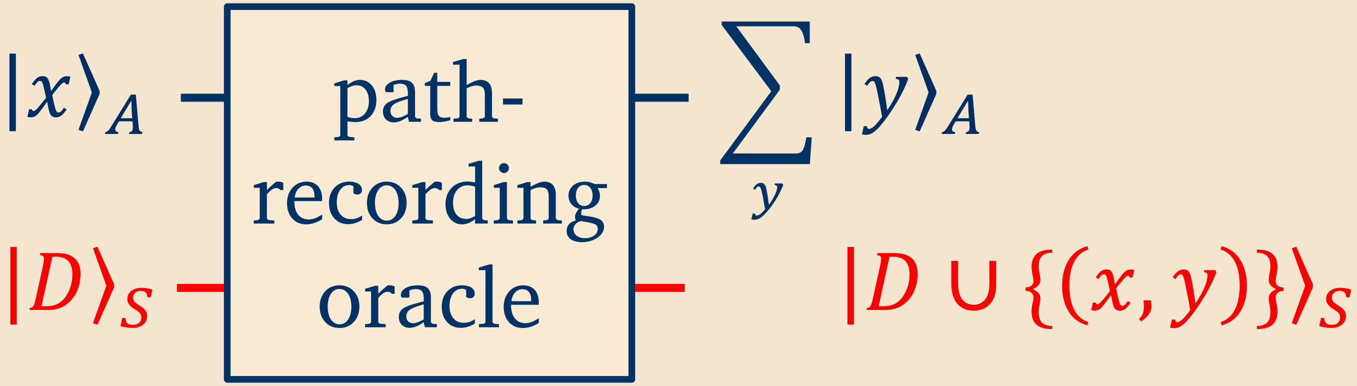
$$\sum_{y_1,y_2,y_3} |y_1\rangle_A |y_2\rangle_B |y_3\rangle_C |\{(0,y_1),(0,y_2),(1,y_3)\}\rangle_S$$

Idea 3: use **ordered pairs** to simulate symmetry "structure"

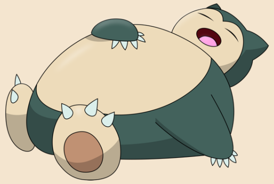
We can generate all of these examples by simply replacing each query to U with this:

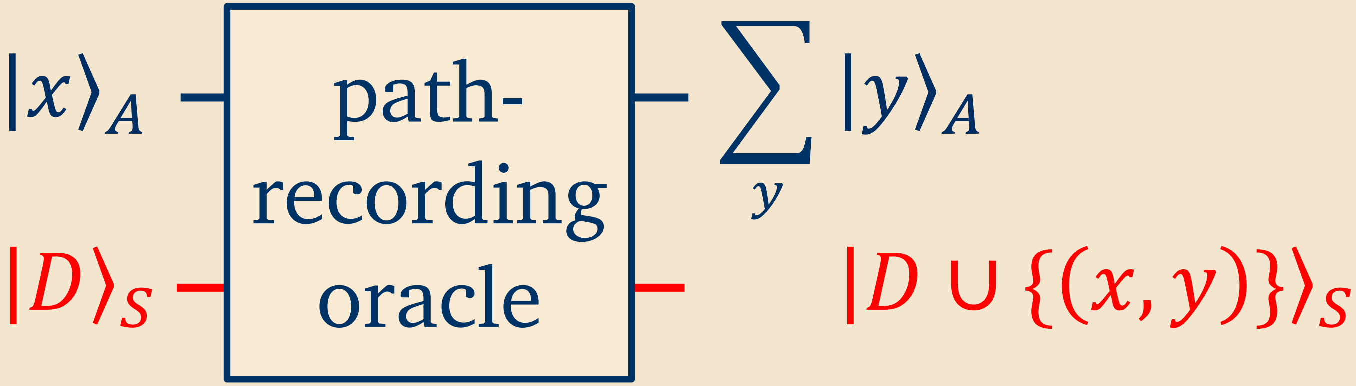
We can generate all of these examples by simply replacing each query to U with this:





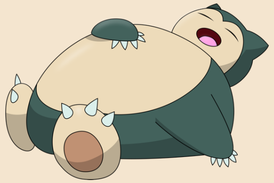
We can generate all of these examples by simply replacing each query to U with this:

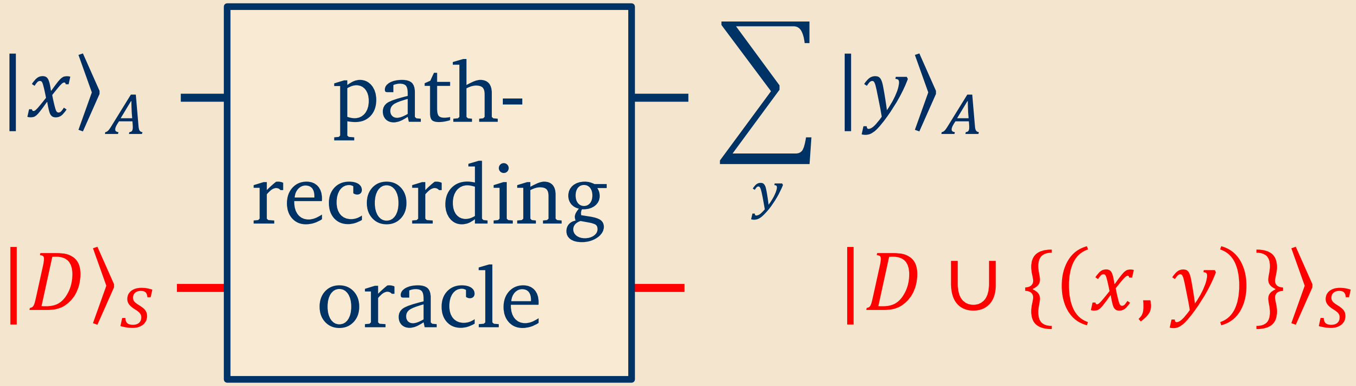




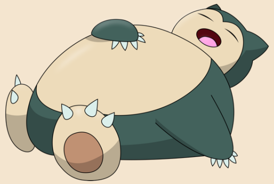
Note the similarity to classical lazy sampling:

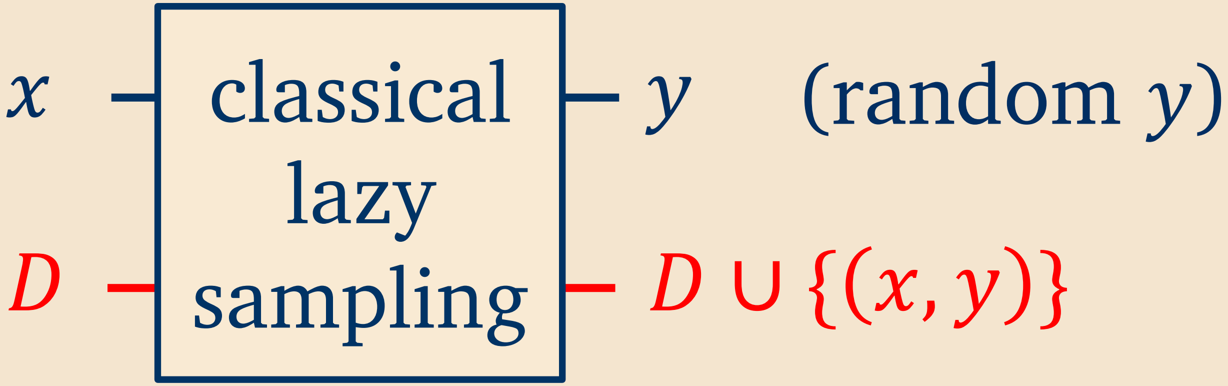
We can generate all of these examples by simply replacing each query to U with this:





Note the similarity to classical lazy sampling:





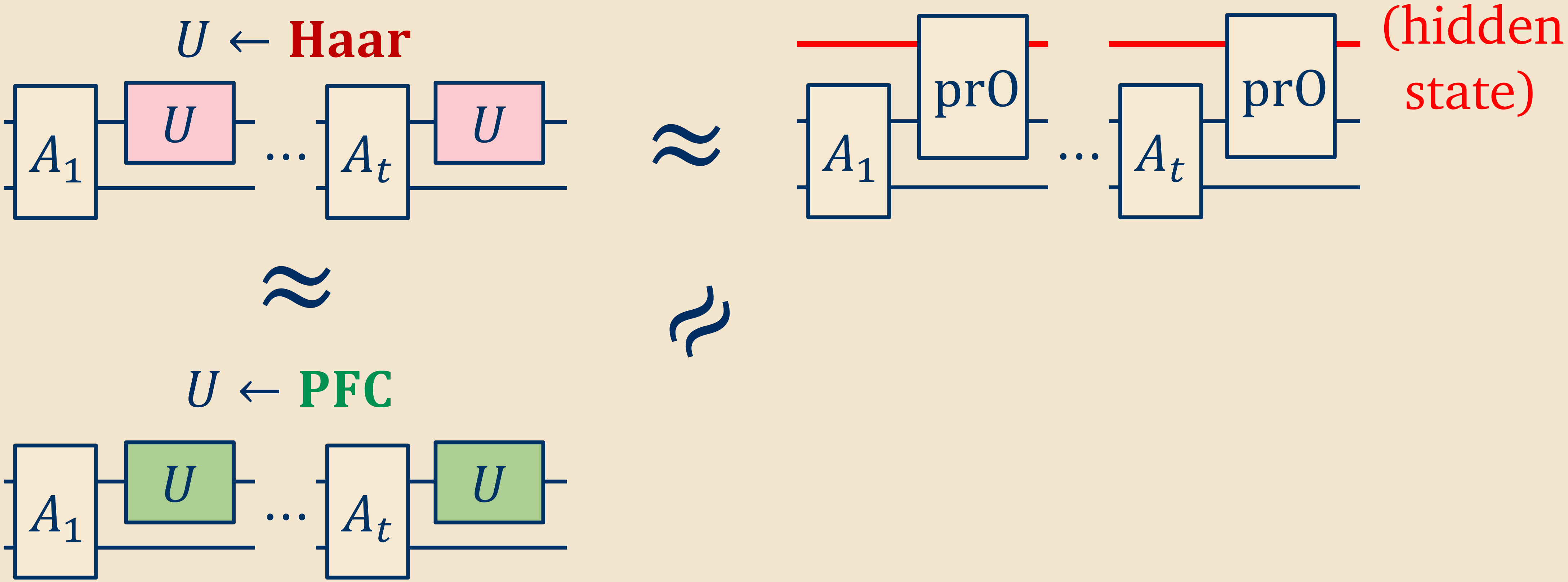
Rest of this talk

- Lazy sampling of a random function
- Lazy sampling of a random unitary
- **Proving correctness + PRUs exist**
- Applications

Recall our cartoon proof overview

Want to show:
For all efficient adversaries A,

Proof strategy: show that both are indistinguishable from

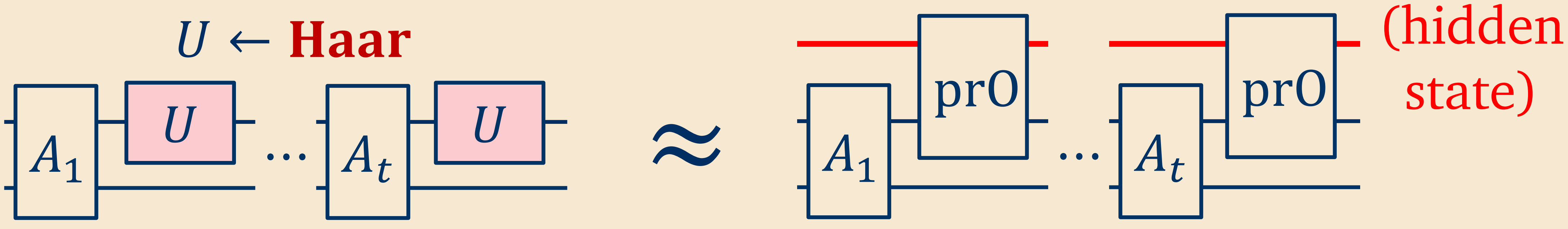


Recall our cartoon proof overview

Want to show:
For all efficient adversaries A,

Proof strategy: show that both
are indistinguishable from

Up next: prove this

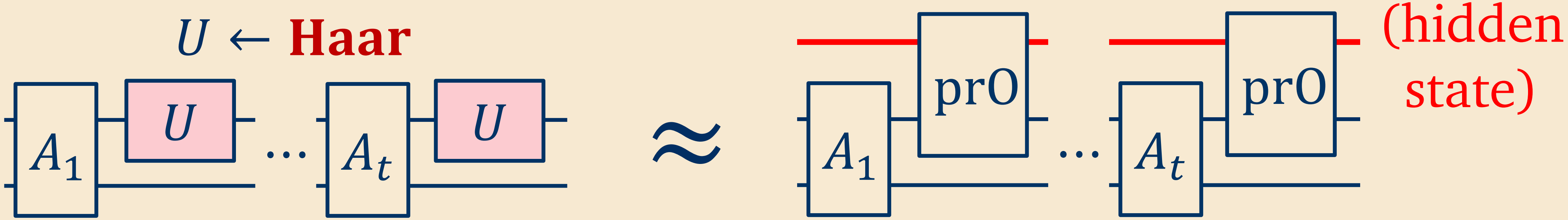


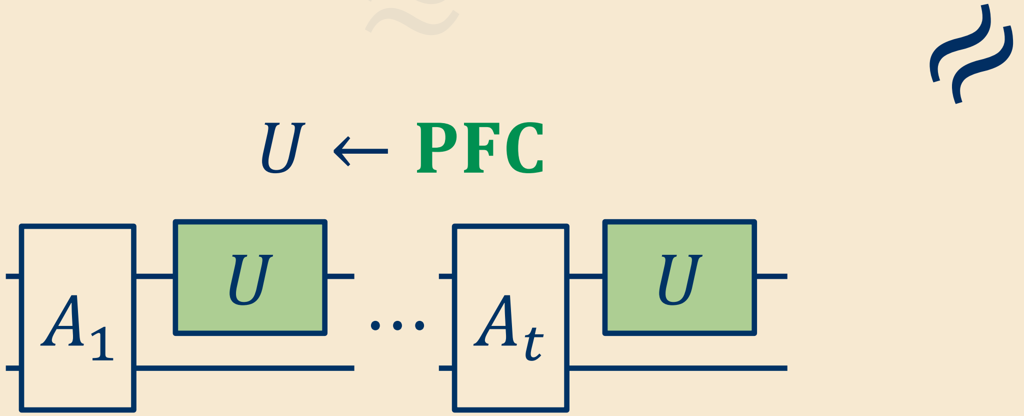
Recall our cartoon proof overview

Want to show:
For all efficient adversaries A,

Proof strategy: show that both
are indistinguishable from

Up next: prove this



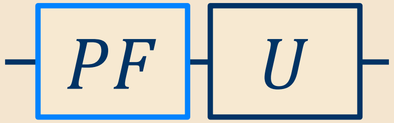


The same proof will show this!

$U \leftarrow \text{Haar}$

Hybrid 0

$U \leftarrow \text{Haar}$



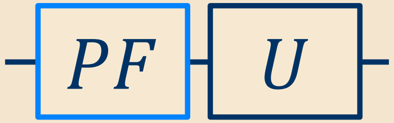
$U \leftarrow \text{Haar}$

$P \leftarrow S_N$

$F \leftarrow \{\pm 1\}^N$

Hybrid 0 $\equiv$ Hybrid 1

$U \leftarrow \text{Haar}$



$U \leftarrow \text{Haar}$

$P \leftarrow S_N$

$F \leftarrow \{\pm 1\}^N$

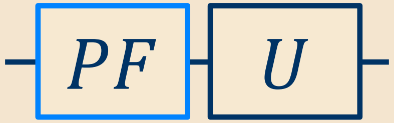
Hybrid 0 $\equiv$ Hybrid 1

Step 1: insert random permutation P random ± 1 diagonal F.

$$P = \begin{pmatrix} & & 1 \\ 1 & & \\ & 1 & \end{pmatrix} \qquad F = \begin{pmatrix} +1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$$

$U \leftarrow \text{Haar}$

Hybrid 0 $\equiv$

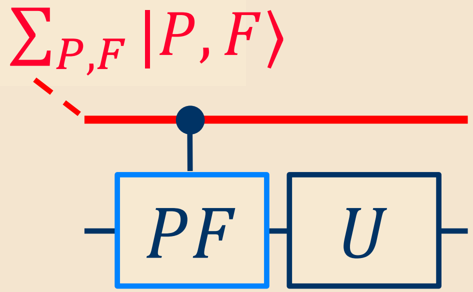


$U \leftarrow \text{Haar}$

$P \leftarrow S_N$

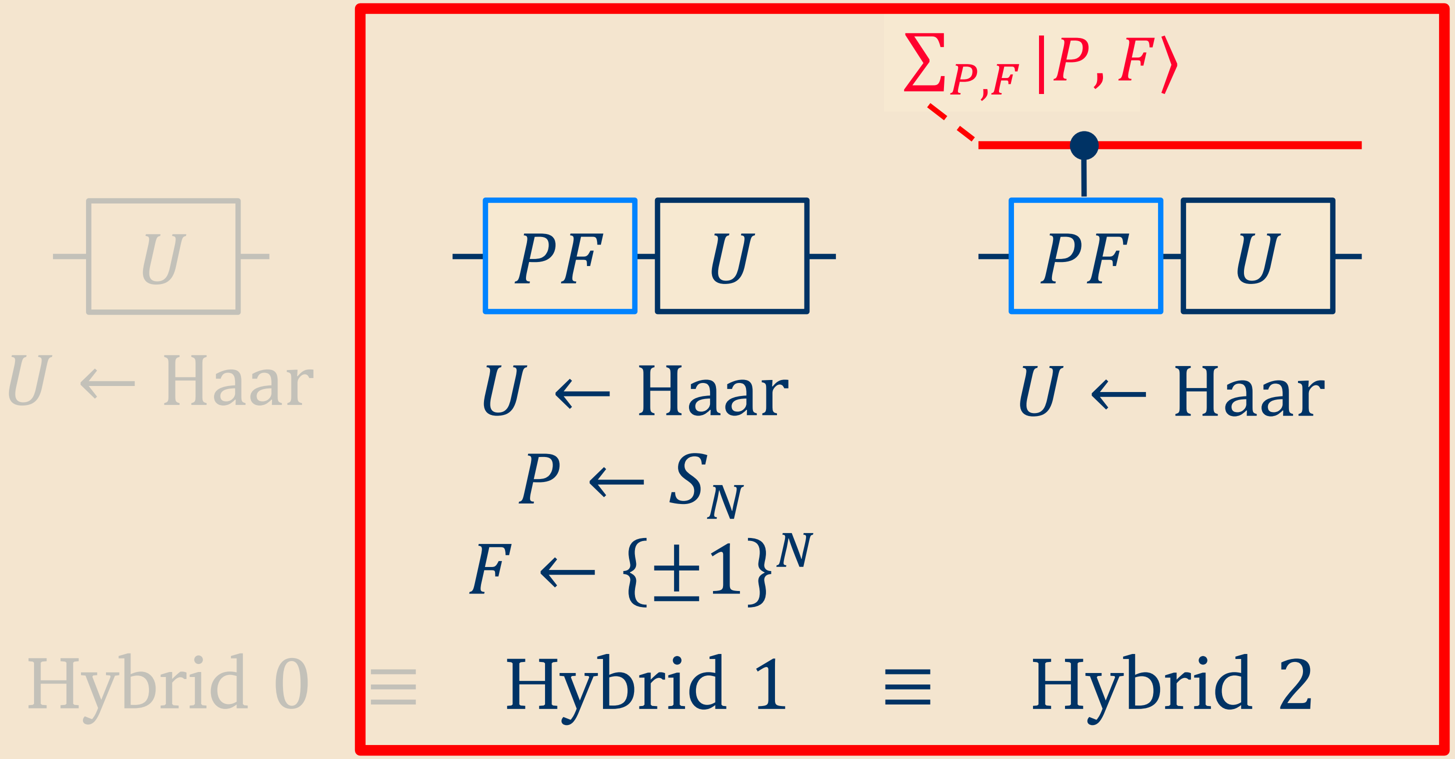
$F \leftarrow \{\pm 1\}^N$

Hybrid 1 $\equiv$



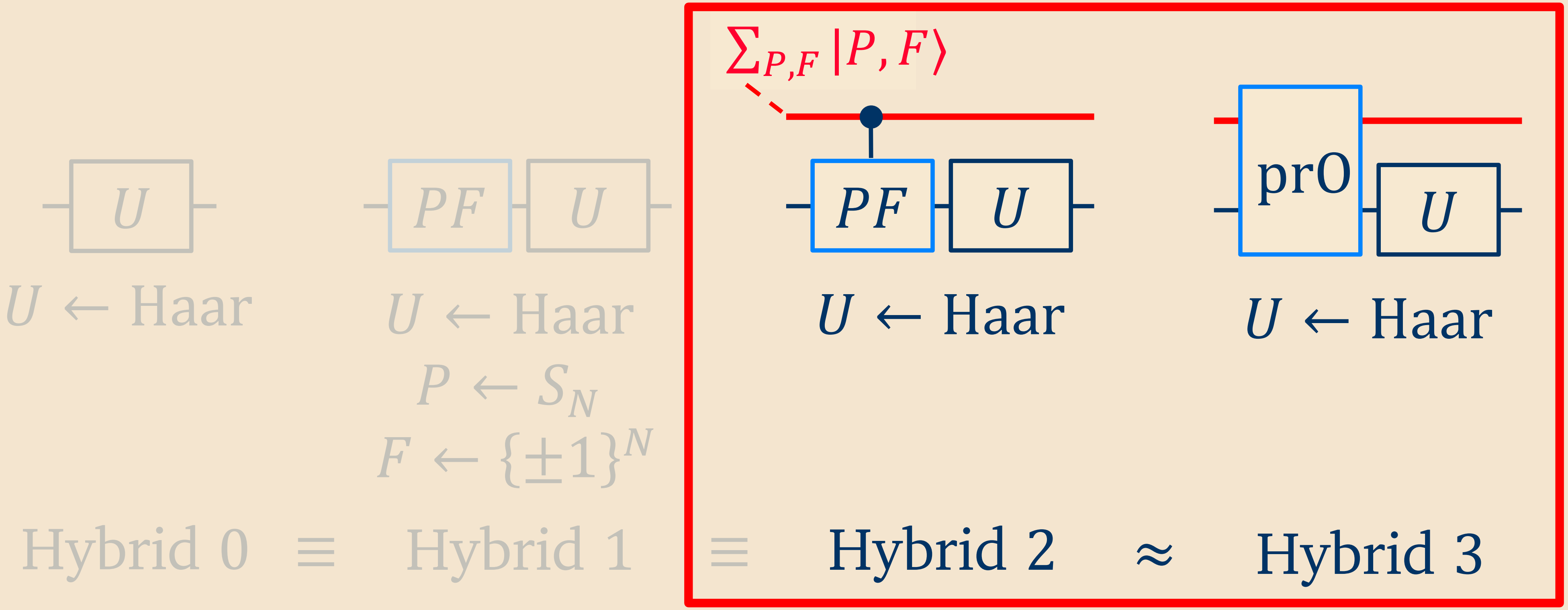
$U \leftarrow \text{Haar}$

Hybrid 2

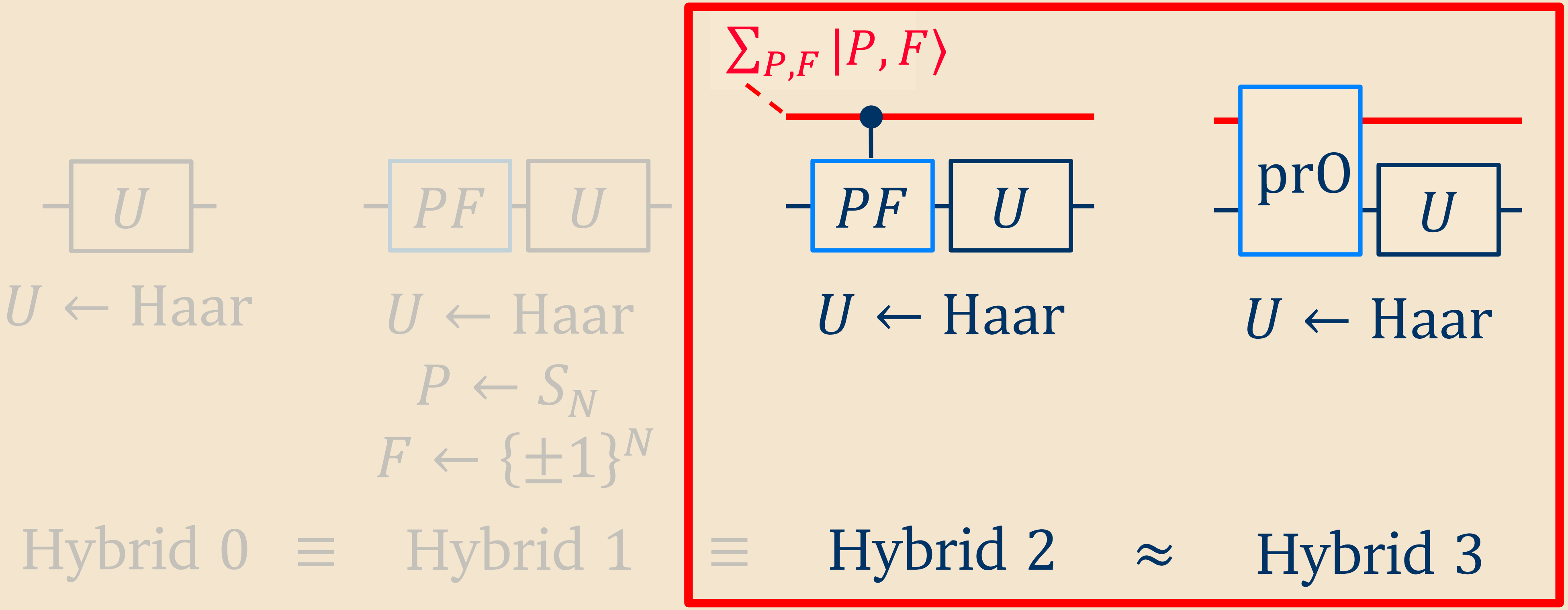


Step 2: replace random P, F with a purification.

- Initialize external/ancilla system to $\sum_{P,F} |P,F\rangle$
- On each query, apply $P \cdot F$ **controlled** on $|P,F\rangle$



U
U ← Haar
PF
U
U ← Haar
P ← S_N
F ← {±1}^N
Hybrid 0 ≡ Hybrid 1 ≡ Hybrid 2 ≈ Hybrid 3
∑_{P,F} |P, F⟩
PF
U
U ← Haar
prO
U
U ← Haar



Step 3: Key idea: analyze ctl-PF in a different basis.

Let's see how this works…

(1) ctl-PF: $|x\rangle \otimes |P, F\rangle \mapsto (-1)^{F(x)}|P(x)\rangle \otimes |P, F\rangle$

(1) ctl-PF: $$|x\rangle \otimes |P, F\rangle \mapsto (-1)^{F(x)}|P(x)\rangle \otimes |P, F\rangle$$

(2) ctl-PF: $$|x\rangle \otimes |P, F\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{F(x)} \cdot \delta_{P(x)=y} \, |P, F\rangle$$

(1) ctl-PF: $$|x\rangle \otimes |P, F\rangle \mapsto (-1)^{F(x)} |P(x)\rangle \otimes |P, F\rangle$$

(2) ctl-PF: $$|x\rangle \otimes |P, F\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{F(x)} \cdot \delta_{P(x)=y} \, |P, F\rangle$$

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

$$|\Phi_D\rangle := \sum_{P,F} (-1)^{F(x_1)+\cdots+F(x_t)} \cdot \delta_{P(x_1)=y_1} \cdots \delta_{P(x_t)=y_t} \, |P, F\rangle$$

(1) ctl-PF: $$|x\rangle \otimes |P,F\rangle \mapsto (-1)^{F(x)}|P(x)\rangle \otimes |P,F\rangle$$

(2) ctl-PF: $$|x\rangle \otimes |P,F\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{F(x)} \cdot \delta_{P(x)=y}\,|P,F\rangle$$

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

$$|\Phi_D\rangle := \sum_{P,F} (-1)^{F(x_1)+\cdots+F(x_t)} \cdot \delta_{P(x_1)=y_1} \cdots \delta_{P(x_t)=y_t}\,|P,F\rangle$$

ctl-PF: $$|x\rangle \otimes |\Phi_D\rangle \mapsto \sum_y |y\rangle \otimes \left|\Phi_{D\cup\{(x,y)\}}\right\rangle$$

(1) ctl-PF: $|x\rangle \otimes |P,F\rangle \mapsto (-1)^{F(x)}|P(x)\rangle \otimes |P,F\rangle$

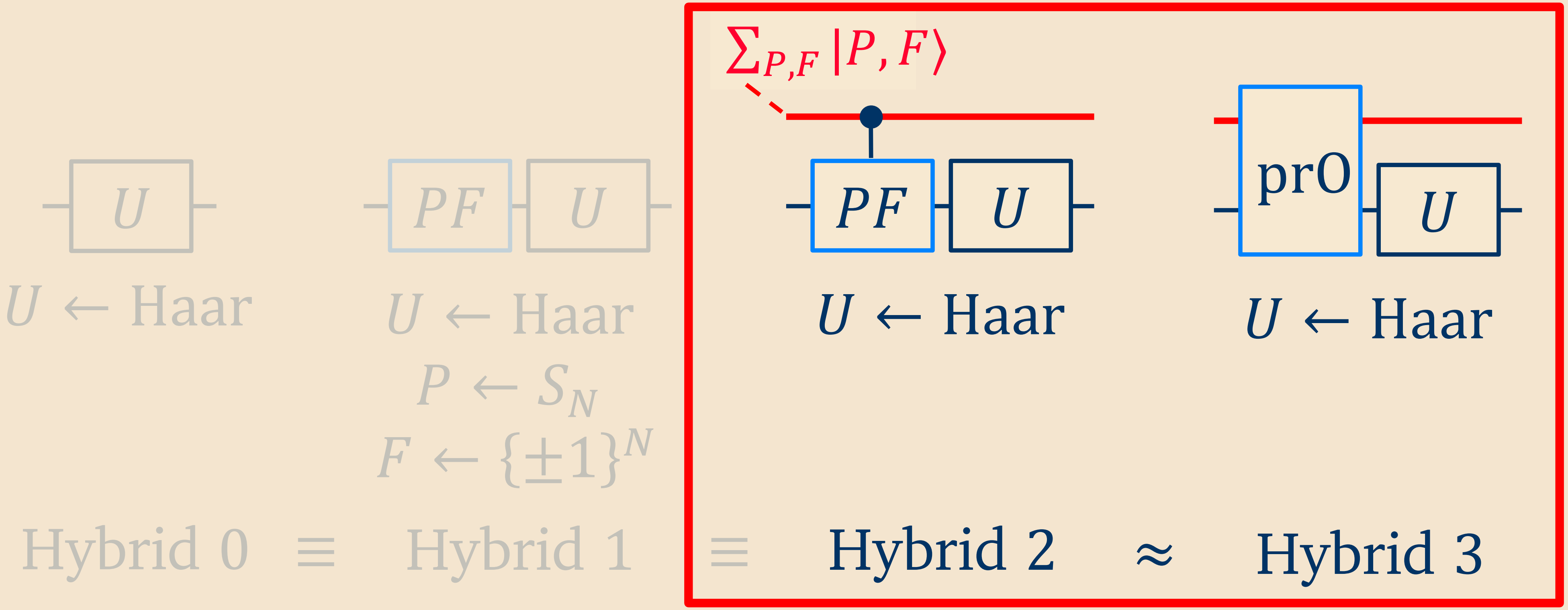
(2) ctl-PF: $|x\rangle \otimes |P,F\rangle \mapsto \sum_y |y\rangle \otimes (-1)^{F(x)} \cdot \delta_{P(x)=y} \, |P,F\rangle$

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

$$|\Phi_D\rangle := \sum_{P,F} (-1)^{F(x_1)+\cdots+F(x_t)} \cdot \delta_{P(x_1)=y_1} \cdots \delta_{P(x_t)=y_t} \, |P,F\rangle$$

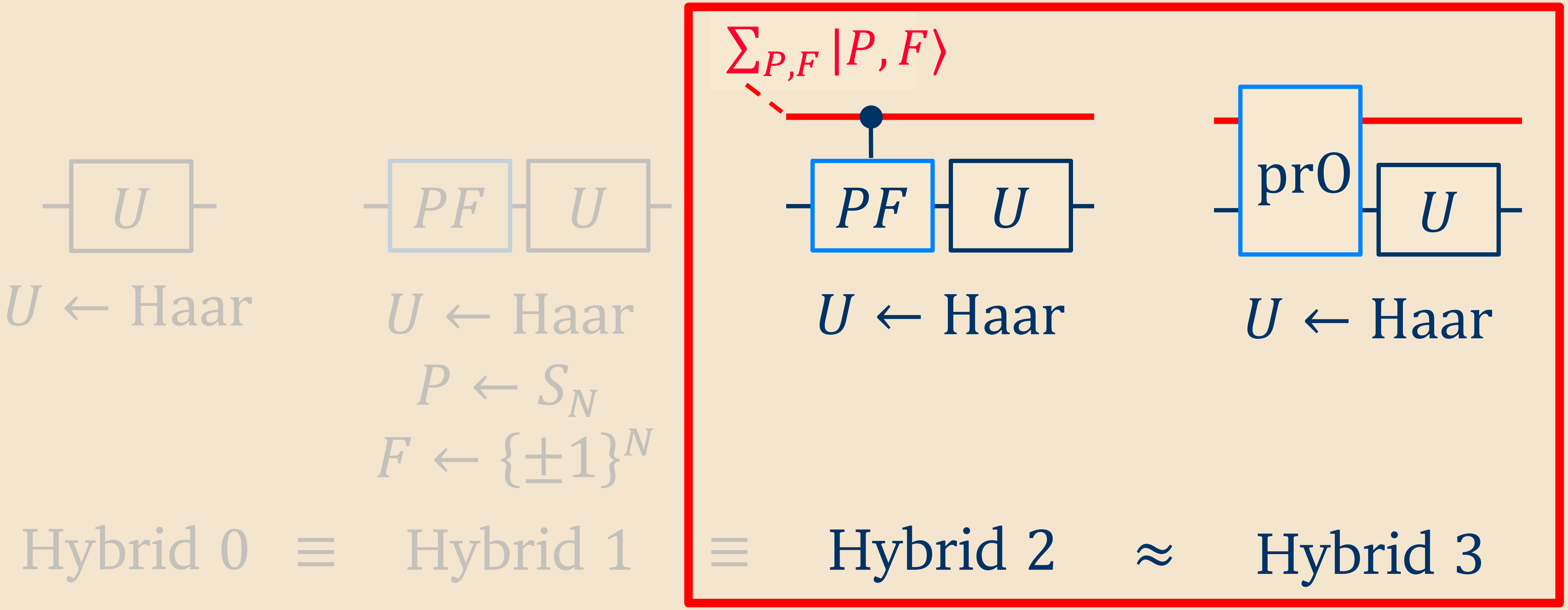
ctl-PF: $|x\rangle \otimes |\Phi_D\rangle \mapsto \sum_y |y\rangle \otimes \left|\Phi_{D\cup\{(x,y)\}}\right\rangle$

pr0: $|x\rangle \otimes |D\rangle \mapsto \sum_y |y\rangle \otimes |D \cup \{(x,y)\}\rangle$



Step 3: For any $D = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ can define $|\Phi_D\rangle$ s.t.

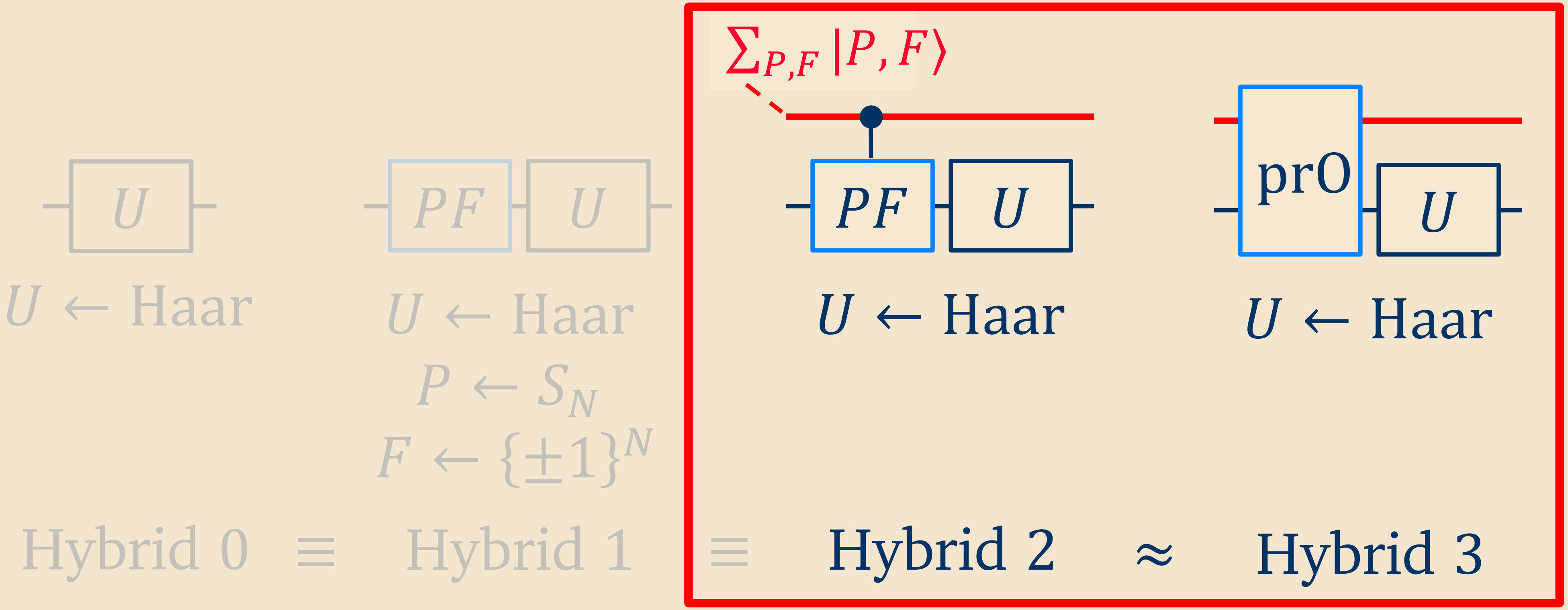
$$\text{ctl-PF} \cdot |x\rangle|\Phi_D\rangle = \sum_y |y\rangle|\Phi_{D\cup\{(x,y)\}}\rangle$$



Step 3: For any $D = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ can define $|\Phi_D\rangle$ s.t.

$$\text{ctl-PF} \cdot |x\rangle|\Phi_D\rangle = \sum_y |y\rangle|\Phi_{D\cup\{(x,y)\}}\rangle$$

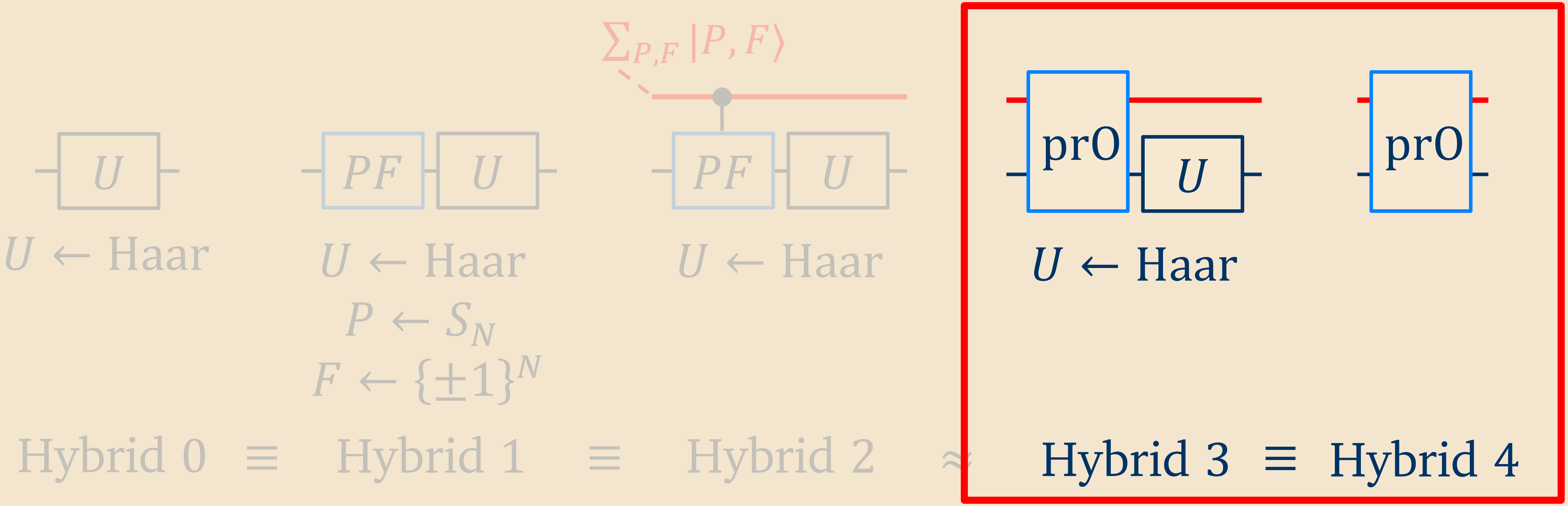
- **Intuition:** ctl-PF behaves like prO, up to relabeling $|\Phi_D\rangle \mapsto |D\rangle$

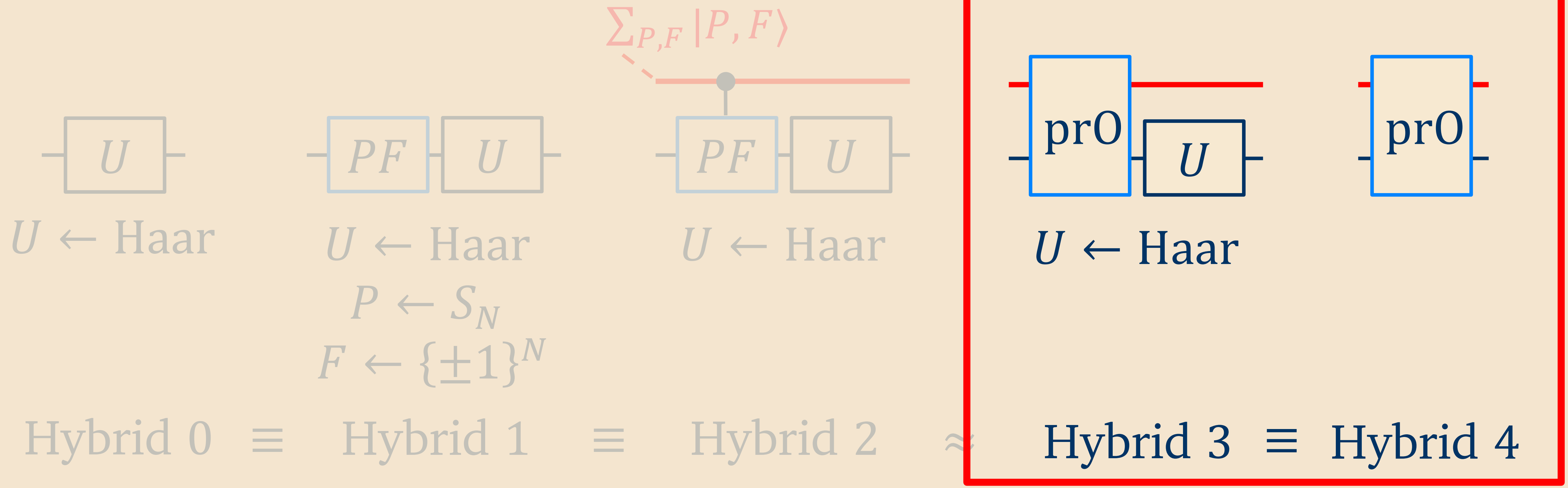


Step 3: For any $D = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ can define $|\Phi_D\rangle$ s.t.

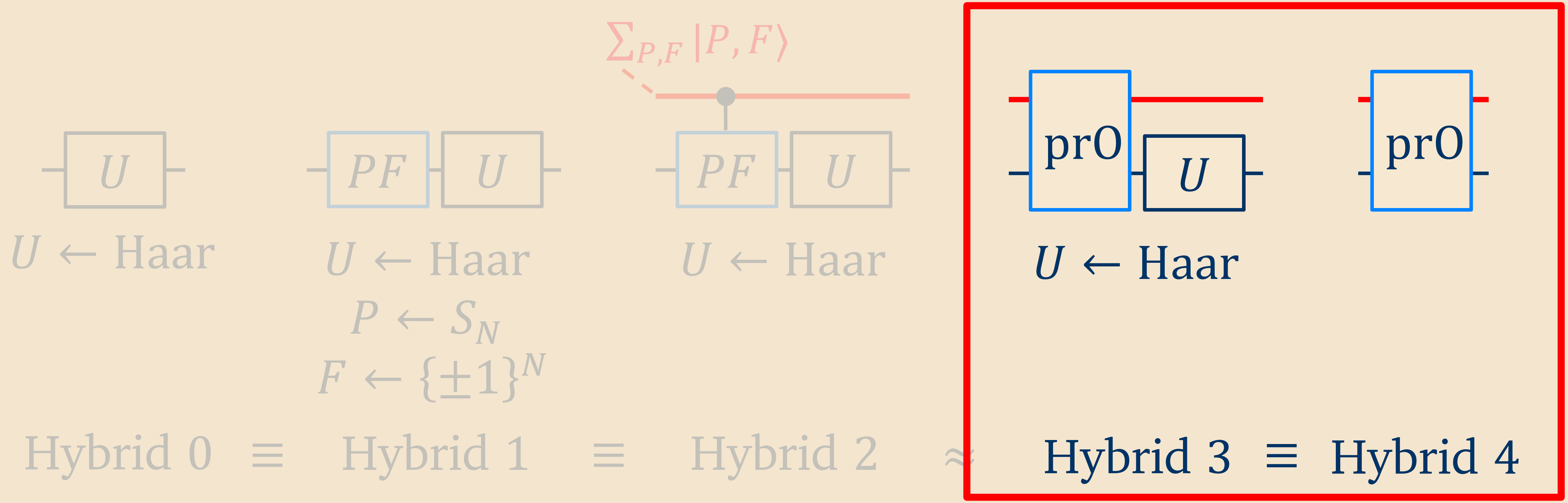
$$\text{ctl-PF} \cdot |x\rangle|\Phi_D\rangle = \sum_y |y\rangle|\Phi_{D\cup\{(x,y)\}}\rangle$$

- **Intuition:** ctl-PF behaves like prO, up to relabeling $|\Phi_D\rangle \mapsto |D\rangle$
- Actually, $\{|\Phi_D\rangle\}_D$ aren't fully orthogonal. But composing with $U \leftarrow$ (2-design) makes the "non-orthogonal" ones hard to find.

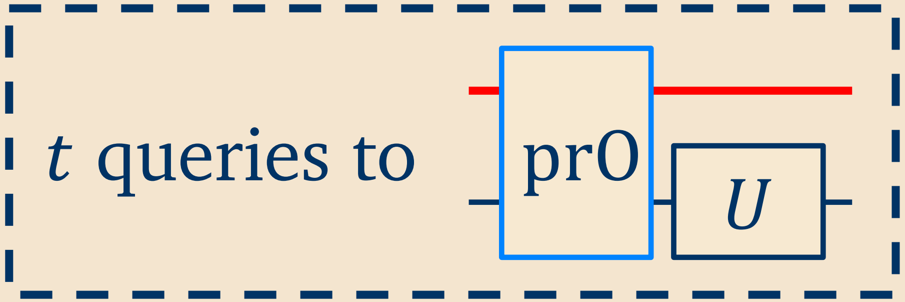

$\sum_{P,F} |P, F\rangle$
U
$U \leftarrow$ Haar
PF
U
$U \leftarrow$ Haar
$P \leftarrow S_N$
$F \leftarrow \{\pm 1\}^N$
PF
U
$U \leftarrow$ Haar
prO
U
$U \leftarrow$ Haar
prO
Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4

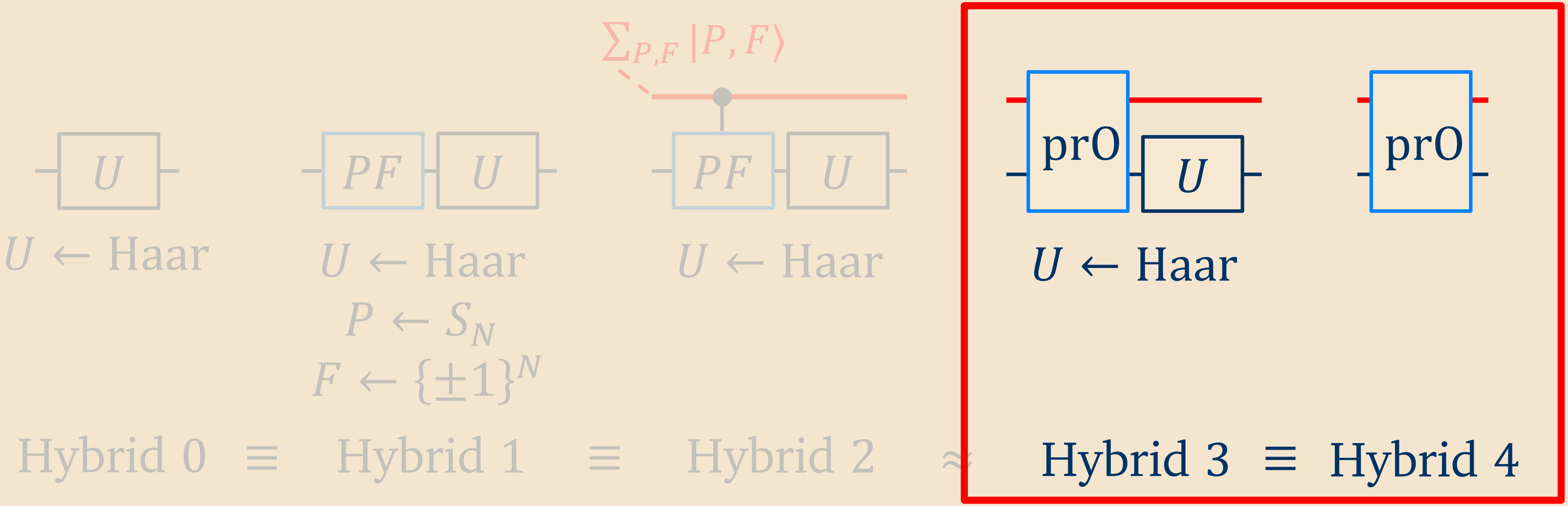


Step 4: Turns out prO has the following unitary invariance property:

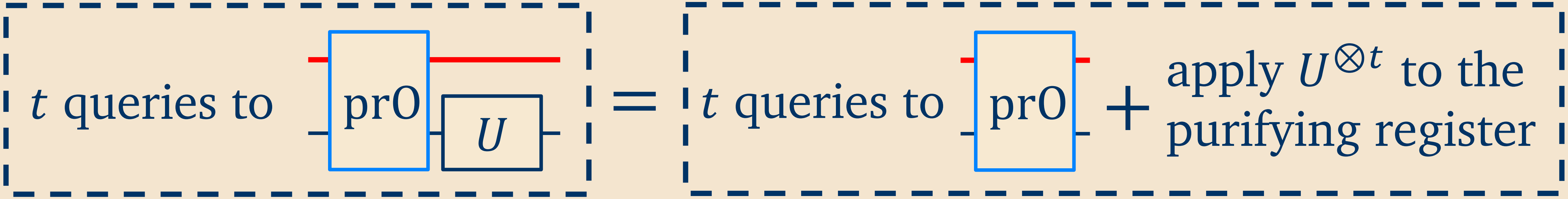


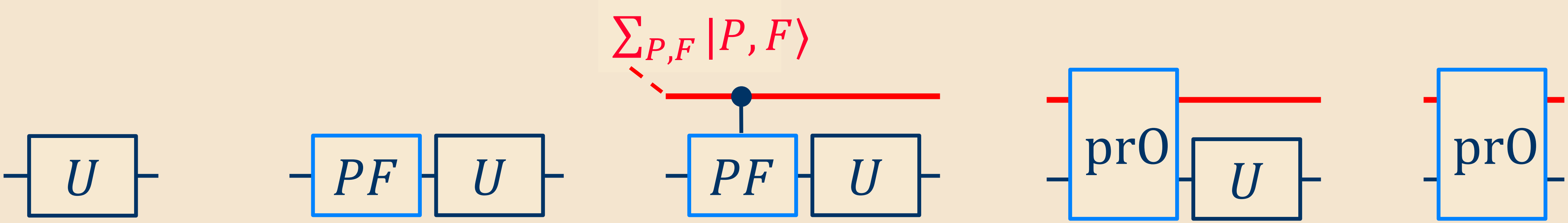
Step 4: Turns out prO has the following unitary invariance property:





Step 4: Turns out prO has the following unitary invariance property:





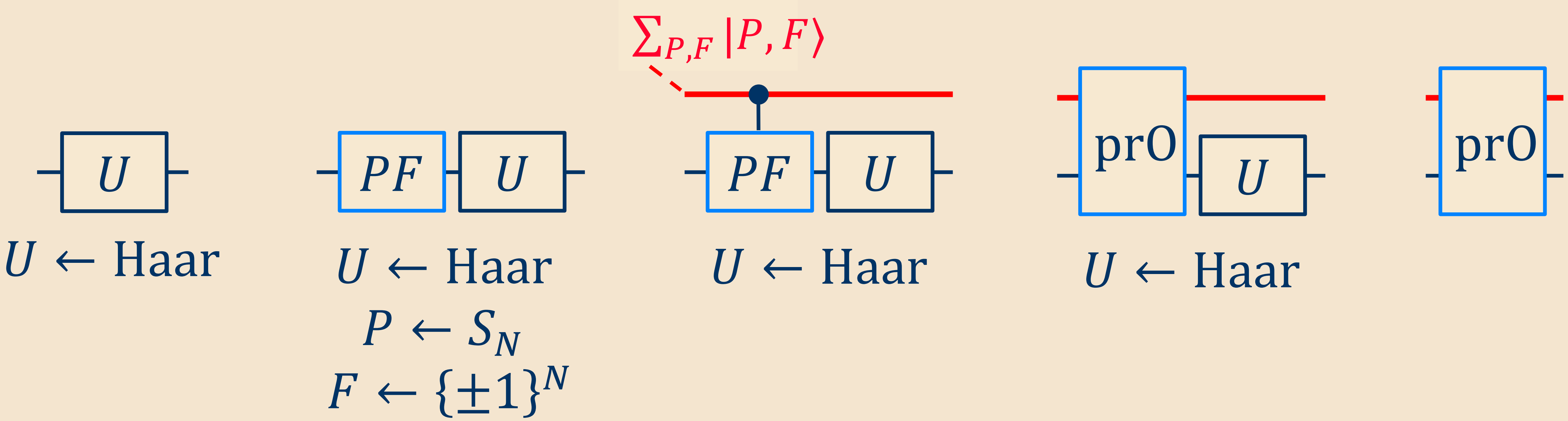
$U \leftarrow$ Haar

$U \leftarrow$ Haar
$P \leftarrow S_N$
$F \leftarrow \{\pm 1\}^N$

$U \leftarrow$ Haar

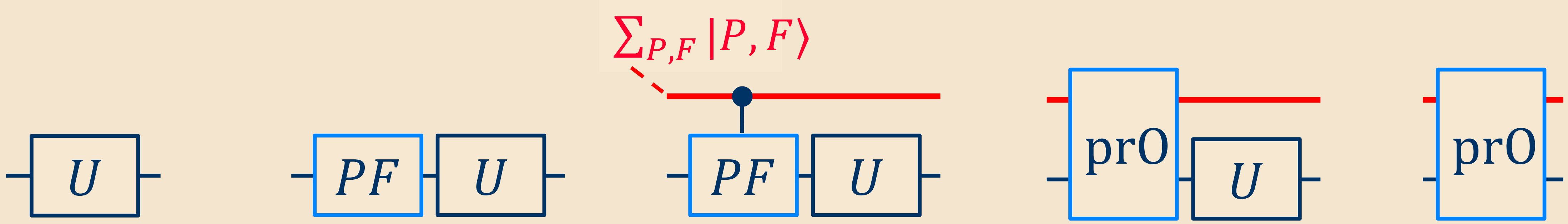
$U \leftarrow$ Haar

Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4



Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4

The PRU proof: Hybrid 2 $\approx$ Hybrid 4 holds for any **2-design**.

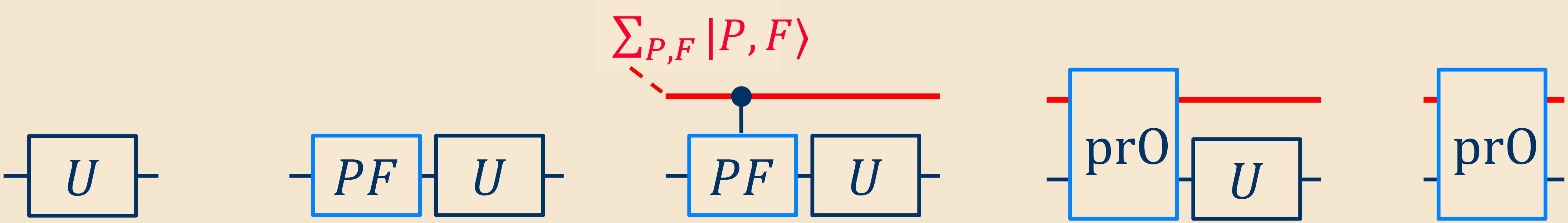


$U \leftarrow$ Haar | $U \leftarrow$ Haar, $P \leftarrow S_N$, $F \leftarrow \{\pm 1\}^N$ | $U \leftarrow$ Haar | $U \leftarrow$ Haar

Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4

The PRU proof: Hybrid 2 $\approx$ Hybrid 4 holds for any **2-design**. So

$$PF \cdot (\text{Clifford C}) \approx PF \cdot (\text{Haar } U)$$



$U \leftarrow \text{Haar}$

$U \leftarrow \text{Haar}$
$P \leftarrow S_N$
$F \leftarrow \{\pm 1\}^N$

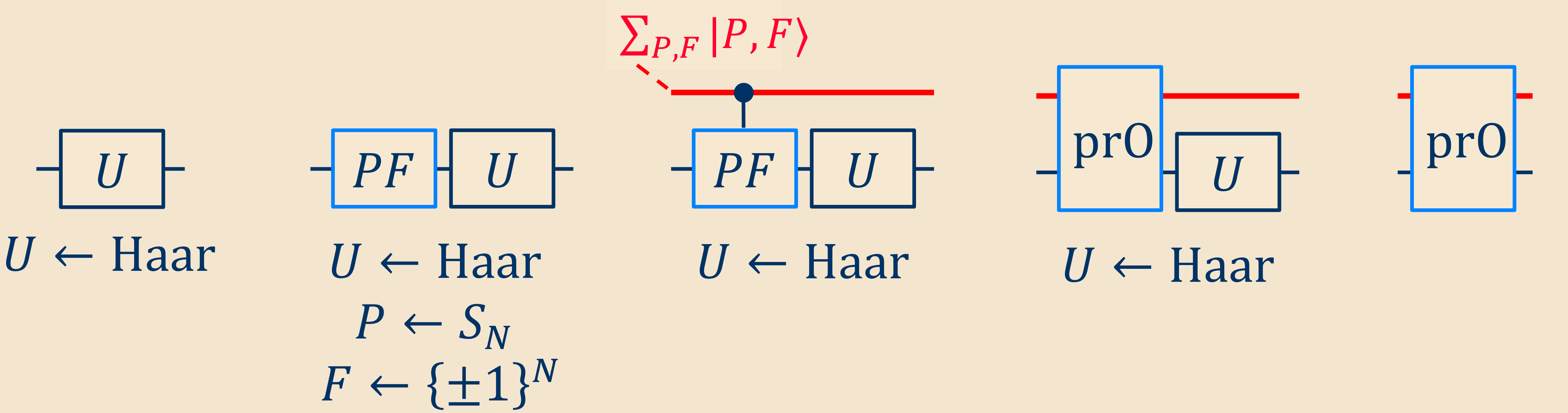
$U \leftarrow \text{Haar}$

$U \leftarrow \text{Haar}$

Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4

The PRU proof: Hybrid 2 $\approx$ Hybrid 4 holds for any **2-design**. So

$$PF \cdot (\text{Clifford } C) \approx PF \cdot (\text{Haar } U) \equiv \text{Haar } U$$



Hybrid 0 $\equiv$ Hybrid 1 $\equiv$ Hybrid 2 $\approx$ Hybrid 3 $\equiv$ Hybrid 4

The PRU proof: Hybrid 2 $\approx$ Hybrid 4 holds for any **2-design**. So

$$PF \cdot (\text{Clifford C}) \approx PF \cdot (\text{Haar } U) \equiv \text{Haar } U$$

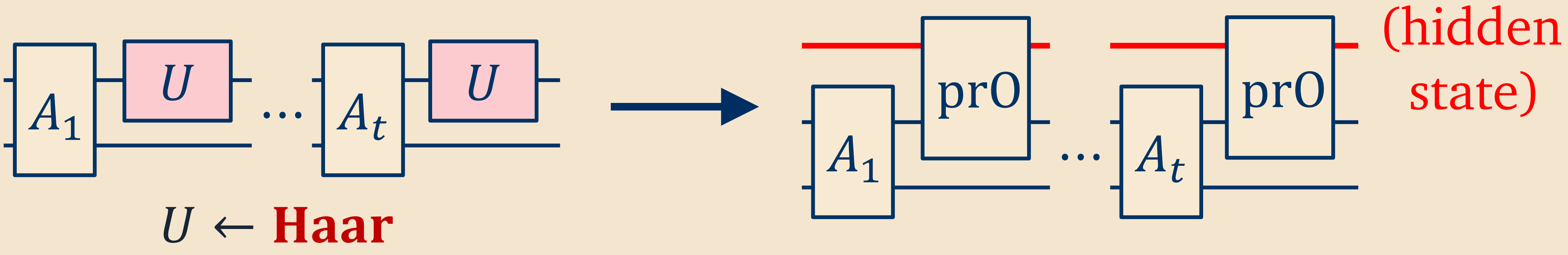
Finally, replace P and F with pseudorandom.

Rest of this talk

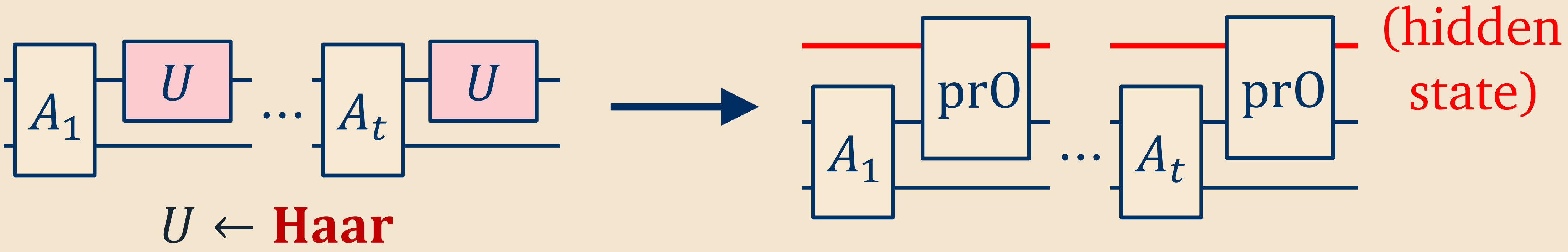
- Lazy sampling of a random function
- Lazy sampling of a random unitary
- Proving correctness + PRUs exist
- **Applications**

The path-recording oracle is a general-purpose tool for analyzing Haar-random unitaries.

The path-recording oracle is a general-purpose tool for analyzing Haar-random unitaries.

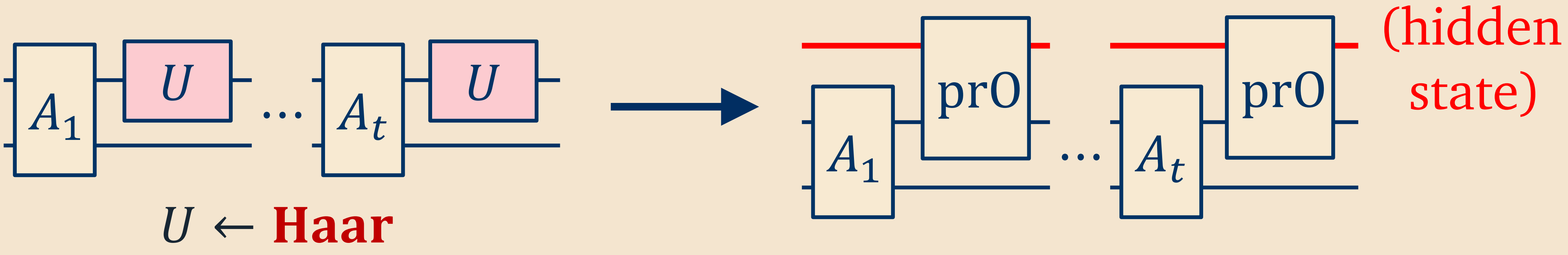


The path-recording oracle is a general-purpose tool for analyzing Haar-random unitaries.



Many statements about Haar-random U can be reduced to simple claims about this data structure

The path-recording oracle is a general-purpose tool for analyzing Haar-random unitaries.



Many statements about Haar-random U can be reduced to simple claims about this data structure

- [**M**H24]: elementary proof of [SHH24] gluing lemma
- [S**M**LBH25]: existence of low-depth PRUs

Let's see an example.

Application: a simpler proof of the “gluing” lemma

Application: a simpler proof of the "gluing" lemma

Gluing lemma [SHH24]:

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

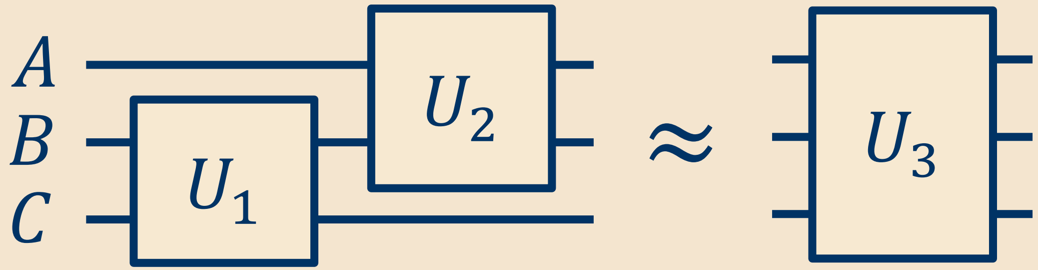
$$U_2 \cdot U_1 \approx U_3.$$

Application: a simpler proof of the "gluing" lemma

Gluing lemma [SHH24]:

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

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



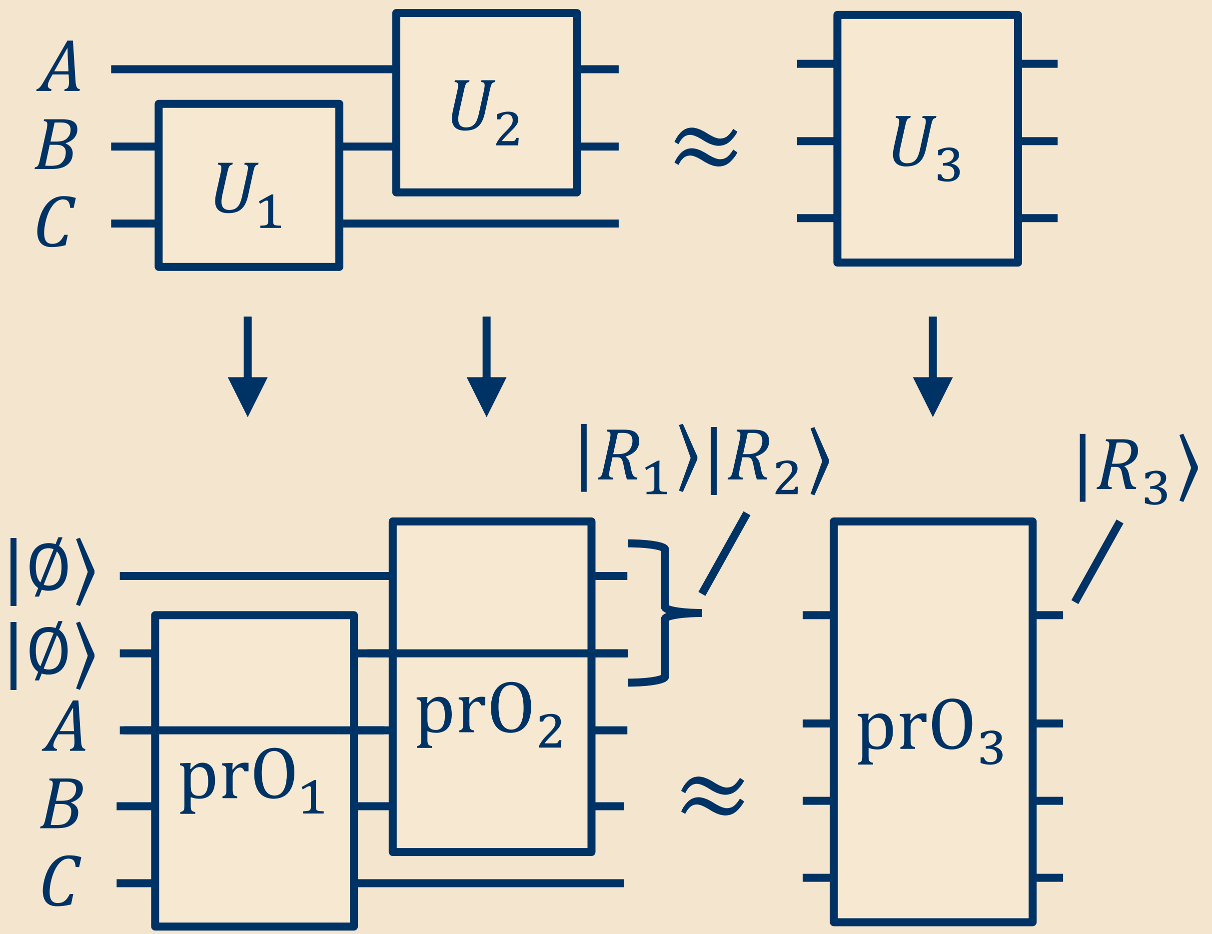
Application: a simpler proof of the "gluing" lemma

Gluing lemma [SHH24]:

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

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

New proof: combinatorial claim about path-recording oracle.



Future directions

Future directions

1) Is a random quantum circuit a PRU?

Future directions

1) Is a random quantum circuit a PRU?

2) Unitary natural proof barrier?

Future directions

1) Is a random quantum circuit a PRU?

2) Unitary natural proof barrier?

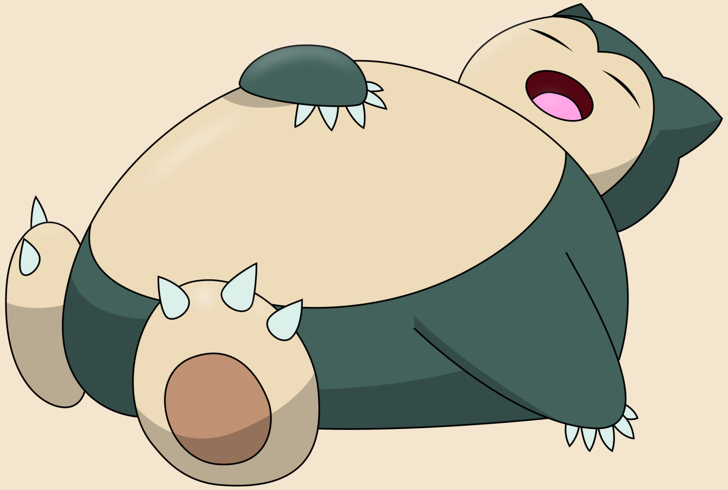
3) Cryptographic applications of PRUs?

Future directions

1) Is a random quantum circuit a PRU?

2) Unitary natural proof barrier?

3) Cryptographic applications of PRUs?



Thanks!