Quantum One-Time Programs, Revisited

Aparna Gupte
MIT

Jiahui Liu
Fujitsu Research

Justin Raizes
CMU

Bhaskar Roberts
UC Berkeley

Vinod Vaikuntanathan
MIT

One-Time Programs (OTP)

[Goldwasser-Kalai-Rothblum'08]

I grant you one evaluation!

Image generated by ChatGPT

One-Time Programs (OTP)

[Goldwasser-Kalai-Rothblum'08]

I grant you one evaluation!

Program

$f : X \to Y$

Image generated by ChatGPT

One-Time Programs (OTP)

[Goldwasser-Kalai-Rothblum'08]

I grant you one evaluation!

Program

$f : X \to Y$

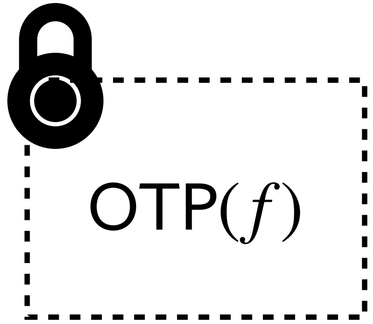


Image generated by ChatGPT

One-Time Programs (OTP)

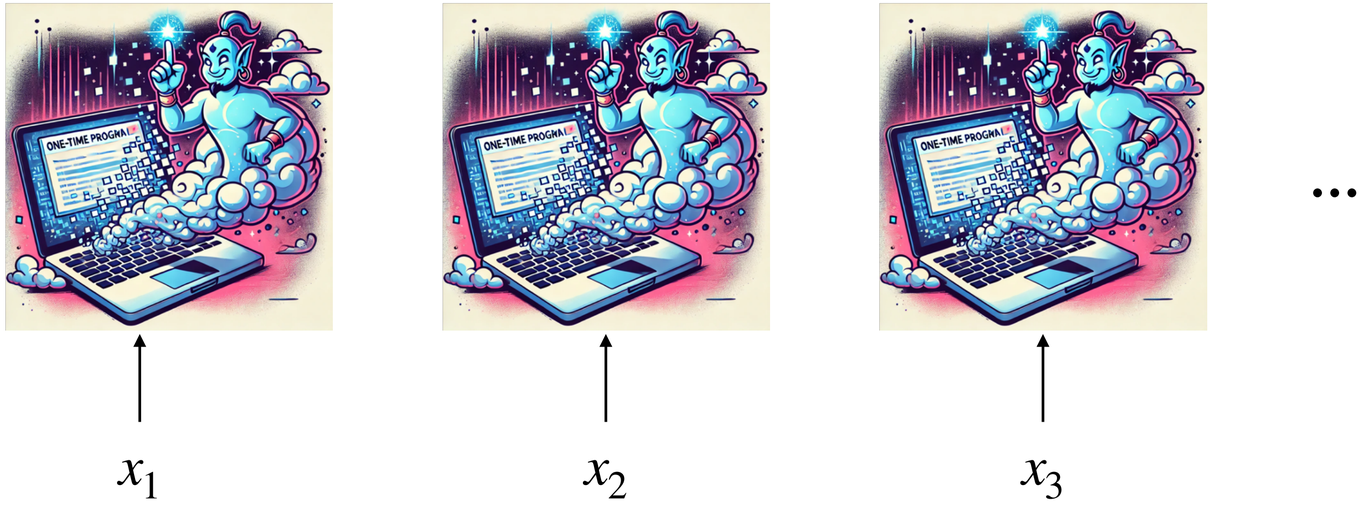
[Goldwasser-Kalai-Rothblum'08]

- Classically impossible 😢 (without hardware assumptions)

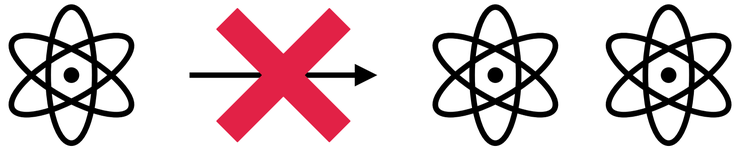
One-Time Programs (OTP)

[Goldwasser-Kalai-Rothblum'08]

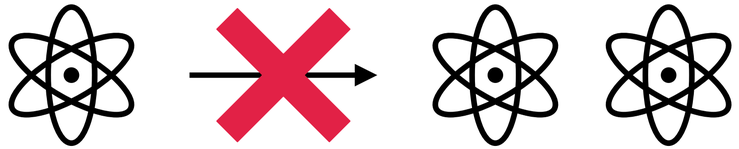
- Classically impossible 😢 (without hardware assumptions)
  - Can copy program many times, evaluate each copy



Quantum One-Time Programs?

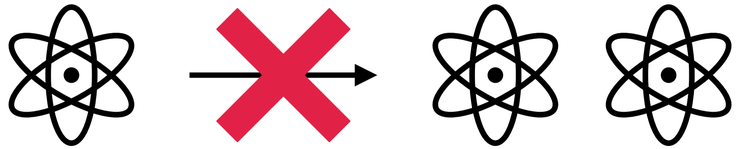
Quantum No-cloning Theorem: 

Quantum One-Time Programs?

Quantum No-cloning Theorem: 

Does quantum no-cloning enable *quantum* one-time programs?

Quantum One-Time Programs?

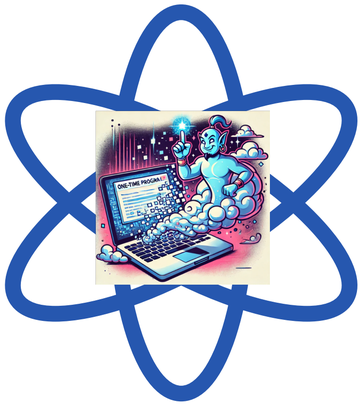
Quantum No-cloning Theorem: 

Does quantum no-cloning enable *quantum* one-time programs?

Classical program

$f : X \to Y$

$|\mathsf{OTP}_f\rangle$



Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible

Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible

$|\mathrm{OTP}_f\rangle$

Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible

$|x\rangle_{\text{in}} |\text{OTP}_f\rangle$

Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible

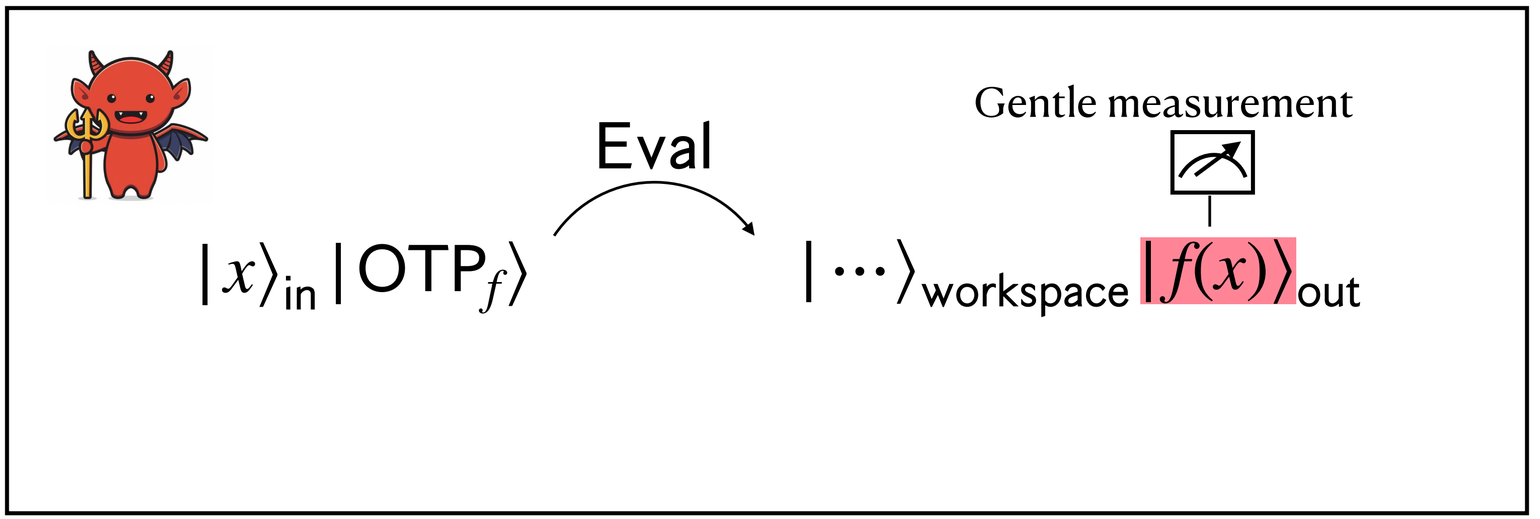
Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible



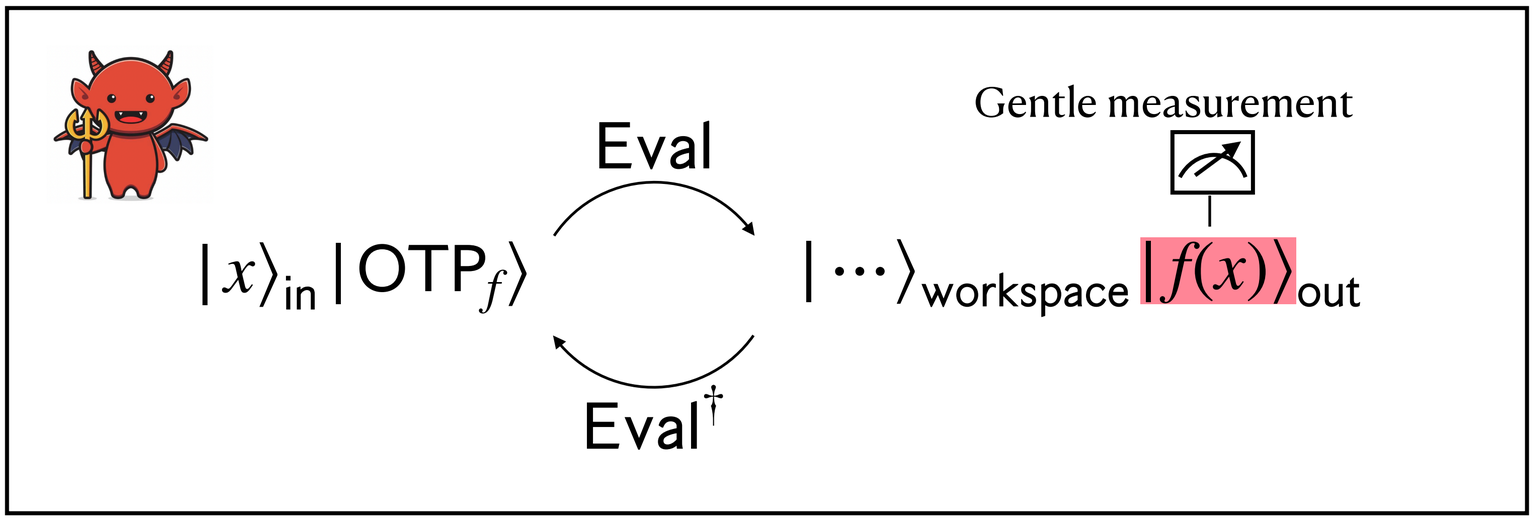
Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible



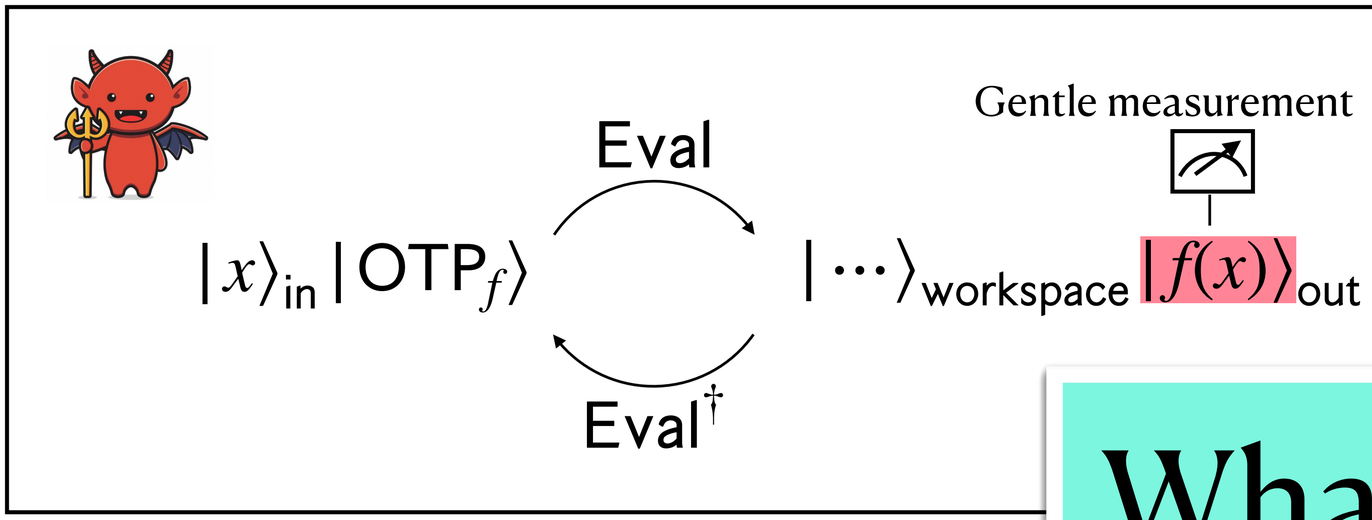
Quantum One-Time Programs?

No 😢

[Broadbent-Gutoski-Stebila '13]

No Quantum OTP for deterministic programs

Quantum computation is reversible



What next? 🤔

Quantum One-Time Programs?

for randomized programs

Quantum One-Time Programs?

Quantum One-Time Programs?

for randomized programs

Randomized function $f : X \times R \to Y$

- User chooses x but not r

Quantum One-Time Programs?

for randomized programs

Maybe? 🤔

Randomized function $f : X \times R \to Y$

- User chooses x but not r

for randomized programs

Quantum One-Time Programs?

Maybe? 🤔

Randomized function $f : X \times R \to Y$

- User chooses x but not r

Eval

$$|x\rangle_{\text{in}} |\text{OTP}_f\rangle \qquad \sum_r |\cdots_r\rangle_{\text{workspace}} |f(x,r)\rangle_{\text{out}}$$

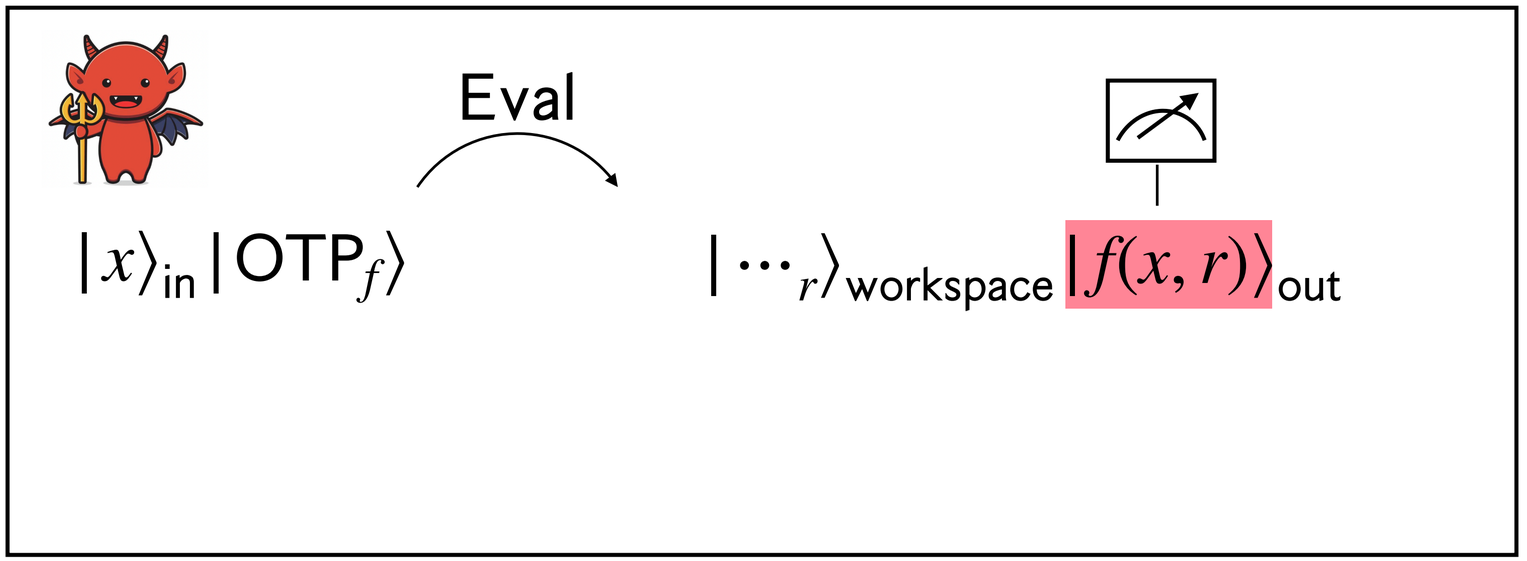
for randomized programs

Quantum One-Time Programs?

Randomized function $f : X \times R \to Y$

- User chooses x but not r

Maybe? 🤔



Attack doesn't work! For f with high min-entropy,
measuring output may destroy state and prevent further evaluations

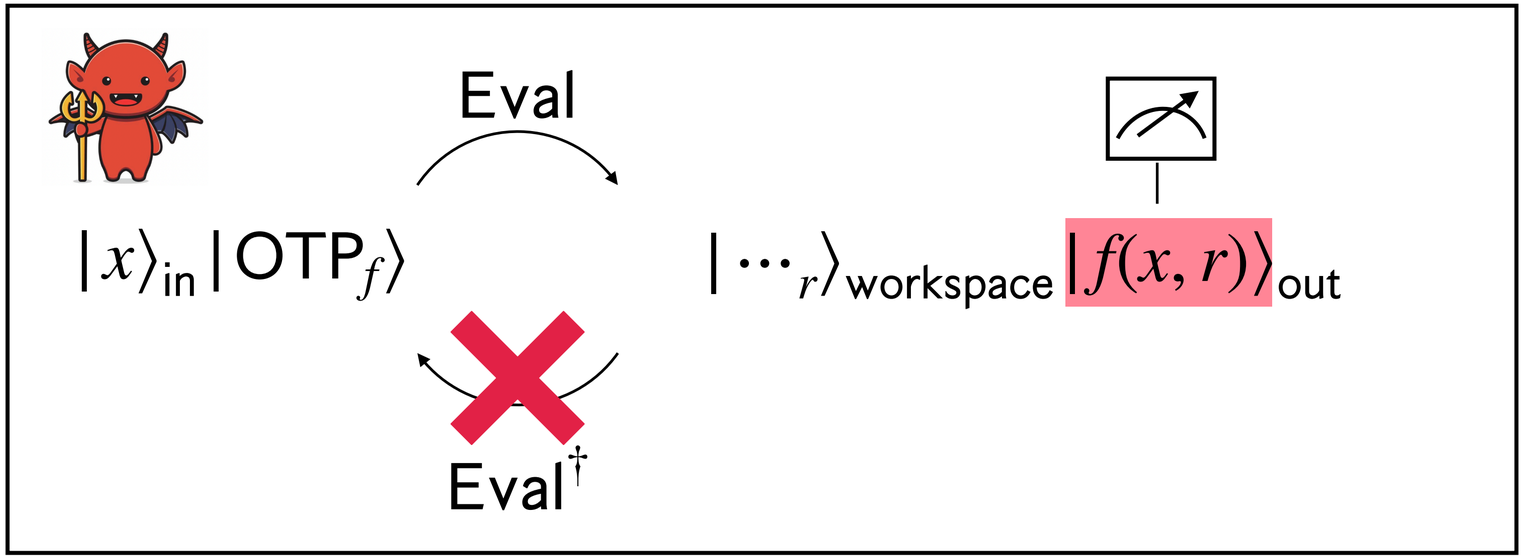
for randomized programs

Quantum One-Time Programs?

Randomized function $f : X \times R \to Y$

- User chooses x but not r

Maybe? 🤔



Attack doesn't work! For f with high min-entropy,
measuring output may destroy state and prevent further evaluations

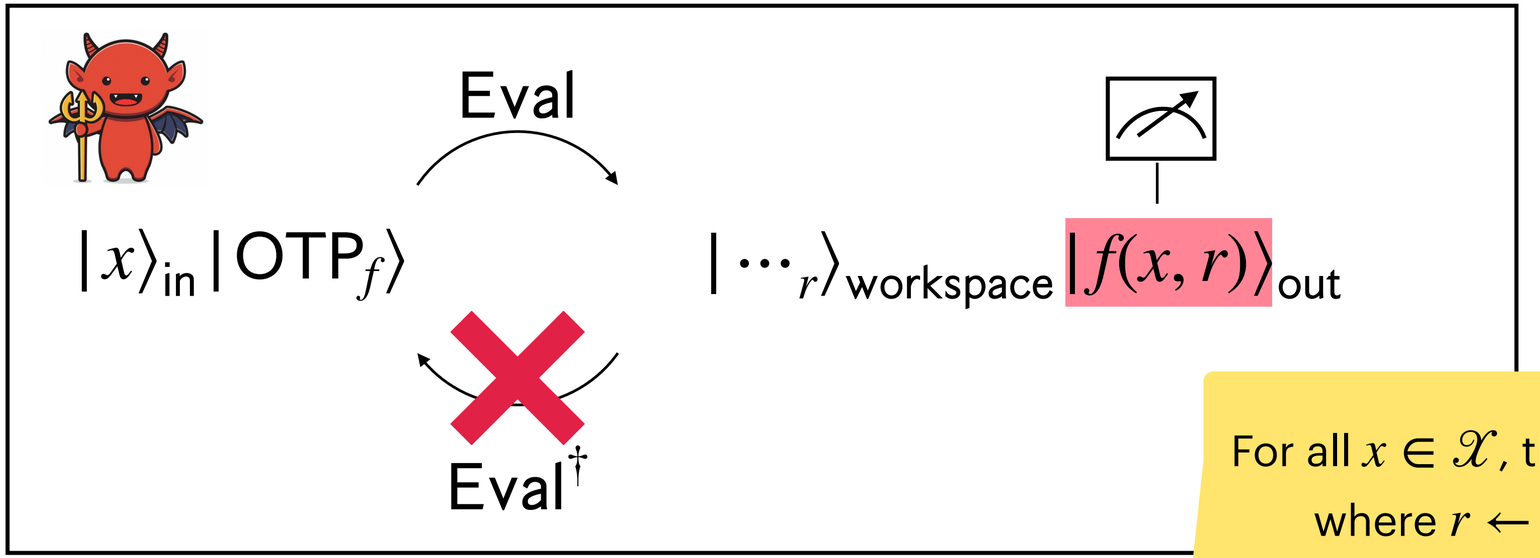
for randomized programs

Quantum One-Time Programs?

Maybe? 🤔

Randomized function $f : X \times R \to Y$

- User chooses x but not r



For all $x \in \mathcal{X}$, the distribution of $f(x, r)$ where $r \leftarrow R$ has high entropy

Attack doesn't work! For f with high min-entropy, measuring output may destroy state and prevent further evaluations

for randomized programs

Quantum One-Time Programs?

Maybe! 😀

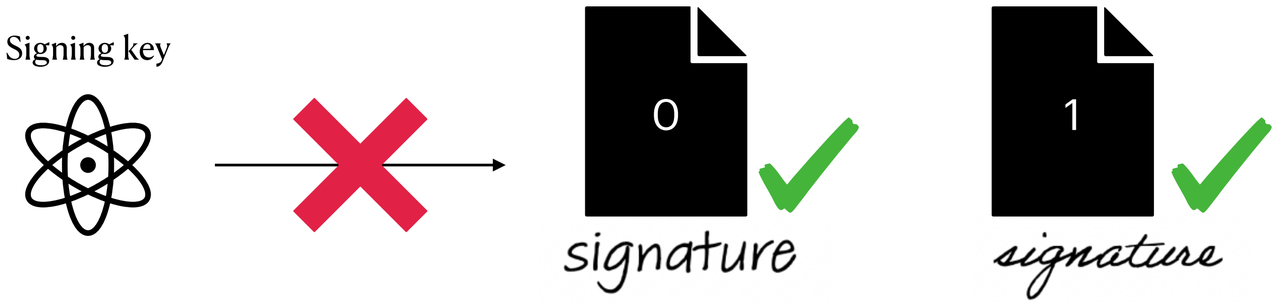
Signature tokens [Ben-David, Sattath '16]

for randomized programs

Quantum One-Time Programs?

Maybe! 😀

Signature tokens [Ben-David, Sattath '16]



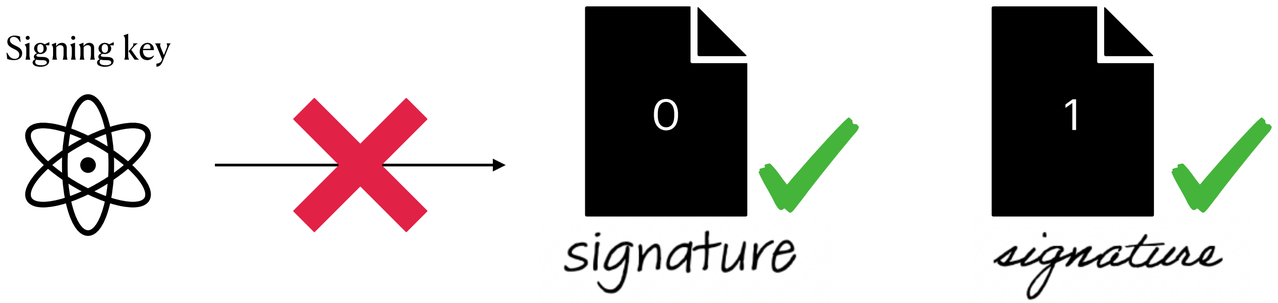
for randomized programs

Quantum One-Time Programs?

Maybe! 😀

Signature tokens [Ben-David, Sattath '16]

≈ quantum OTP
for a **specific** randomized program



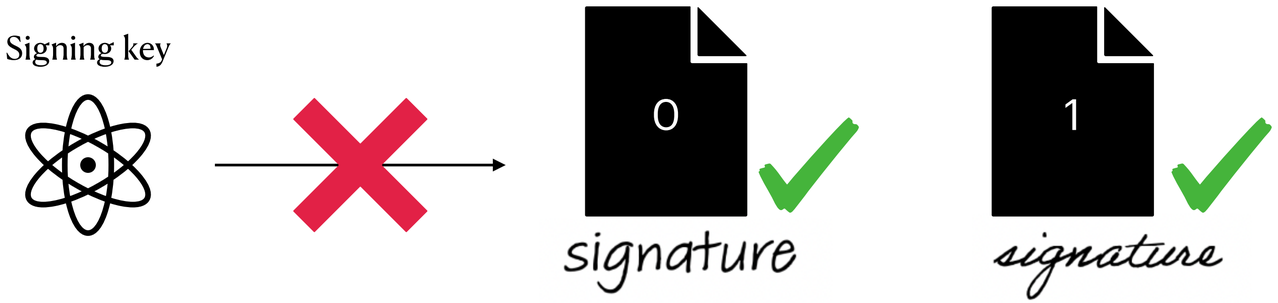
for randomized programs

Quantum One-Time Programs?

Maybe! 😀

Signature tokens [Ben-David, Sattath '16]

≈ quantum OTP
for a **specific** randomized program



But we cannot be too greedy! 😢

No quantum OTPs that satisfy a *strong simulation security* notion, for any (randomized) programs [Broadbent-Gutoski-Stebila '13]

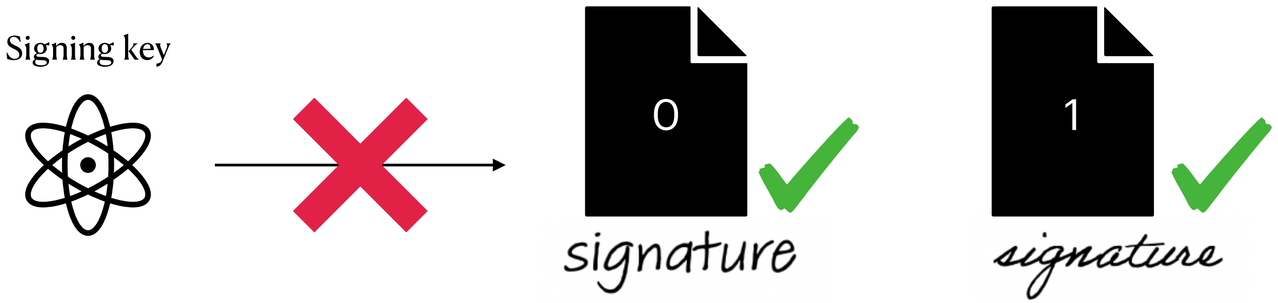
for randomized programs

Quantum One-Time Programs?

Maybe! 😀

Signature tokens [Ben-David, Sattath '16]

≈ quantum OTP
for a **specific** randomized program



- I will define this soon!
- Signature token does **not** satisfy this strong security definition

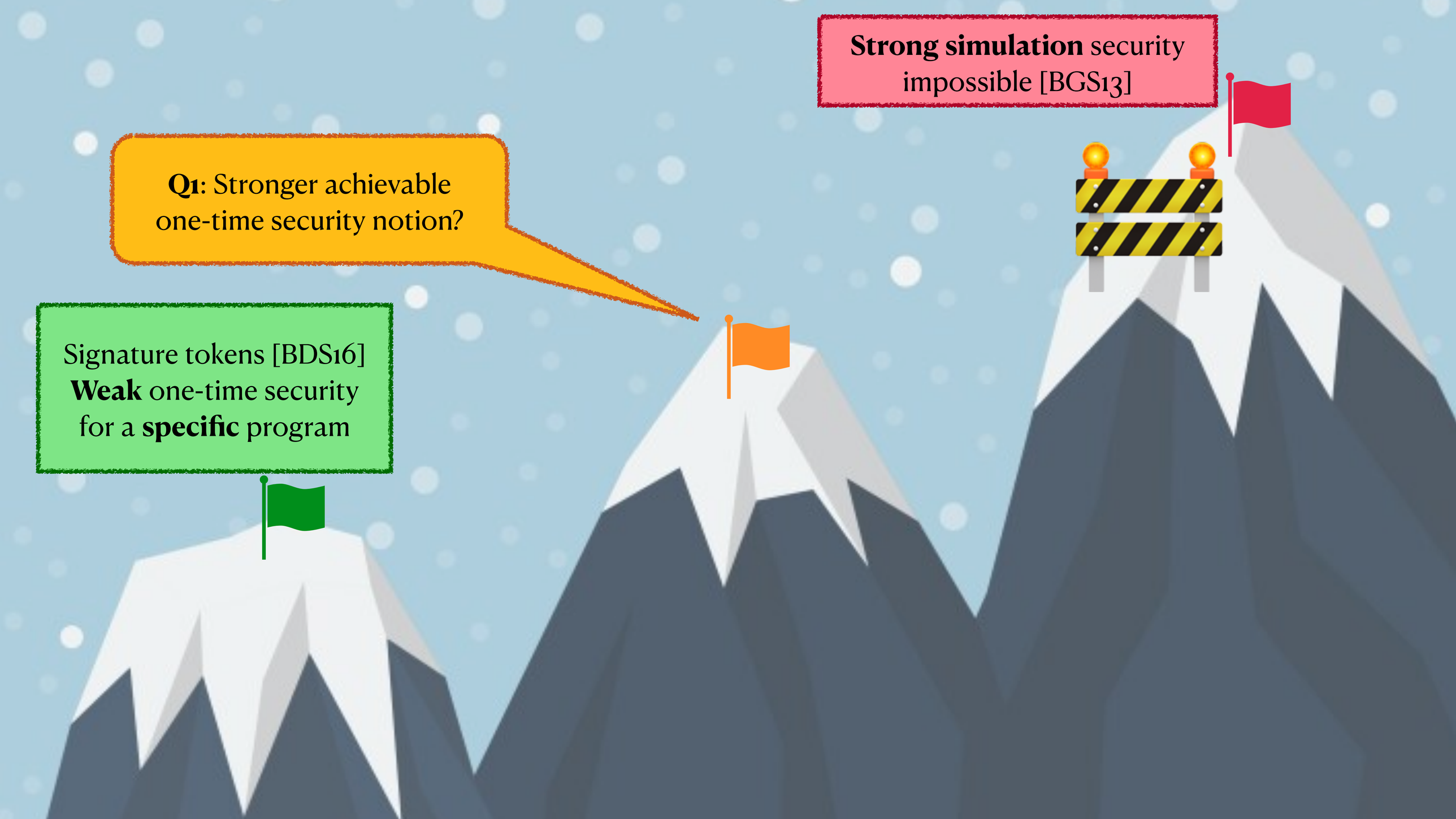
But we cannot be too greedy! 😢

No quantum OTPs that satisfy a *strong simulation security* notion, for any (randomized) programs [Broadbent-Gutoski-Stebila '13]

Strong simulation security impossible [BGS13]

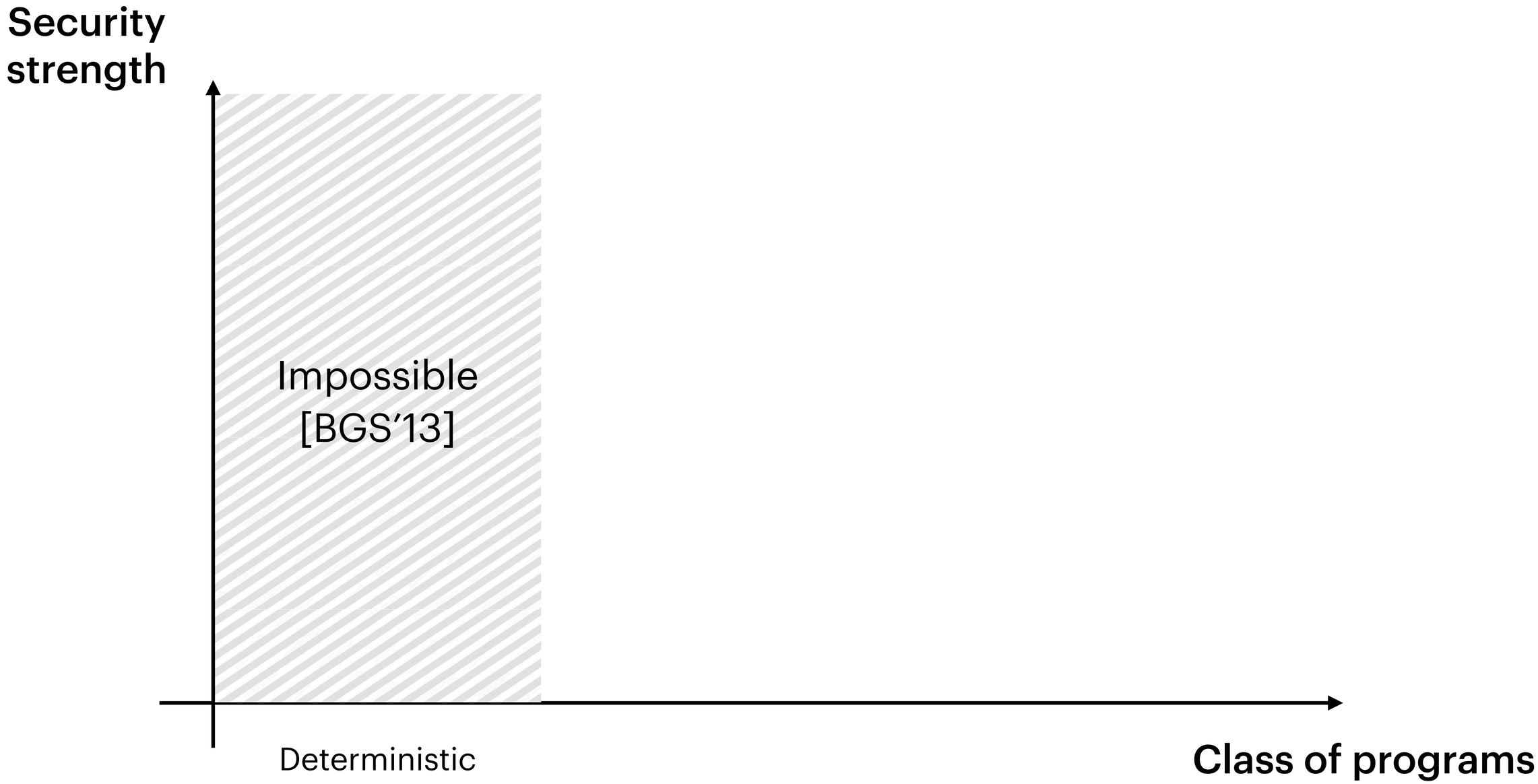
Strong simulation security
impossible [BGS13]

Signature tokens [BDS16]
Weak one-time security
for a **specific** program

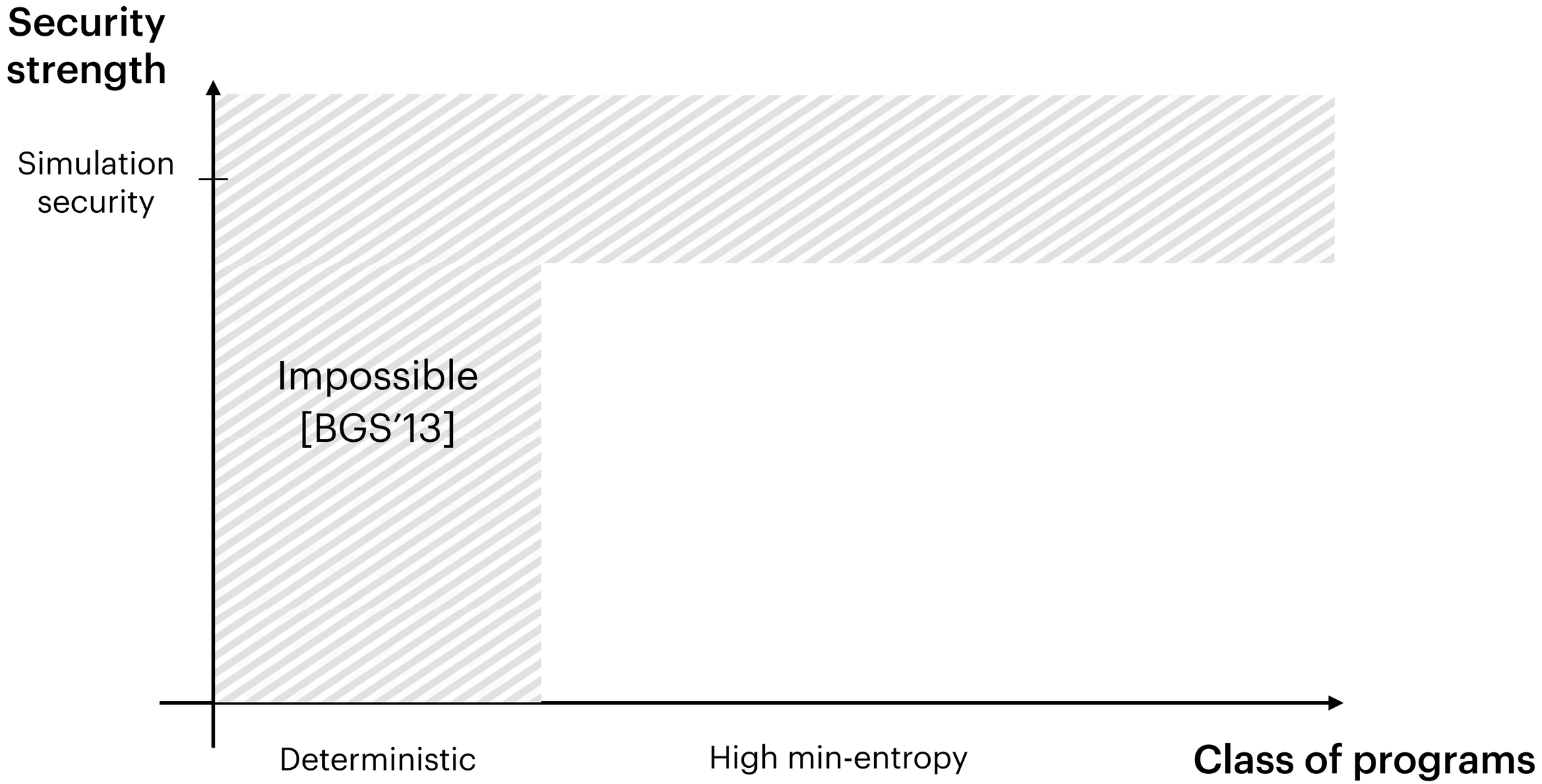

Strong simulation security impossible [BGS13]
Q1: Stronger achievable one-time security notion?
Signature tokens [BDS16] **Weak** one-time security for a **specific** program

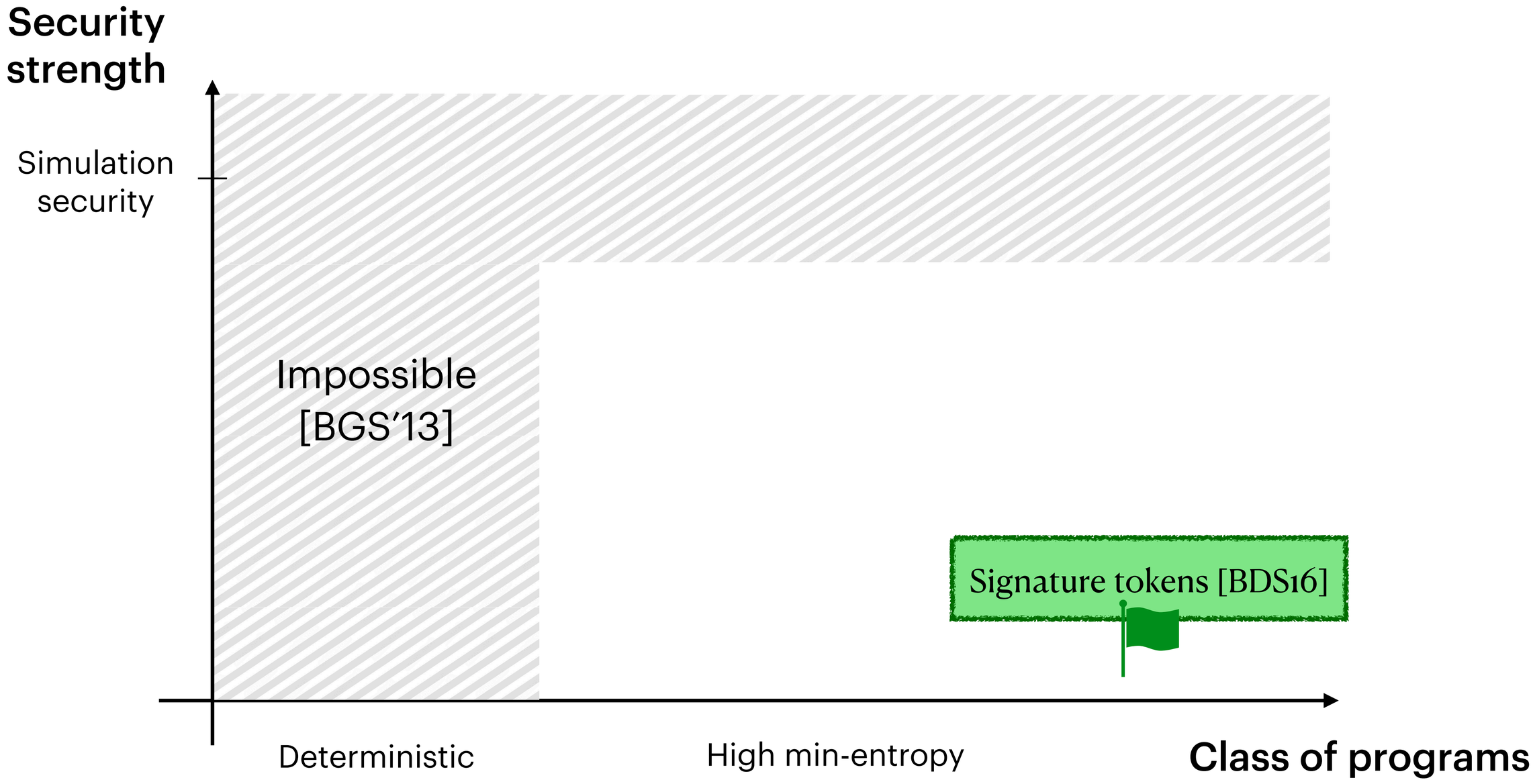

Strong simulation security impossible [BGS13]
Q1: Stronger achievable one-time security notion?
Q2: What programs can be one-time protected? All high-min entropy programs?
Signature tokens [BDS16] **Weak** one-time security for a **specific** program

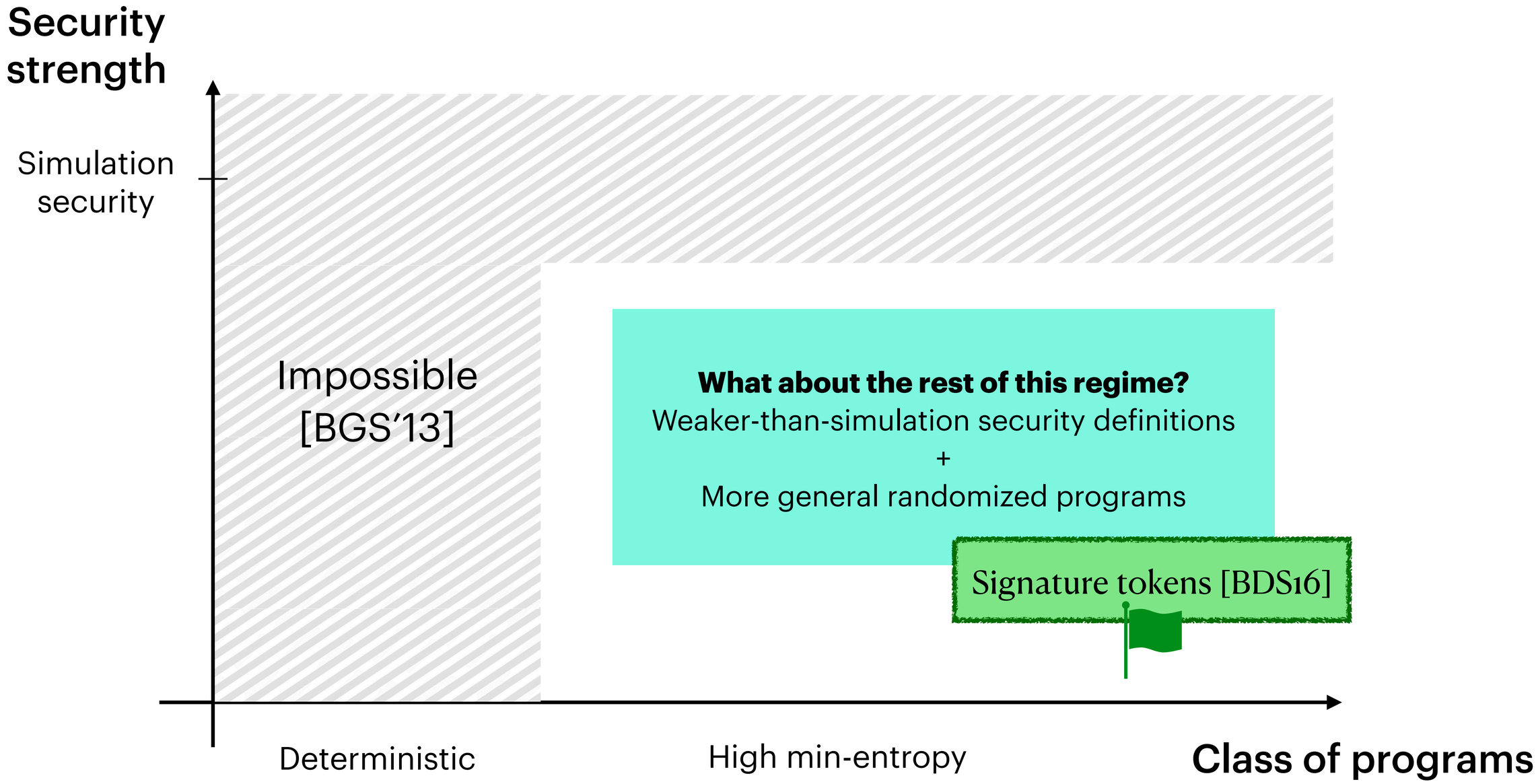

Security
strength
Class of programs


Security strength
Impossible [BGS'13]
Deterministic
Class of programs


Security strength
Simulation security
Impossible [BGS'13]
Deterministic
Class of programs


Security strength
Simulation security
Impossible [BGS'13]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
Impossible [BGS'13]
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
Impossible [BGS'13]
What about the rest of this regime?
Weaker-than-simulation security definitions
+
More general randomized programs
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs

Our Work

Our Work

Definition
Achievable intermediate
one-time security notion

Our Work

🌟 **Definition**
Achievable intermediate
one-time security notion

Our Work

🌟 **Definition**
Achievable intermediate
one-time security notion

Construction

Our Work

🌟Definition

Achievable intermediate one-time security notion

Construction

- OTP in classical oracle model secure for “unlearnable” programs

Our Work

🌟 Definition

Achievable intermediate one-time security notion

Construction

- OTP in classical oracle model secure for “unlearnable” programs
- Inspired by [BDS’16] signature tokens

Our Work

🌟 **Definition**

Achievable intermediate one-time security notion

Construction

- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Plain model

Our Work

🌟 **Definition**

Achievable intermediate one-time security notion

Construction

- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Plain model

- One-time programs for programs related to constrained PRFs

Our Work

🌟Definition
Achievable intermediate one-time security notion

Construction
- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Plain model
- One-time programs for programs related to constrained PRFs
- Impossibility for some other programs

Our Work

🌟Definition
Achievable intermediate one-time security notion

Construction
- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Applications

Plain model
- One-time programs for programs related to constrained PRFs
- Impossibility for some other programs

Our Work

🌟 Definition

Achievable intermediate one-time security notion

Construction

- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Applications

- More signature tokens
- One-time NIZK proofs
- Quantum money

Plain model

- One-time programs for programs related to constrained PRFs
- Impossibility for some other programs

Our Work

🌟Definition
Achievable intermediate one-time security notion

Construction
- OTP in classical oracle model secure for "unlearnable" programs
- Inspired by [BDS'16] signature tokens

(Post-quantum) IO

Applications
- More signature tokens
- One-time NIZK proofs
- Quantum money

Plain model
- One-time programs for programs related to constrained PRFs
- Impossibility for some other programs

Concurrent work [Gunn-Movassagh'24] gives similar construction and a simpler but weaker definition of one-time security.

Our Construction

Warm up: Signature Tokens [BDS16]

Background: Subspace states

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

- Hides subspace S

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

  - Hides subspace S
  - Uncloneable [AC'12]: $|S\rangle \nrightarrow |S\rangle \otimes |S\rangle$

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle \longrightarrow s \in S \text{ (Measure in computational basis)}$$

  - Hides subspace S
  - Uncloneable [AC'12]: $|S\rangle \nrightarrow |S\rangle \otimes |S\rangle$

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

$s \in S$ (Measure in computational basis)

OR

$s' \in S^\perp$ (Measure in Hadamard basis)

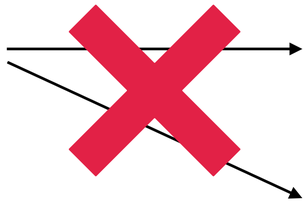
- Hides subspace S
- Uncloneable [AC'12]: $|S\rangle \nrightarrow |S\rangle \otimes |S\rangle$

Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$



$s \in S$ (Measure in computational basis)

AND

$s' \in S^\perp$ (Measure in Hadamard basis)

- Hides subspace S
- Uncloneable [AC'12]: $|S\rangle \nrightarrow |S\rangle \otimes |S\rangle$

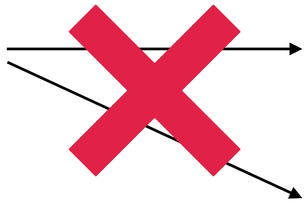
Warm up: Signature Tokens [BDS16]

Background: Subspace states

- Random subspace $S \subset \mathbb{F}_2^\lambda$ with $\dim(S) = \lambda/2$
- Subspace states [Aaronson-Christiano'12]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracles $O_S, O_{S^\perp}$



$s \in S$ (Measure in computational basis)

AND

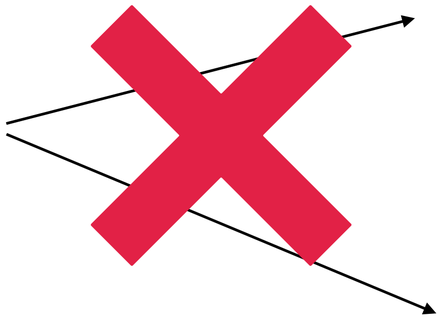
$s' \in S^\perp$ (Measure in Hadamard basis)

Direct Product Hardness [BDS'16]. Even given membership oracles $O_S, O_{S^\perp}$ hard for query-bounded adv to find both $s, s' \neq 0$ such that $s \in S$ and $s' \in S^\perp$.

Warm up: Signature Tokens [BDS16]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracles $O_S, O_{S^\perp}$



$\sigma := s \in S$ (Measure in computational basis)

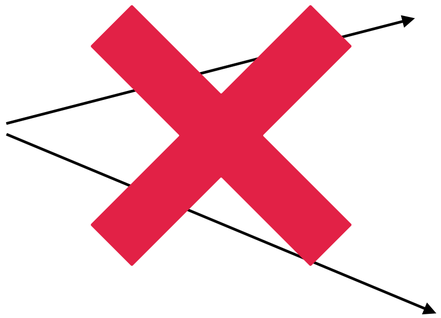
AND

$\sigma := s' \in S^\perp$ (Measure in Hadamard basis)

Warm up: Signature Tokens [BDS16]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracles $O_S, O_{S^\perp}$



<u>To sign 0:</u>

$\sigma := s \in S$ (Measure in computational basis)

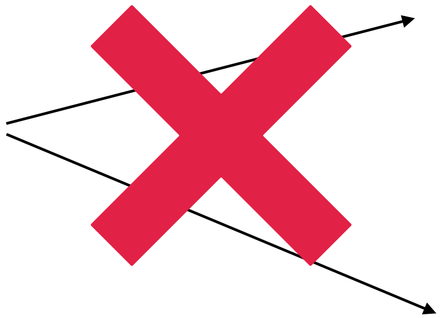
AND

$\sigma := s' \in S^\perp$ (Measure in Hadamard basis)

Warm up: Signature Tokens [BDS16]

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracles $O_S, O_{S^\perp}$



To sign 0:

$\sigma := s \in S$ (Measure in computational basis)

AND

To sign 1:

$\sigma := s' \in S^\perp$ (Measure in Hadamard basis)

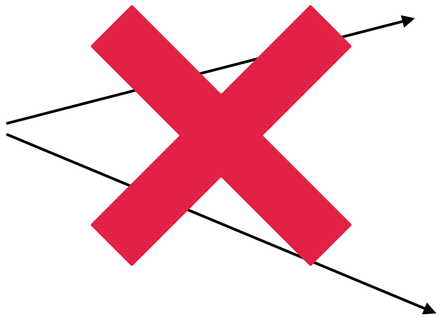
Warm up: Signature Tokens [BDS16]

Signing key

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Verification key

Oracles $O_S, O_{S^\perp}$



To sign 0:

$\sigma := s \in S$ (Measure in computational basis)

AND

To sign 1:

$\sigma := s' \in S^\perp$ (Measure in Hadamard basis)

Warm up: Signature Tokens [BDS16]

Signing key

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Verification key

Oracles $O_S, O_{S^\perp}$

To sign 0:

$\sigma := s \in S$ (Measure in computational basis)

AND

To sign 1:

$\sigma := s' \in S^\perp$ (Measure in Hadamard basis)

To verify (m, σ):

1. Measure σ
2. If $m = 0$, check $\sigma \in S$
3. If $m = 1$, check $\sigma \in S^\perp$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Sample from $f(0,\cdot)$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Sample from $f(0,\cdot)$

OR

Sample from $f(1,\cdot)$

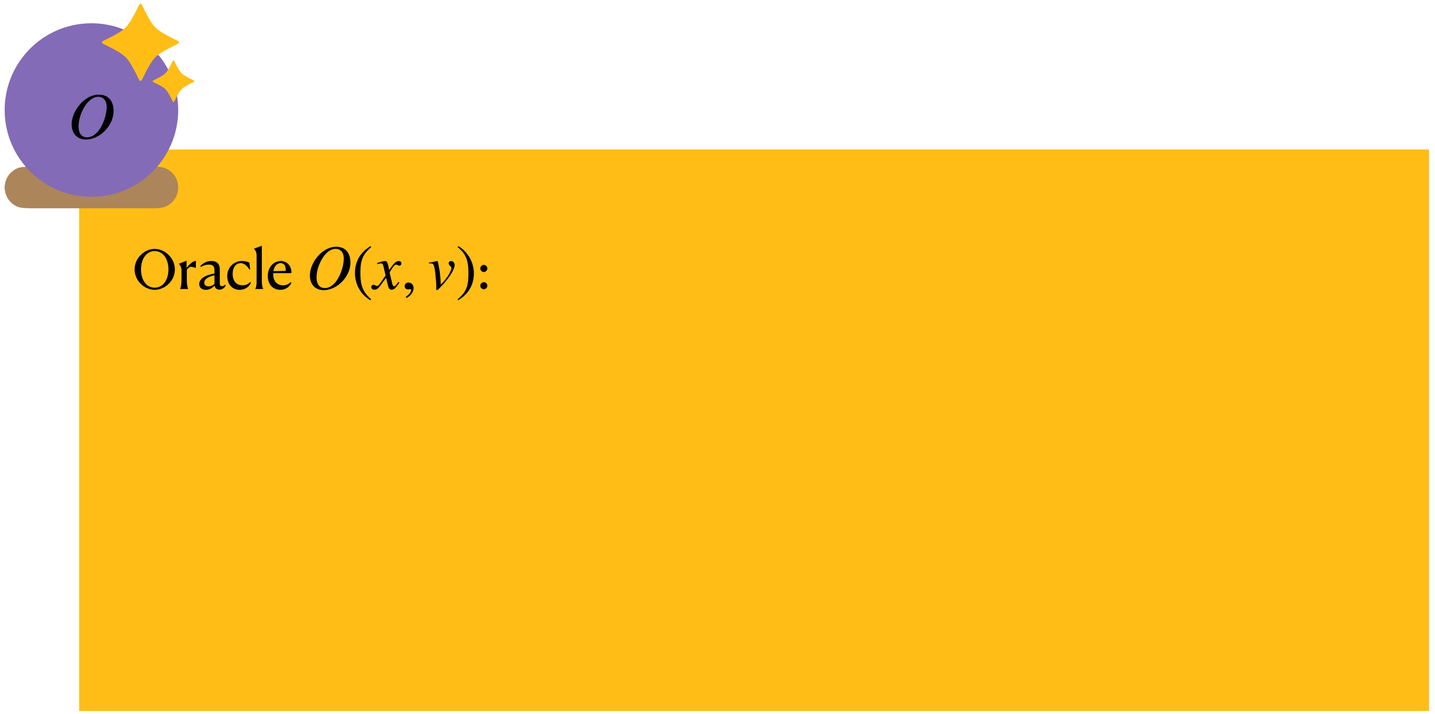
not both

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$



Sample from $f(0,\cdot)$

OR

Sample from $f(1,\cdot)$

not both

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f: \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0,r)$

Sample from $f(0,\cdot)$

OR

Sample from $f(1,\cdot)$

not both

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f: \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0,r)$

If $x = 1$ AND $v \in S^{\perp}$, output $f(1,r)$

Sample from $f(0,\cdot)$

OR

Sample from $f(1,\cdot)$

not both

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f: \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0,r)$

If $x = 1$ AND $v \in S^{\perp}$, output $f(1,r)$

Else, abort

Sample from $f(0,\cdot)$

OR

Sample from $f(1,\cdot)$

not both

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f: \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0,r)$

If $x = 1$ AND $v \in S^{\perp}$, output $f(1,r)$

Else, abort

Broken! Measuring output does not destroy $|S\rangle$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f: \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$\sum_{s \in S} |s\rangle$$

1. submit $x = 0$, $|S\rangle$ in superposition
2. measure $f(0,r)$ and then
3. repeat with $|S^{\perp}\rangle$ to get $f(1,r)$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0,r)$

If $x = 1$ AND $v \in S^{\perp}$, output $f(1,r)$

Else, abort

Broken! Measuring output does not destroy $|S\rangle$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$$|S\rangle := \sum_{s \in S} |s\rangle$$

Entangle $|S\rangle$ with randomness r

Measuring output destroys $|S\rangle$

(if f has high min-entropy)

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output ~~$f(0,r)$~~ $f(0, H(v))$

If $x = 1$ AND $v \in S^{\perp}$, output ~~$f(1,r)$~~ $f(1, H(v))$

Else, abort

Random oracle $H : \{0,1\}^{\lambda} \to \mathscr{R}$

Our OTP Construction

Goal: OTP for $f : \{0,1\}^n \times R \to Y$. For simplicity, $n = 1$.

$\mathsf{OTP}_f := (|S\rangle, O)$

Oracle $O(x, v)$:

If $x = 0$ AND $v \in S$, output $f(0, H(v))$

If $x = 1$ AND $v \in S^{\perp}$, output $f(1, H(v))$

Else, abort

Measuring output destroys $|S\rangle$ (if f has high min-entropy)

Random oracle $H : \{0,1\}^{\lambda} \to R$

Our OTP Construction

(In the classical oracle model)

Goal: OTP for $f : \{0,1\}^n \times R \to Y$ into a one-time program. For $n > 1$:

$\mathsf{OTP}_f := \left(|S_1\rangle, \ldots, |S_n\rangle, \right.$  $\left.\right)$

Oracle $O(x, v_1, \ldots, v_n)$:

If $v_i \in S_i^{x_i}$ for all $i \in [n]$: output $f(x, H(v_1, \ldots, v_n))$

Else, abort

$S_i^0 := S_i$

$S_i^1 := S_i^{\perp}$

Random oracle $H : \{0,1\}^{\lambda} \to R$

$\mathsf{OTP}_f := (|S\rangle,$  $)$

$\mathsf{OTP}_f := (|S\rangle,$  $)$

Seems like if f is "**random enough**", this construction is a **good OTP**!

$\mathsf{OTP}_f := (|S\rangle,$  $)$

Seems like if f is "**random enough**", this construction is a **good OTP**!

1. What is the strongest possible one-time security this construction satisfies?

$\mathsf{OTP}_f := (|S\rangle,$  $)$

Seems like if f is "**random enough**", this construction is a **good OTP**!

1. What is the strongest possible one-time security this construction satisfies?
2. What are the "random enough" f? All high min-entropy f?

Security Definitions

Operational Definition

Adversary cannot produce two valid outputs

$|\mathrm{OTP}_f\rangle$

f

Operational Definition

Adversary cannot produce two valid outputs

$|\mathrm{OTP}_f\rangle$

$(x_1, y_1),$
(x_2, y_2)

f

Operational Definition

Adversary cannot produce two valid outputs

$|\mathrm{OTP}_f\rangle$

f

$(x_1, y_1),$
(x_2, y_2)

Adv wins if

$\mathrm{Verify}_f(x_1, y_1) = \mathrm{Accept}$ and

$\mathrm{Verify}_f(x_2, y_2) = \mathrm{Accept}$

Operational Definition

Adversary cannot produce two valid outputs

$|\mathrm{OTP}_f\rangle$

f

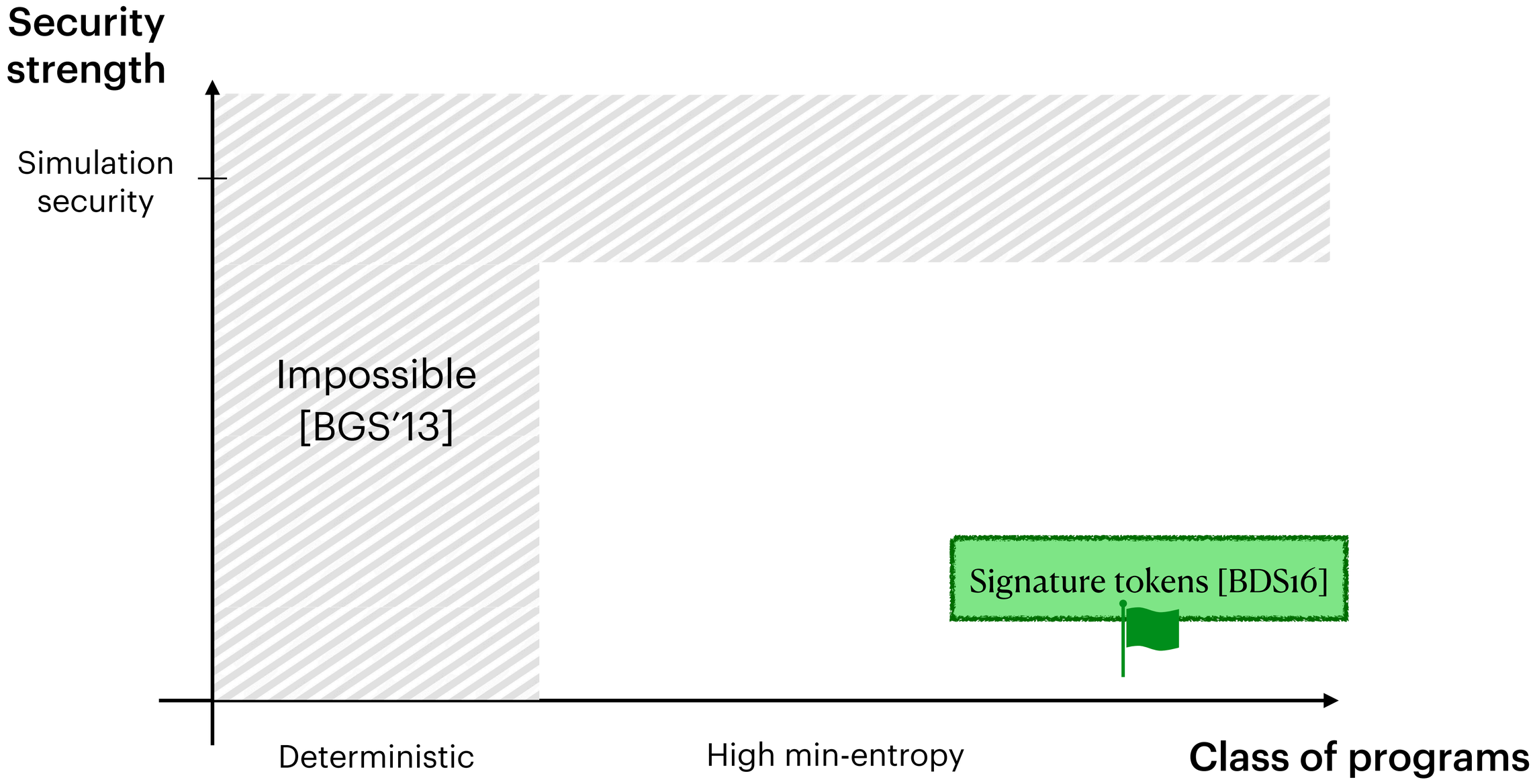
$(x_1, y_1),$
(x_2, y_2)

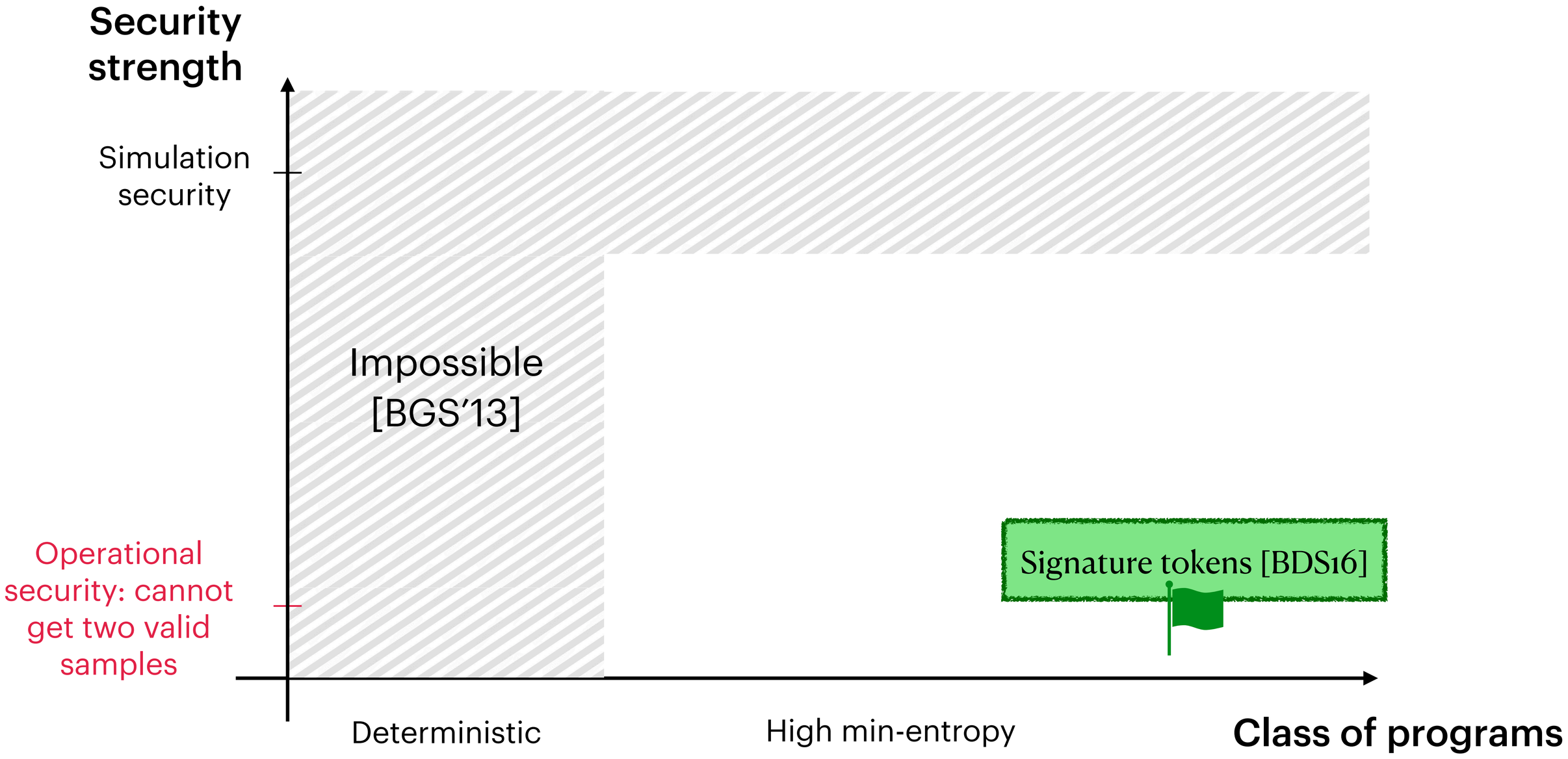
Need some way to verify that a sample is valid/correct

Adv wins if

$\mathrm{Verify}_f(x_1, y_1) = \mathrm{Accept}$ and

$\mathrm{Verify}_f(x_2, y_2) = \mathrm{Accept}$


Security strength
Simulation security
Impossible [BGS'13]
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
Impossible [BGS'13]
Operational security: cannot get two valid samples
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs

$|\mathrm{OTP}_f\rangle$

output 1, output 2

- “Two-valid-outputs” security might not be enough in some settings

$|\mathrm{OTP}_f\rangle$

output 1, output 2

- “Two-valid-outputs” security might not be enough in some settings
- Even if adversary cannot outputs two entire correct outputs, maybe it learns some other secret about f that we still want to protect

$|\mathrm{OTP}_f\rangle$

output 1, output 2

- "Two-valid-outputs" security might not be enough in some settings
- Even if adversary cannot outputs two entire correct outputs, maybe it learns some other secret about f that we still want to protect
    - Eg. Adversary $\not\to$ output 1, half of output 2

$|\mathrm{OTP}_f\rangle$

output 1, output 2

- "Two-valid-outputs" security might not be enough in some settings
- Even if adversary cannot outputs two entire correct outputs, maybe it learns some other secret about f that we still want to protect
  - Eg. Adversary $\not\to$ output 1, half of output 2
  - Eg. For PRFs, adversary should not be able to distinguish PRF output from true randomness on two different queries

$|\mathrm{OTP}_f\rangle$

output 1, output 2

- "Two-valid-outputs" security might not be enough in some settings
- Even if adversary cannot outputs two entire correct outputs, maybe it learns some other secret about f that we still want to protect
  - Eg. Adversary $\not\to$ output 1, half of output 2
  - Eg. For PRFs, adversary should not be able to distinguish PRF output from true randomness on two different queries

Simulation security?
"Adversary should learn nothing other than one sample of its choice."

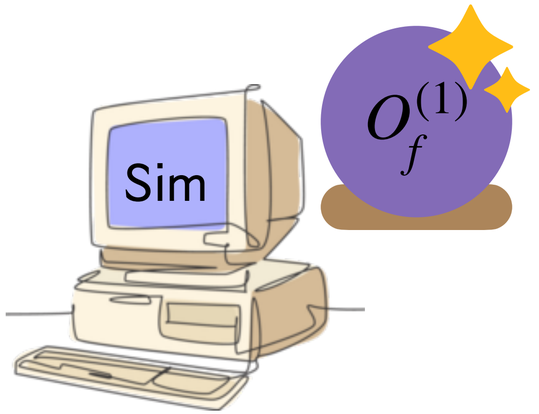
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathsf{OTP}_f\rangle$

Ideal world



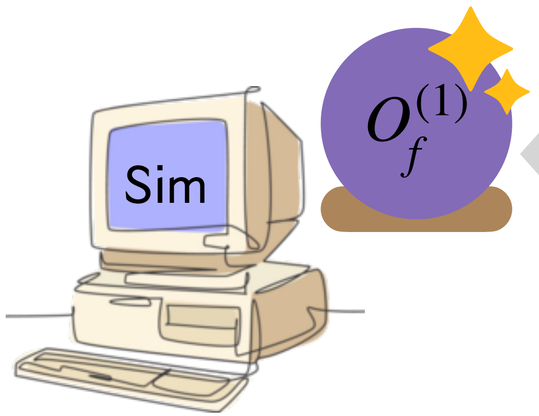
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

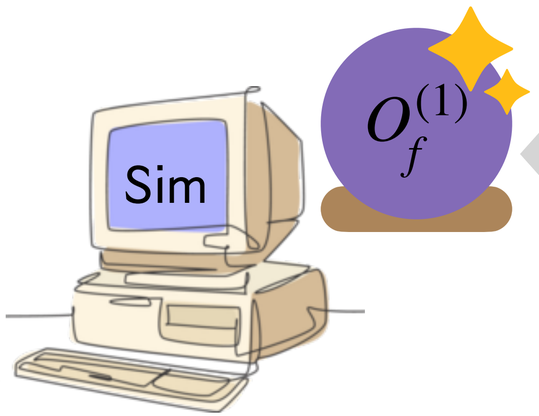
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

f

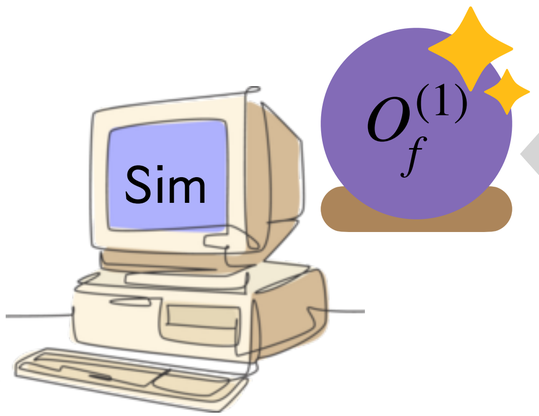
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathsf{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

f

$\forall$ Adv  , $\exists$ Sim  such that $\forall f$, $\forall$ distinguishers 

$$\mathsf{Adv}(|\mathsf{OTP}_f\rangle) \approx_c \mathsf{Sim}^{O_f^{(1)}}$$

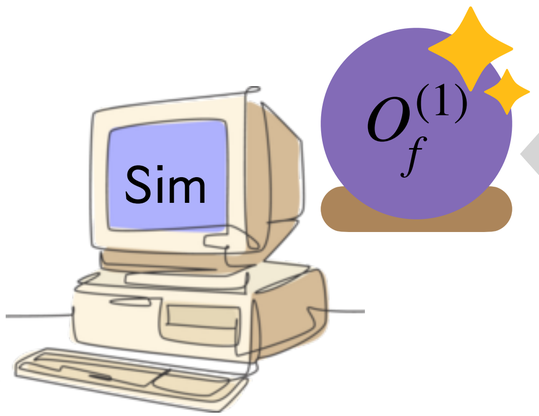
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

f

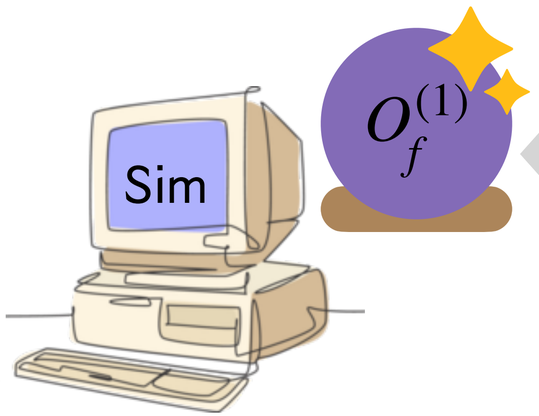
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

f

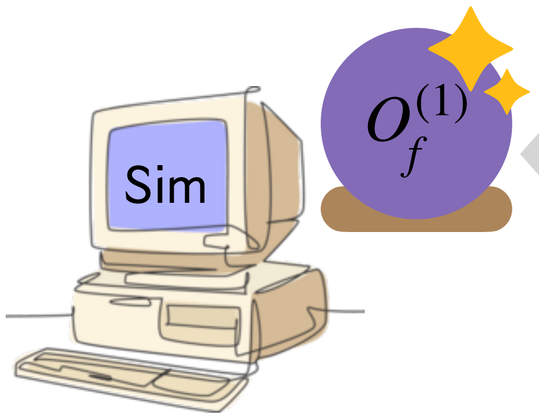
Impossible for all (randomized) programs!
[BGS'13]

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$|\mathrm{OTP}_f\rangle$

$\approx_c$

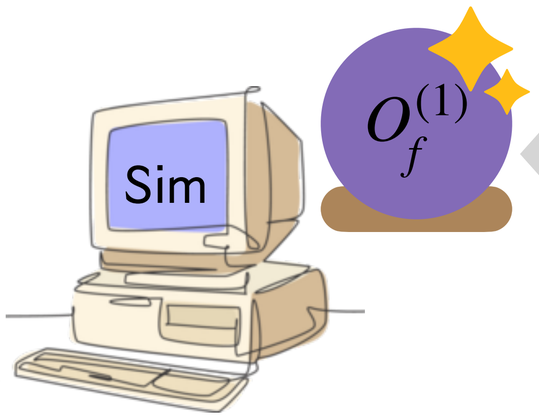
Impossible for all (randomized) programs!
[BGS'13]

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$|\mathsf{OTP}_f\rangle$

$\approx_c$

Impossible for all (randomized) programs!
[BGS'13]

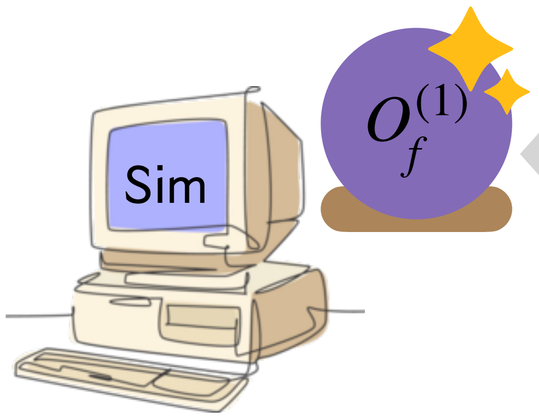
Evaluate on random x

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

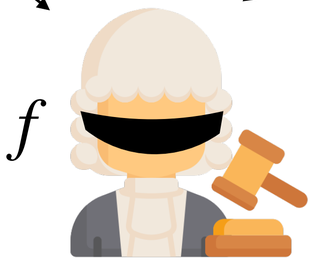
Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$|\mathrm{OTP}_f\rangle$

$\approx_c$



Impossible for all (randomized) programs!
[BGS'13]

Evaluate on random x

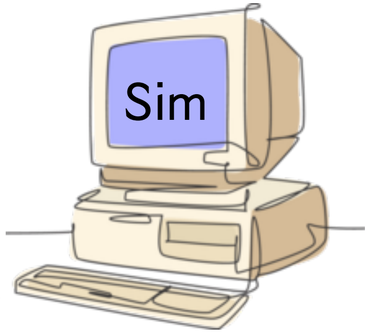
- Sim cannot win in plain model unless it learns the entire program in 1 query.

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

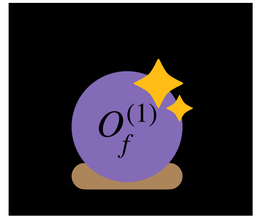
Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$|\mathsf{OTP}_f\rangle$

$\approx_c$



f

Impossible for all (randomized) programs!
[BGS'13]

Evaluate on random x

- Sim cannot win in plain model unless it learns the entire program in 1 query.
- Allow Sim to output one-time oracle-aided program?

Attempt #1: Simulation-based Definition

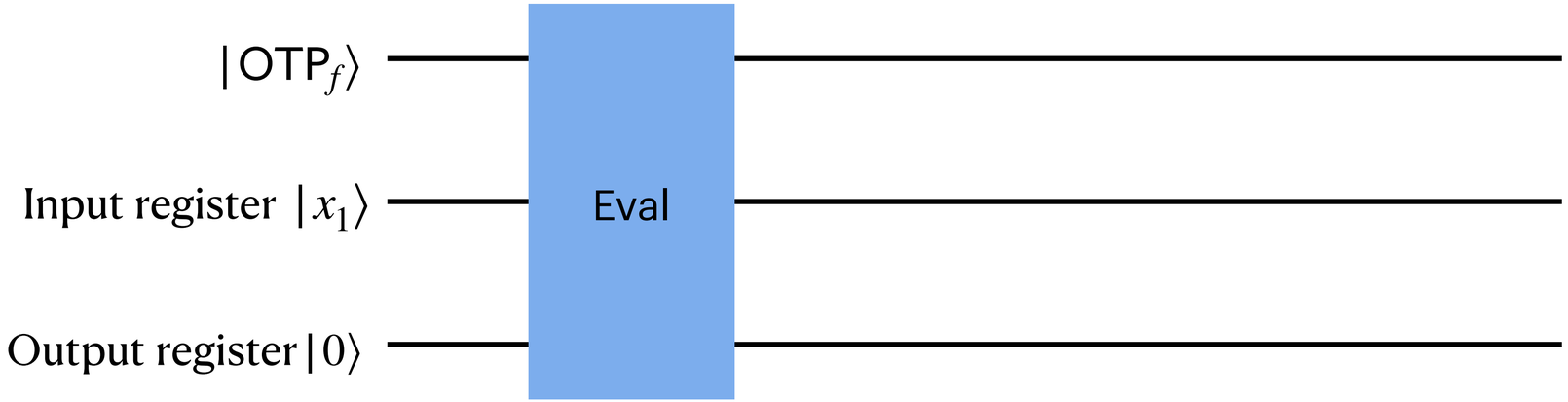
[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

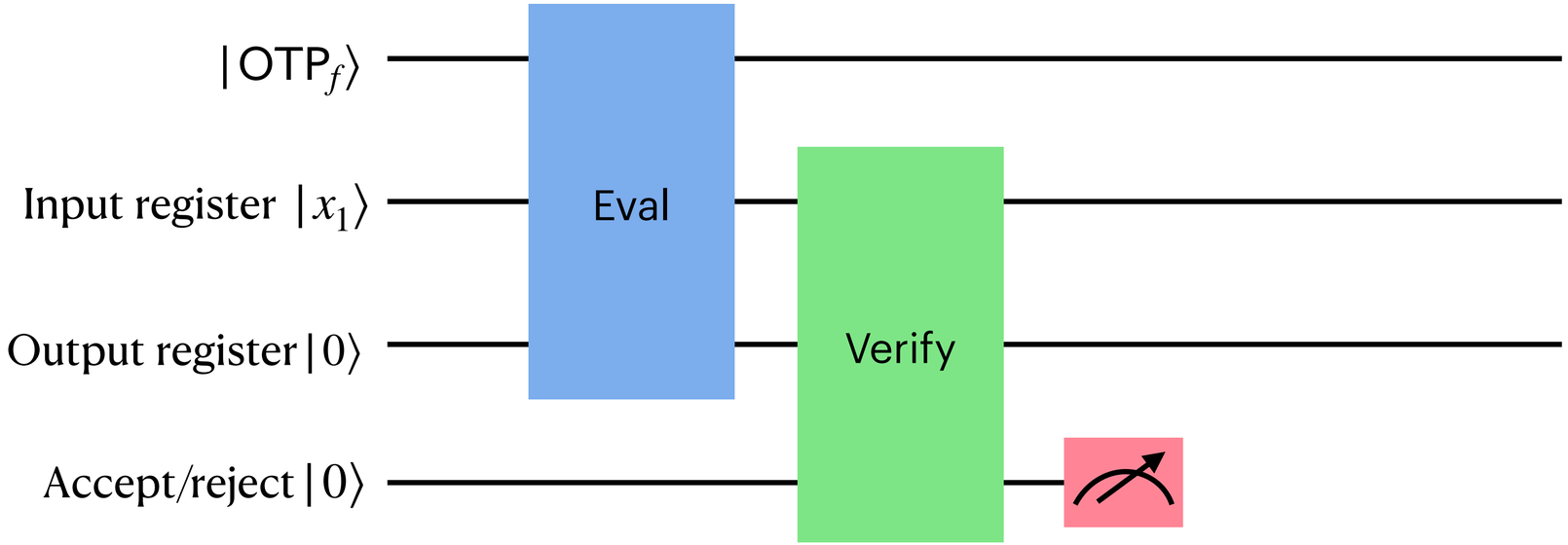
Impossible even in oracle model: The un-computation attack



Attempt #1: Simulation-based Definition

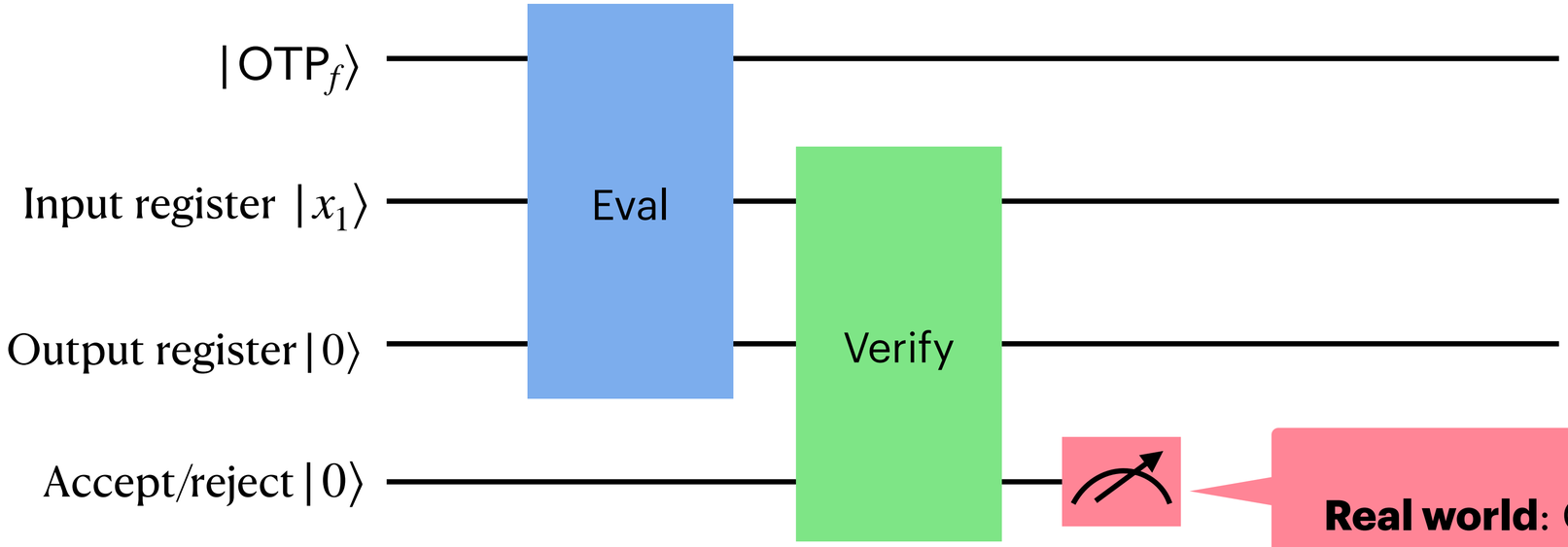
[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack



Attempt #1: Simulation-based Definition

Impossible even in oracle model: The un-computation attack [GKR'08, BGS'13]

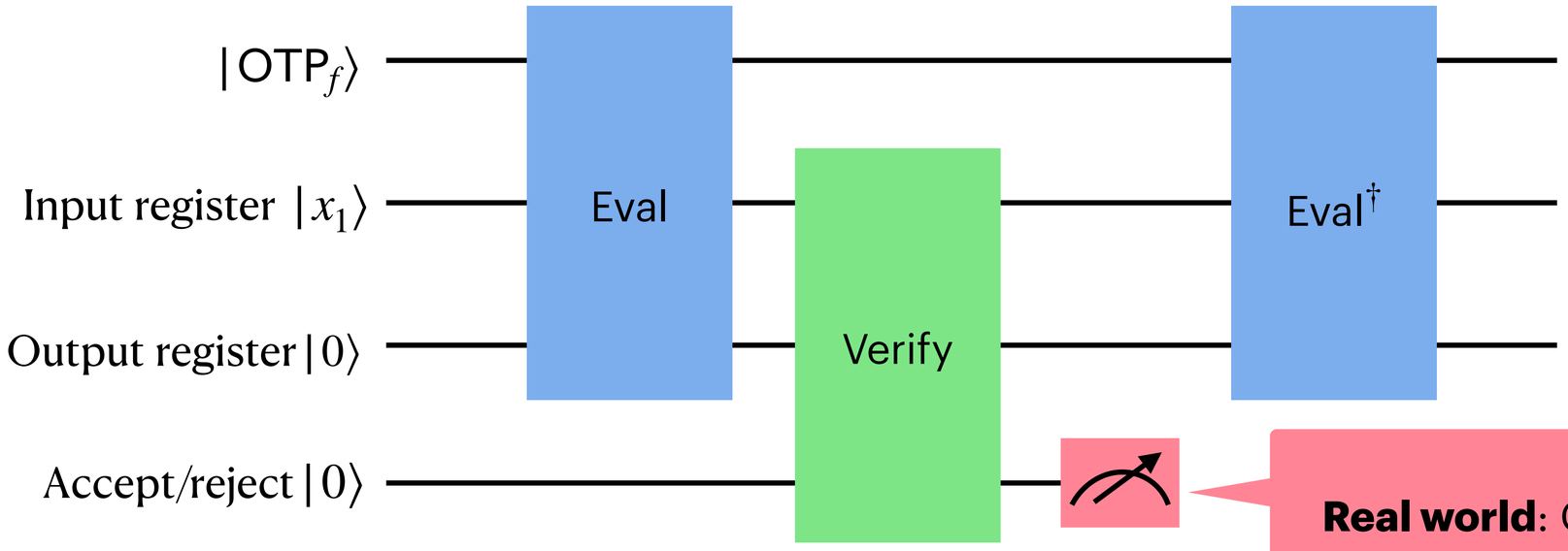


Real world: Correctness of OTP $\Longrightarrow$ gentle measurement, outcome = 1 for both x_1, x_2

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack

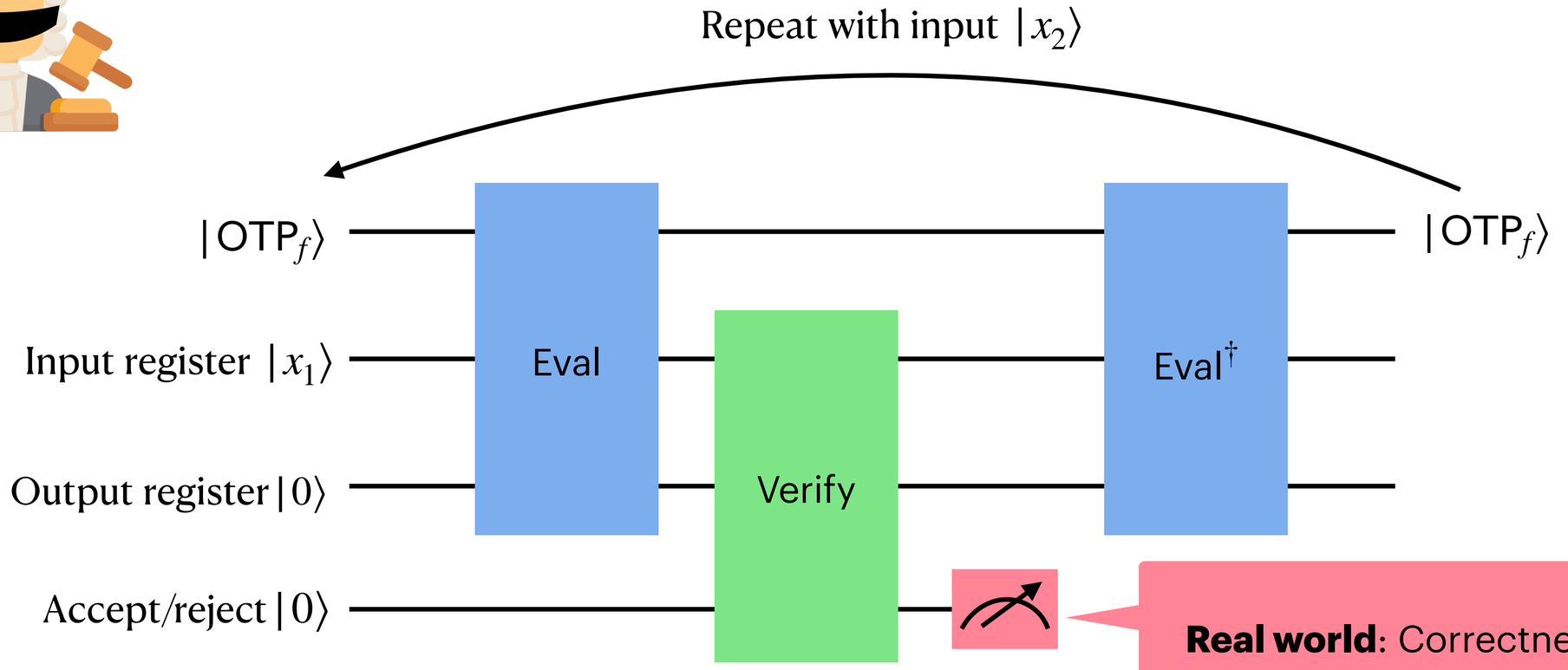


Real world: Correctness of OTP $\Longrightarrow$ gentle measurement, outcome = 1 for both x_1, x_2

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack

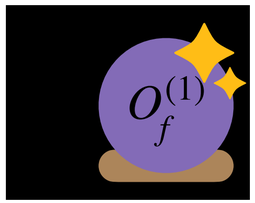


Real world: Correctness of OTP $\Longrightarrow$ gentle measurement, outcome = 1 for both x_1, x_2

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack



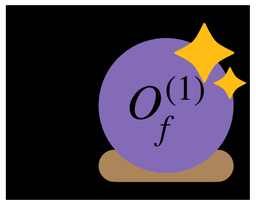
Input register $|x_1\rangle$

Output register $|0\rangle$

Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

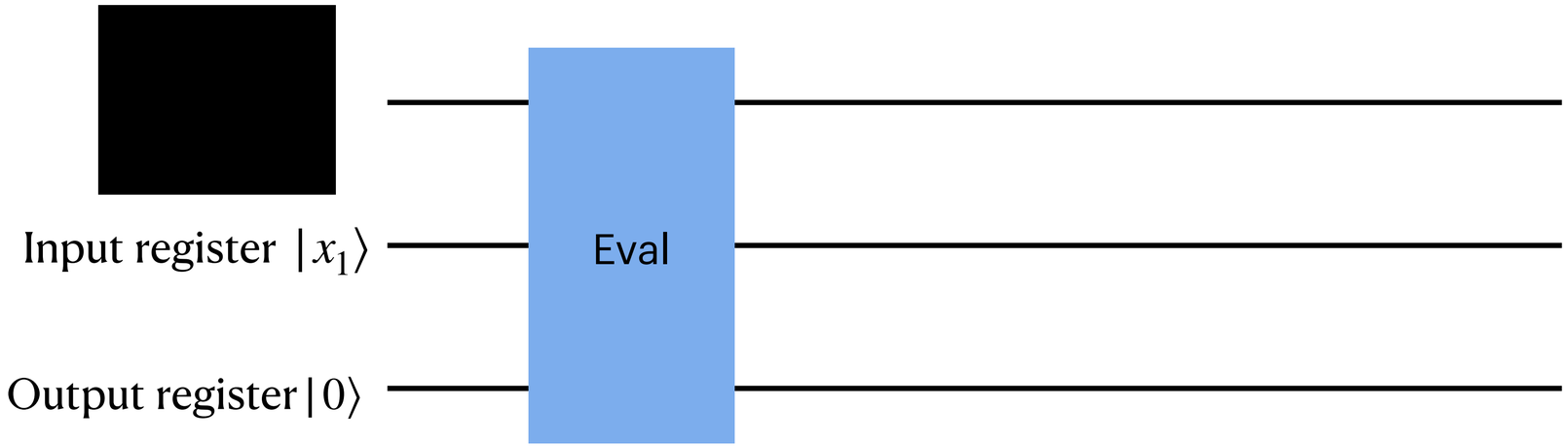
Impossible even in oracle model: The un-computation attack



Attempt #1: Simulation-based Definition

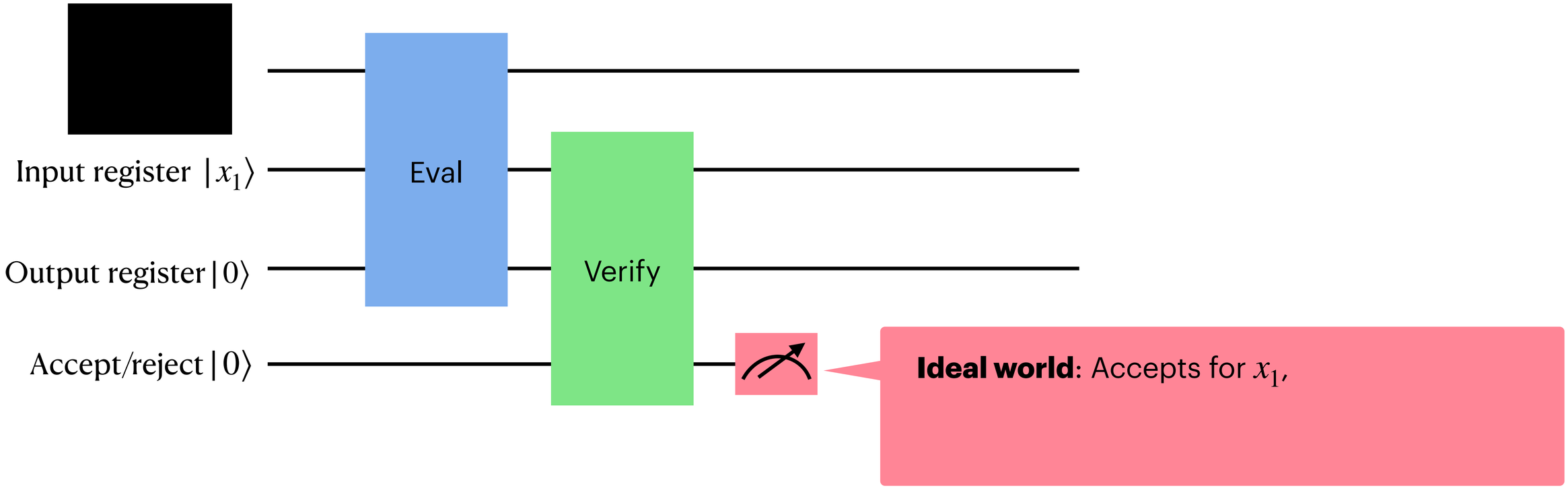
[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack



Attempt #1: Simulation-based Definition

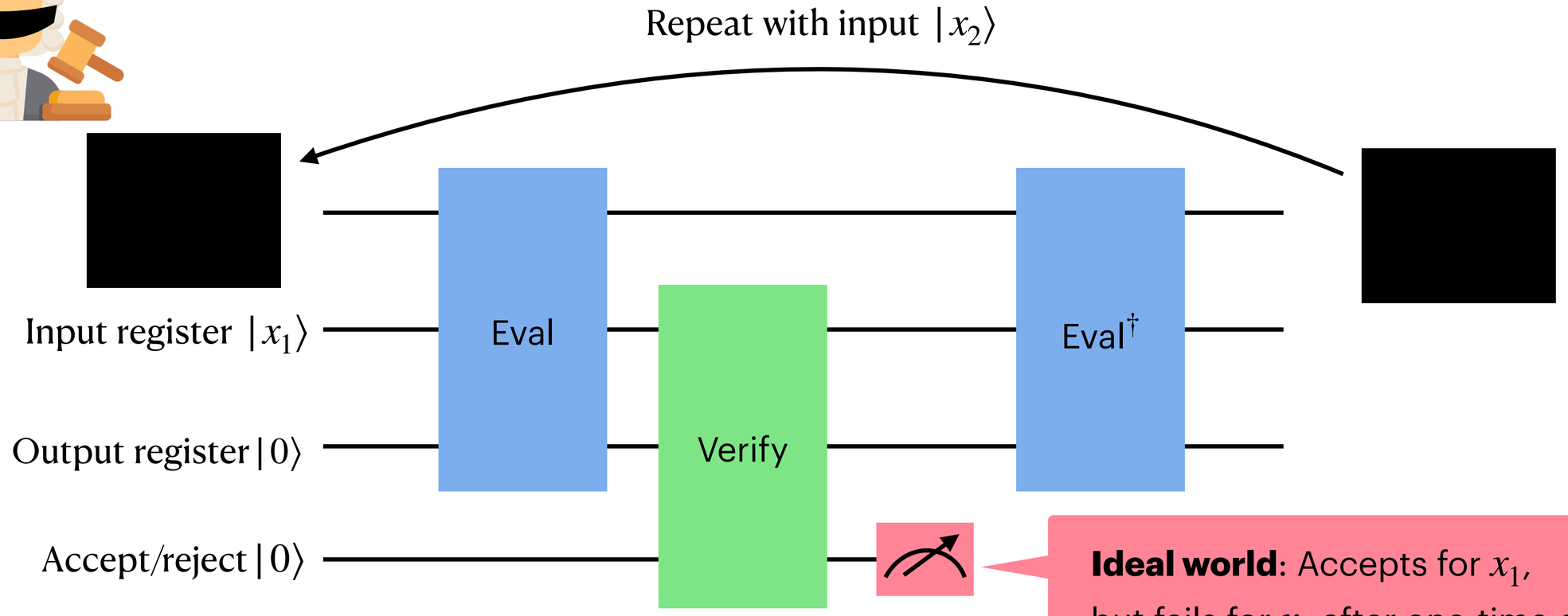
Impossible even in oracle model: The un-computation attack [GKR'08, BGS'13]



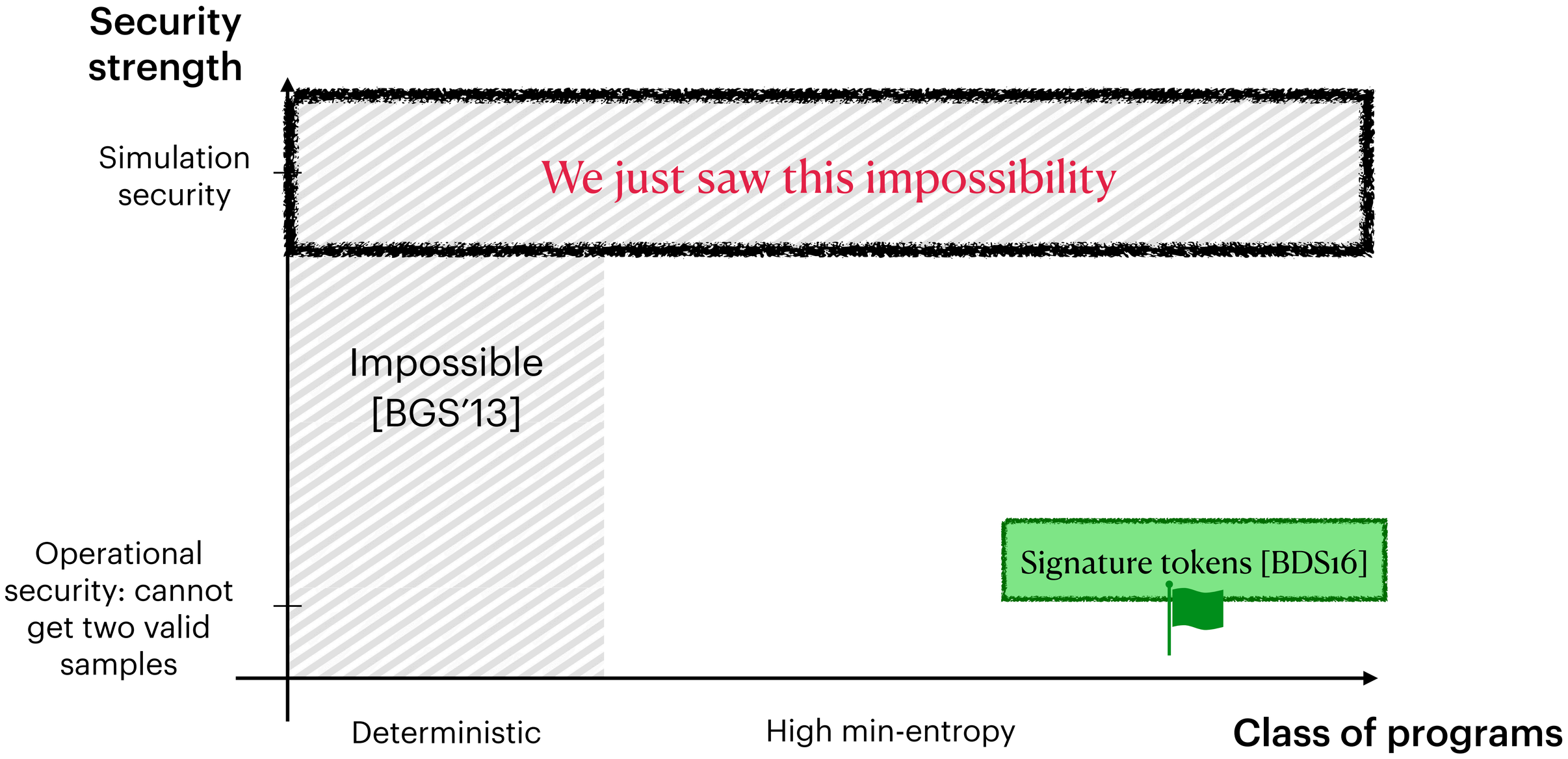
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Impossible even in oracle model: The un-computation attack



Ideal world: Accepts for x_1,
but fails for x_2 after one-time oracle self-destructs.


Security strength
Simulation security
We just saw this impossibility
Impossible [BGS'13]
Operational security: cannot get two valid samples
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs

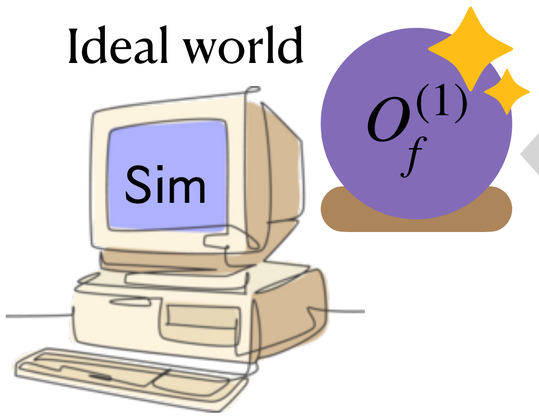
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

Impossible, even for randomized programs, even in the oracle model 😢

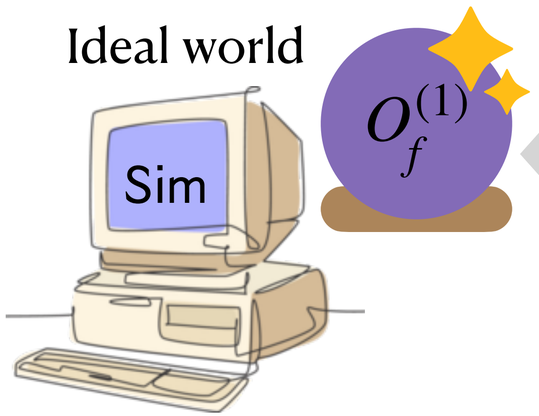
Attempt #1: Simulation-based Definition

[GKR'08, BGS'13]

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle:
Answers 1 (quantum) query, then self-destructs.

$\approx_c$

Impossible, even for randomized programs, even in the oracle model 😢

Unsatisfying impossibility: Adv didn't learn any secrets about the program.

Can we tweak definition slightly to make it achievable?

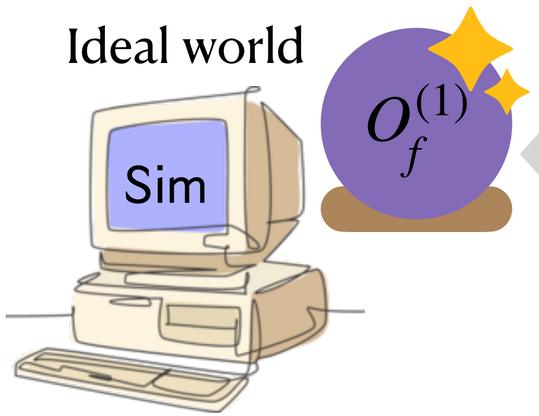
Attempt #2: Restrict adversary to classical output

Rule out "dummy" adversary

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle: Answers 1 query and turns off.

$\approx_c$

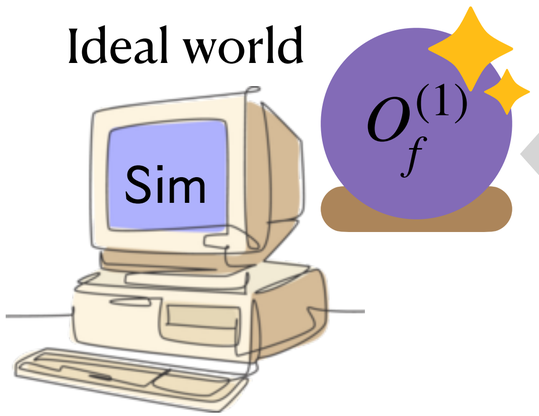
Attempt #2: Restrict adversary to classical output

Rule out "dummy" adversary

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle: Answers 1 query and turns off.

Classical output

$\approx_c$

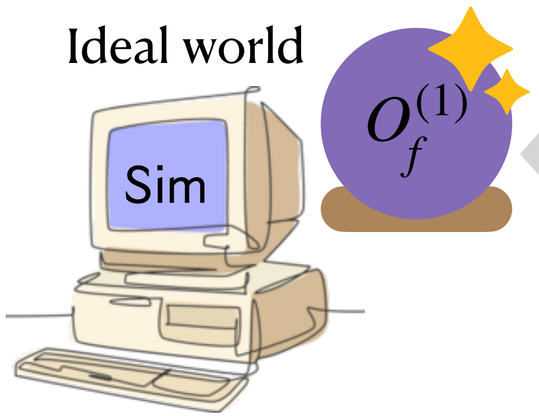
Attempt #2: Restrict adversary to classical output

Rule out "dummy" adversary

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



One-time oracle: Answers 1 query and turns off.

Classical output

$\approx_c$

Impossible, even for high min-entropy programs! [GLRRV'24]

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

Input	Output
x	$f_{a,k}(x\,;\,r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$PRF_k(x \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$PRF_k(x \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

1. Query input $|0\rangle \to$ gentle measure a, uncompute

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$PRF_k(x \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

1. Query input $|0\rangle \to$ gentle measure a, uncompute
2. Query input $|a\rangle \to$ gentle measure k

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$PRF_k(x \,\|\|\, r)$

Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

1. Query input $|0\rangle \rightarrow$ gentle measure a, uncompute
2. Query input $|a\rangle \rightarrow$ gentle measure k

Input x	Output $f_{a,k}(x\,;\,r)$
0	$a \,\|\|\, PRF_k(0 \,\|\| r)$
a	$k \,\|\|\, PRF_k(a \,\|\| r)$
$x \neq 0, a$	$PRF_k(x \,\|\| r)$

Classical output $= k$

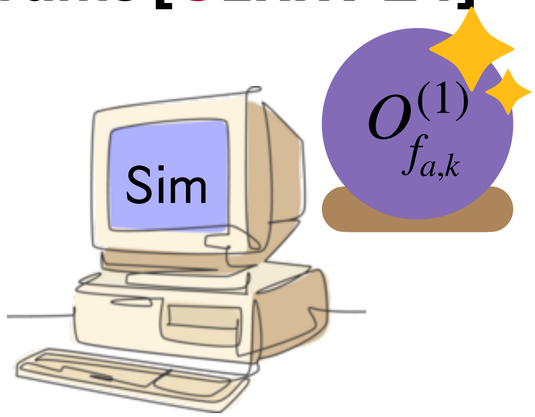
Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

1. Query input $|0\rangle \to$ gentle measure a, uncompute
2. Query input $|a\rangle \to$ gentle measure k

Input x	Output $f_{a,k}(x\,;r)$
0	$a \,\|\|\, PRF_k(0 \,\|\| r)$
a	$k \,\|\|\, PRF_k(a \,\|\| r)$
$x \neq 0, a$	$PRF_k(x \,\|\| r)$



Cannot learn k
when a, k are chosen randomly

Classical output $= k$

$\not\approx_c$

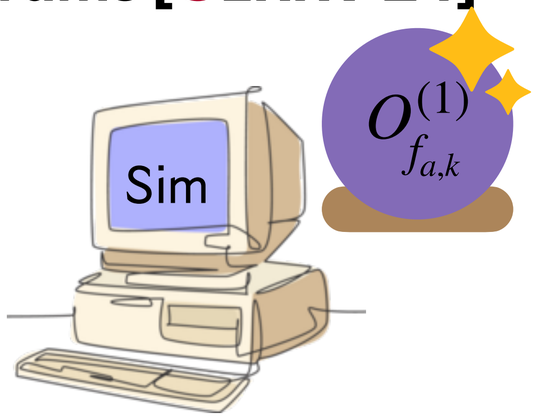
Attempt #2: Restrict adversary to classical output

Impossibility: partially deterministic high-entropy programs [GLRRV'24]

$|\mathrm{OTP}_{f_{a,k}}\rangle$

1. Query input $|0\rangle \to$ gentle measure a, uncompute
2. Query input $|a\rangle \to$ gentle measure k

Input x	Output $f_{a,k}(x\,;r)$
0	$a \,\|\|\, PRF_k(0 \,\|\| r)$
a	$k \,\|\|\, PRF_k(a \,\|\| r)$
$x \neq 0, a$	$PRF_k(x \,\|\| r)$



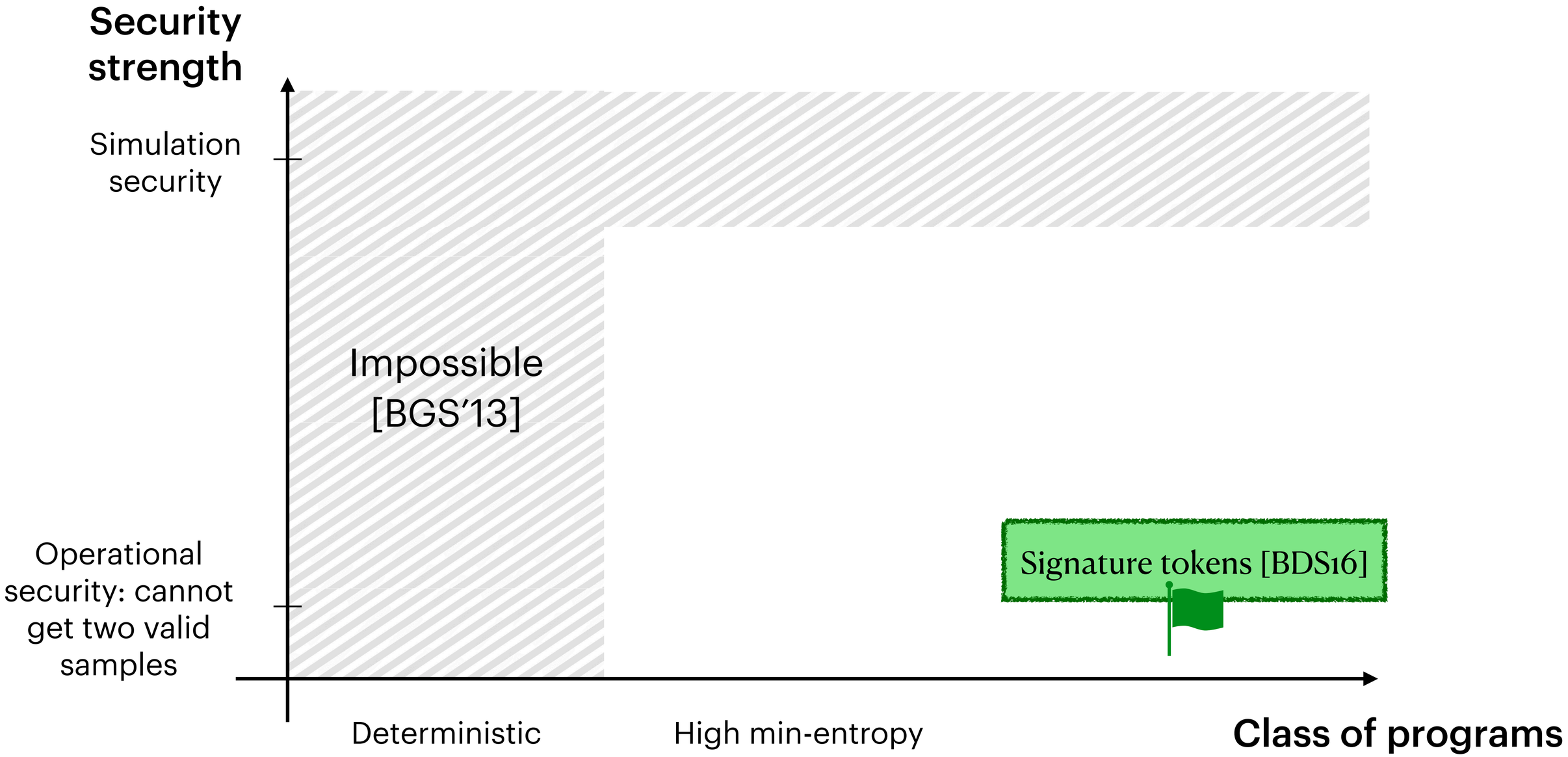
Cannot learn k
when a, k are chosen randomly

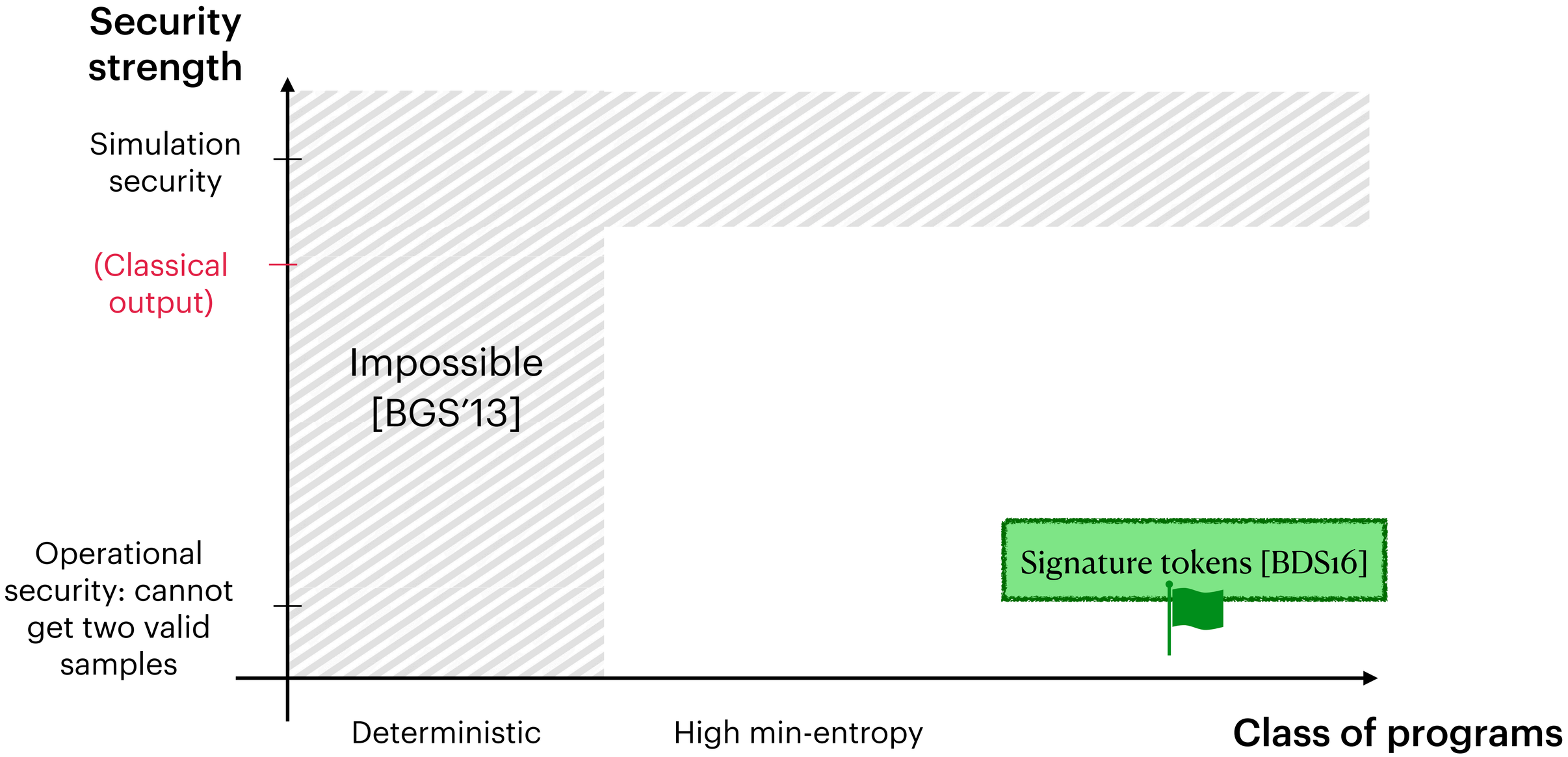
Classical output $= k$

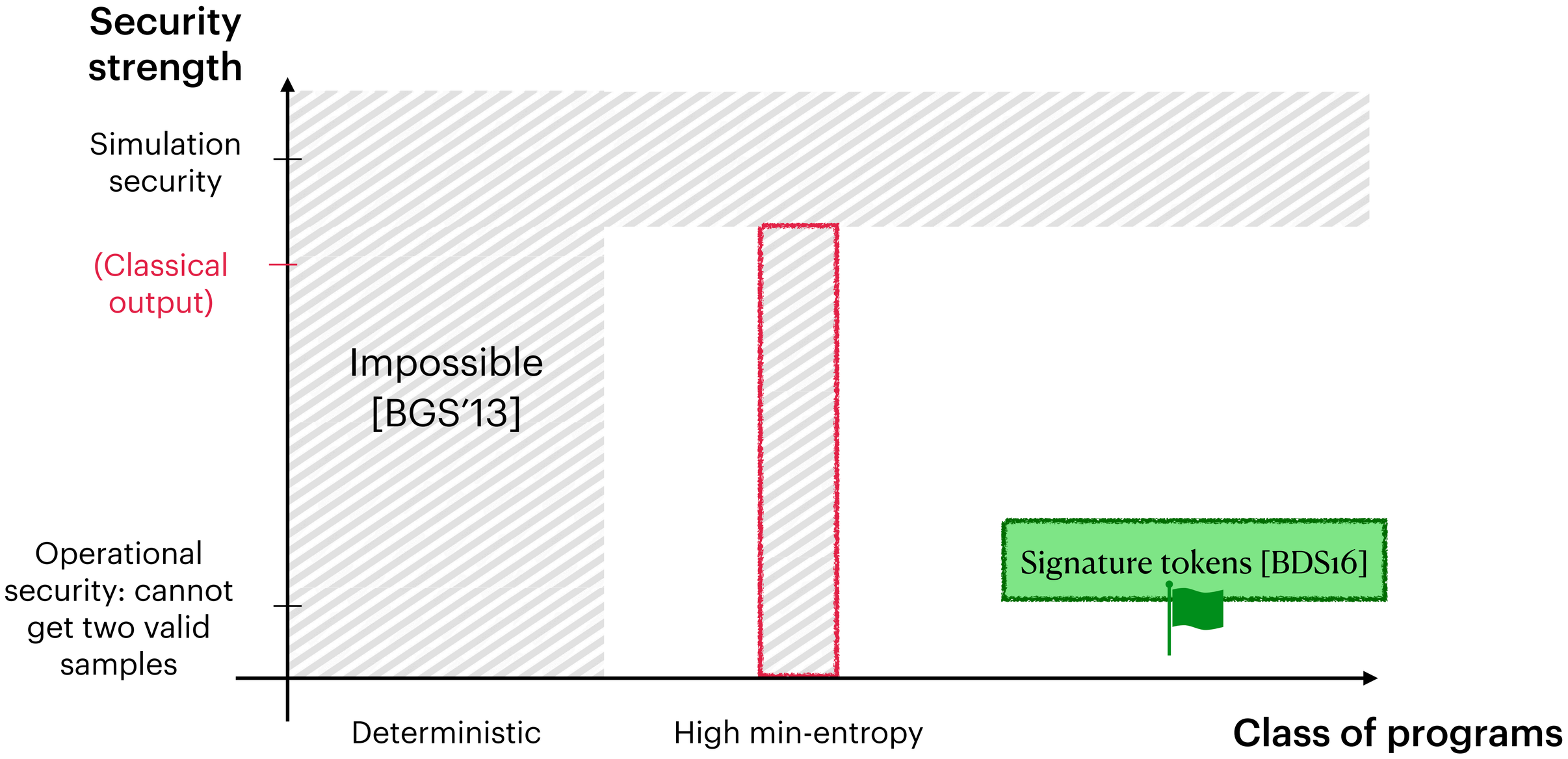
$\not\approx_c$

Lesson #1: Restricting adv to classical outputs does not help

Lesson #2: high min-entropy is the wrong notion!


Security strength
Simulation security
Operational security: cannot get two valid samples
Impossible [BGS'13]
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
(Classical output)
Impossible [BGS'13]
Operational security: cannot get two valid samples
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
(Classical output)
Operational security: cannot get two valid samples
Impossible [BGS'13]
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs

Key Idea

- Cannot force the adversary to make destructive measurements

Key Idea

- Cannot force the adversary to make destructive measurements
- Adversary can always **(1) make many gentle measurements** and **(2) uncompute** before making a destructive measurement

Key Idea

- Cannot force the adversary to make destructive measurements
- Adversary can always **(1) make many gentle measurements** and **(2) uncompute** before making a destructive measurement

- Intuition: For “random enough” programs,

  No destructive measurement $\Longrightarrow$ adversary does not learn useful information

Key Idea

- Cannot force the adversary to make destructive measurements
- Adversary can always **(1) make many gentle measurements** and **(2) uncompute** before making a destructive measurement

- Intuition: For “random enough” programs,

  No destructive measurement $\Longrightarrow$ adversary does not learn useful information

- So let’s allow Sim to also make gentle measurements & uncompute!

  (As long as it makes only 1 destructive measurement)

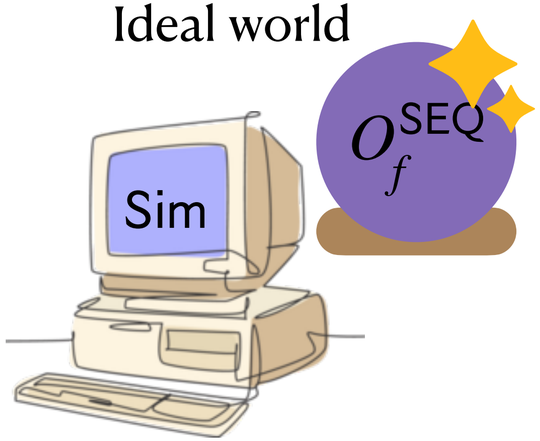
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



$\approx_c$

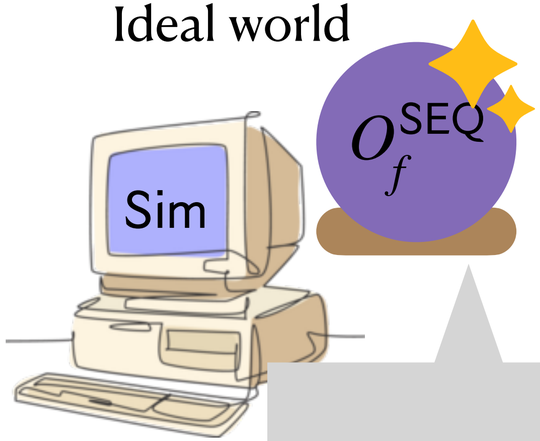
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



Single Effective query oracle:
At most 1 **destructive** query

$\approx_c$

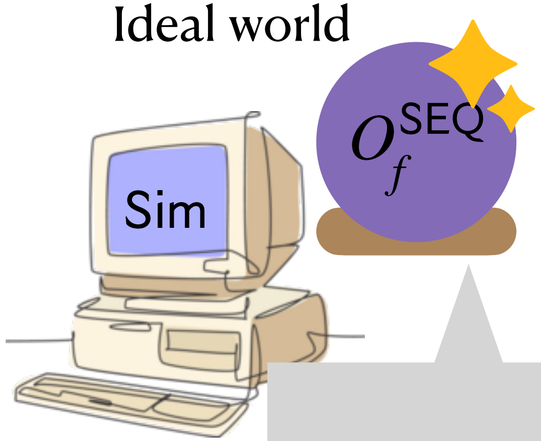
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



Single Effective query oracle:
At most 1 **destructive** query

$\approx_c$

Keep track of destructive queries using Zhandry's compressed oracle technique.

Why is the SEQ model any good?

Why is the SEQ model any good?

1. Achievable!

Theorem [GLRRV'24]. In the classical oracle model, there is a OTP compiler that achieves SEQ simulation-based security for every program f.

Why is the SEQ model any good?

1. Achievable!

Theorem [GLRRV'24]. In the classical oracle model, there is a OTP compiler that achieves SEQ simulation-based security for every program f.

But wait! No OTP for deterministic programs, even in the weakest sense of one-time security …

Why is the SEQ model any good?

1. Achievable!

Theorem [GLRRV'24]. In the classical oracle model, there is a OTP compiler that achieves SEQ simulation-based security for every program f.

But wait! No OTP for deterministic programs, even in the weakest sense of one-time security …

- SEQ security is meaningless for deterministic programs, just like obfuscation and copy-protection are meaningless for "learnable" programs.

Why is the SEQ model any good?

1. Achievable!

Theorem [GLRRV'24]. In the classical oracle model, there is a OTP compiler that achieves SEQ simulation-based security for every program f.

2. Meaningful: can recover operational one-time security definitions for "random enough" programs

For "random enough" programs,

$$\text{SEQ security} \implies \text{operational security}$$

“Random enough”

= SEQ-unlearnability

“Random enough”

= SEQ-unlearnability

“Random enough”

= SEQ-unlearnability

(x_1, y_1),
(x_2, y_2)

Adv wins if
$\mathsf{Verify}_f(x_1, y_1) = \mathsf{Accept}$ and
$\mathsf{Verify}_f(x_2, y_2) = \mathsf{Accept}$

“Random enough”

= SEQ-unlearnability

(x_1, y_1),
(x_2, y_2)

Adv wins if
$\mathsf{Verify}_f(x_1, y_1) = \mathsf{Accept}$ and
$\mathsf{Verify}_f(x_2, y_2) = \mathsf{Accept}$

Can define more general learning games.
Eg. One-time PRF security game.

SEQ (Un)learnability

SEQ (Un)learnability

[GLRRV'24] The following programs are **SEQ-unlearnable**:

- Truly random functions
- Pairwise independence
- Functions related to Blind Unforgeable signatures [AMRS20]

SEQ (Un)learnability

[GLRRV'24] The following programs are **SEQ-unlearnable**:

- Truly random functions
- Pairwise independence
- Functions related to Blind Unforgeable signatures [AMRS20]

Examples of **SEQ-learnable** programs:

- Deterministic programs
- Even some high min-entropy (but partially deterministic) programs

SEQ (Un)learnability

[GLRRV'24] The following programs are **SEQ-unlearnable**:

- Truly random functions
- Pairwise independence
- Functions related to Blind Unforgeable signatures [AMRS20]

Examples of **SEQ-learnable** programs:

- Deterministic programs
- Even some high min-entropy (but partially deterministic) programs

Input	Output
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$0 \,\|\|\, PRF_k(x \,\|\|\, r)$

SEQ (Un)learnability

[GLRRV'24] The following programs are **SEQ-unlearnable**:

- Truly random functions
- Pairwise independence
- Functions related to Blind Unforgeable signatures [AMRS20]

Examples of **SEQ-learnable** programs:

- Deterministic programs
- Even some high min-entropy (but partially deterministic) programs

Input	Output
0	$a \,\|\|\, PRF_k(0 \,\|\|\, r)$
a	$k \,\|\|\, PRF_k(a \,\|\|\, r)$
$x \neq 0, a$	$0 \,\|\|\, PRF_k(x \,\|\|\, r)$

No hope of ever producing a OTP secure for two-valid-sample game.

Why is the SEQ model any good?

Makes progress on both our guiding questions:

Why is the SEQ model any good?

Makes progress on both our guiding questions:

1. Stronger-than-operational notion of OTP that is achievable?

Why is the SEQ model any good?

Makes progress on both our guiding questions:

1. Stronger-than-operational notion of OTP that is achievable?

SEQ simulation security

Why is the SEQ model any good?

Makes progress on both our guiding questions:

1. Stronger-than-operational notion of OTP that is achievable?

SEQ simulation security

2. What class of programs can be compiled into an OTP?

Why is the SEQ model any good?

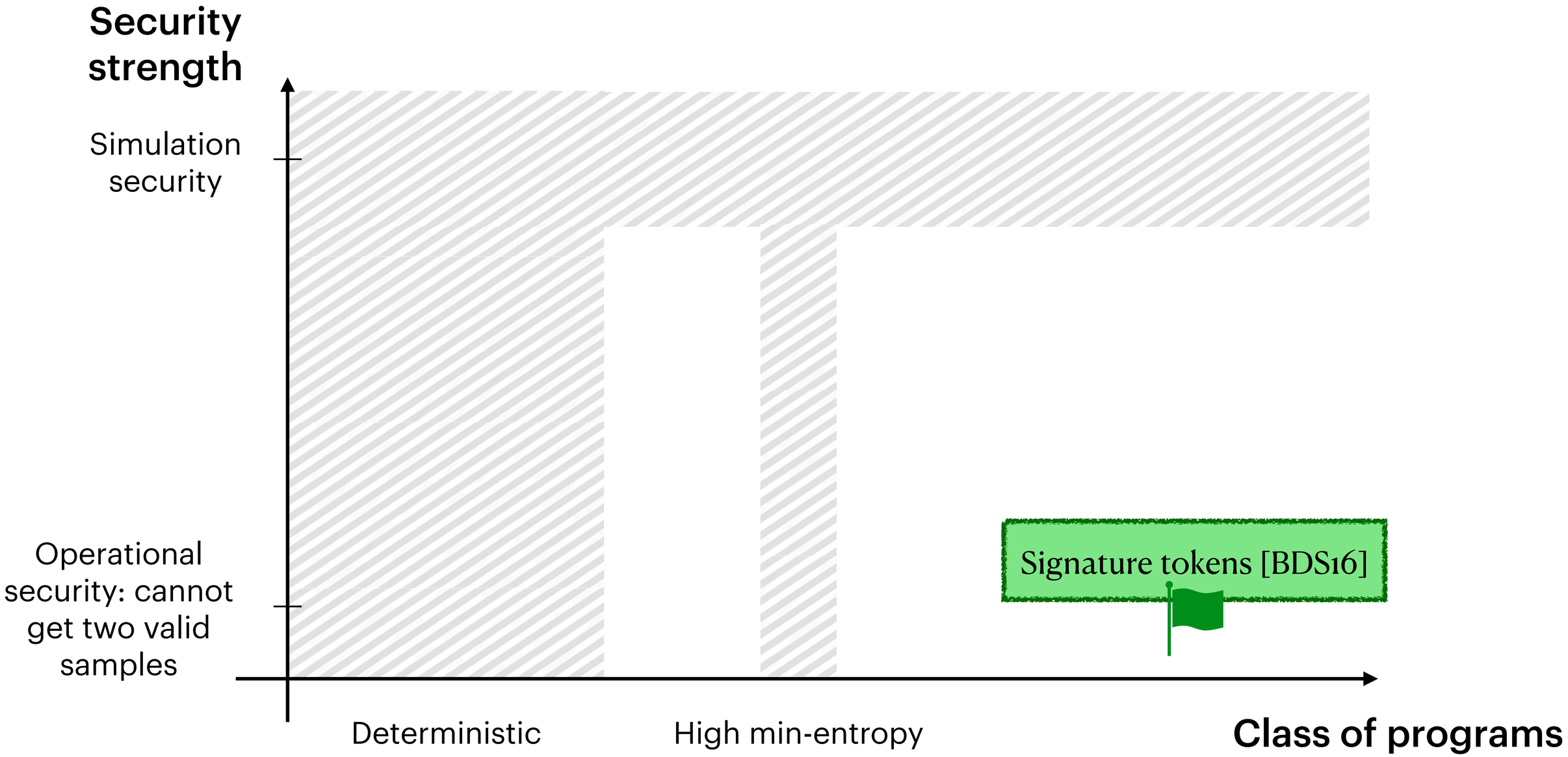
Makes progress on both our guiding questions:

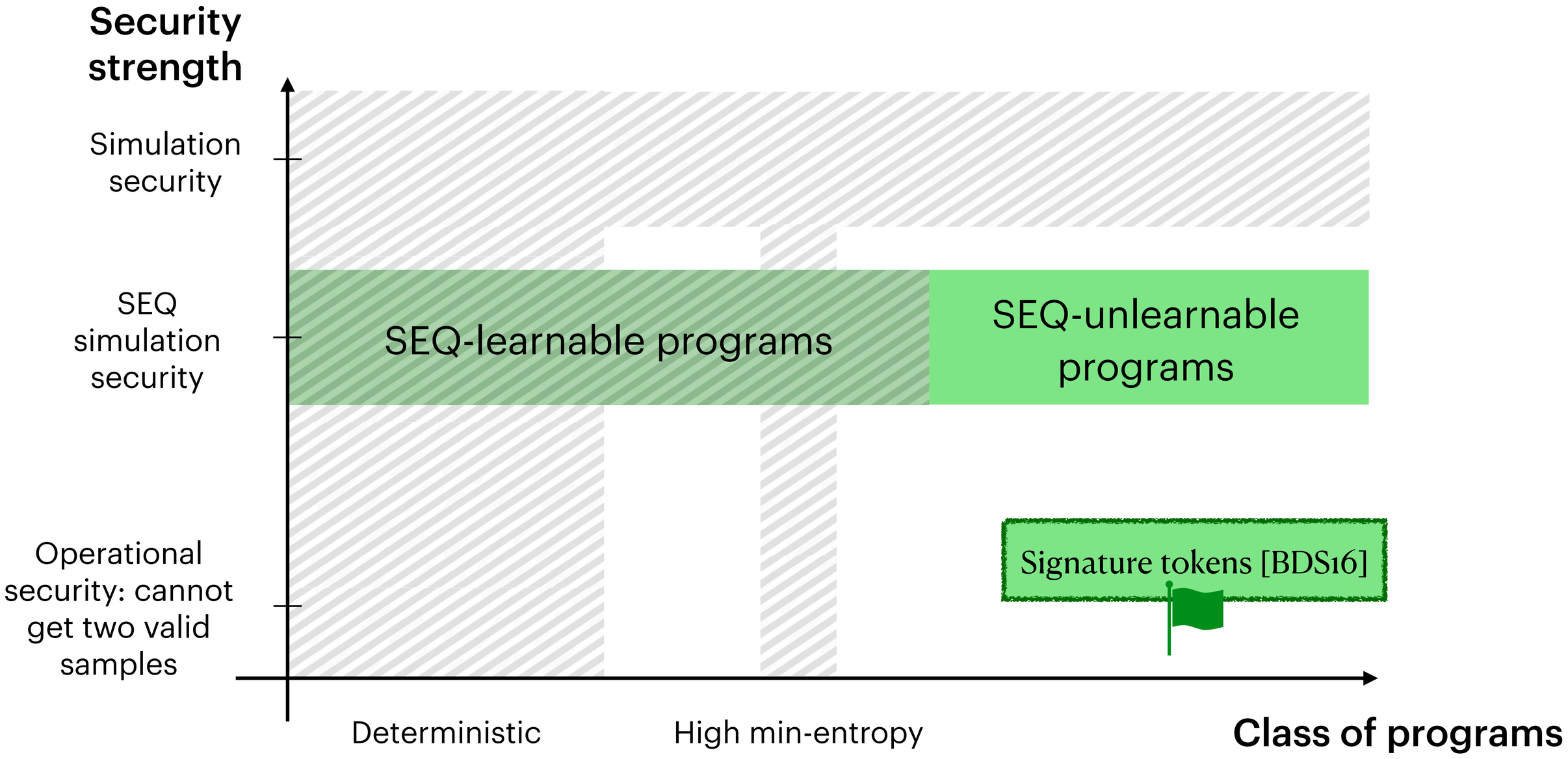
1. Stronger-than-operational notion of OTP that is achievable?

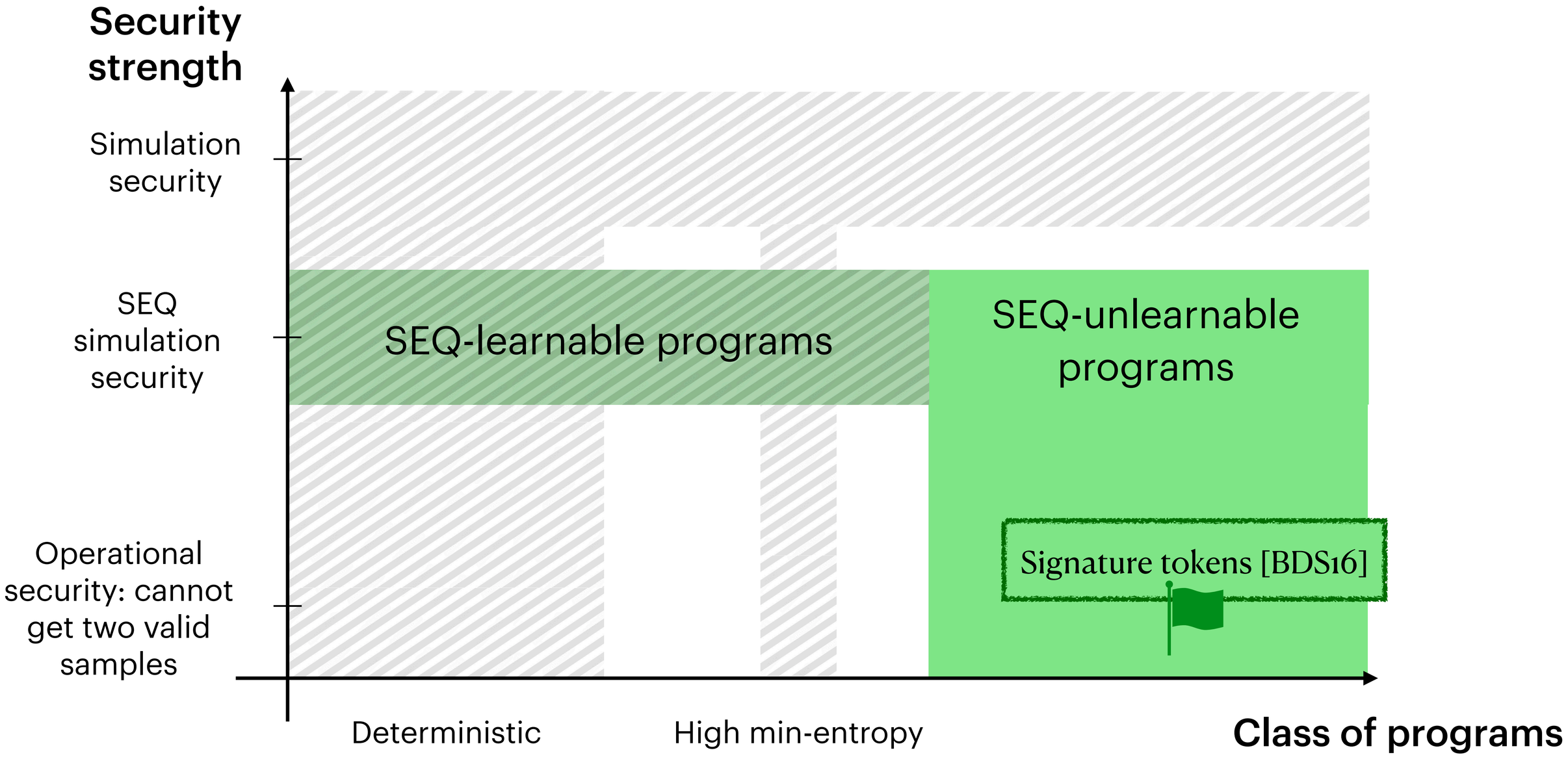
SEQ simulation security

2. What class of programs can be compiled into an OTP?

SEQ-unlearnable functions


Security strength
Simulation security
Operational security: cannot get two valid samples
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
SEQ simulation security
Operational security: cannot get two valid samples
SEQ-learnable programs
SEQ-unlearnable programs
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs


Security strength
Simulation security
SEQ simulation security
Operational security: cannot get two valid samples
SEQ-learnable programs
SEQ-unlearnable programs
Signature tokens [BDS16]
Deterministic
High min-entropy
Class of programs

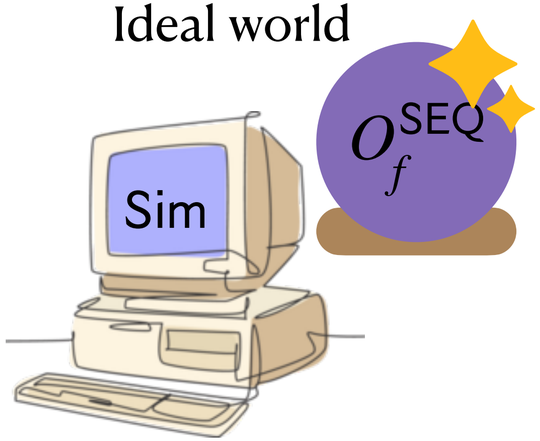
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



$\approx_c$

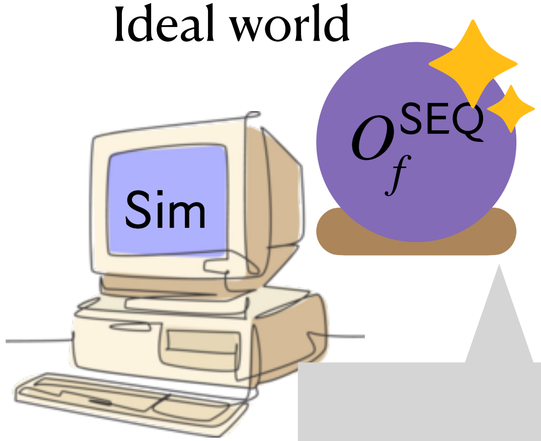
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



Single Effective query oracle:
At most 1 **destructive** query

$\approx_c$

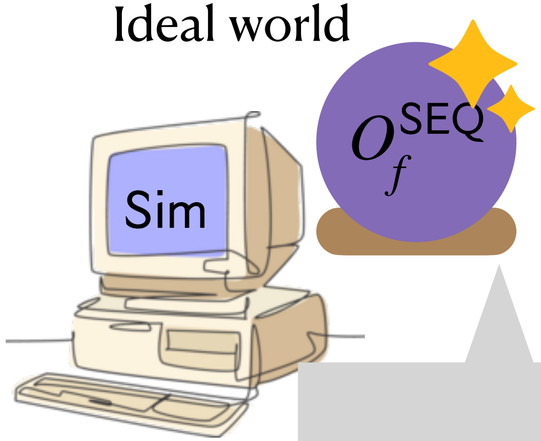
Our Definition: Single Effective Query Model

Allow simulator also to make gentle queries and un-computations

Real world

$|\mathrm{OTP}_f\rangle$

Ideal world



Single Effective query oracle:
At most 1 **destructive** query

$\approx_c$

Keep track of destructive queries using Zhandry's compressed oracle technique.

Defining the “Single Effective Query” Oracle

Defining the “Single Effective Query” Oracle

- Step 1: SEQ **Random** oracle

Defining the “Single Effective Query” Oracle

- Step 1: SEQ **Random** oracle
  - Use compressed oracle technique as a blackbox

Bonus slides

Defining the “Single Effective Query” Oracle

- Step 1: SEQ **Random** oracle
  - Use compressed oracle technique as a blackbox
- Step 2: SEQ oracle for randomized functions

Bonus slides

Step 0: *Classical* SEQ Random oracle

Step 0: *Classical* SEQ Random oracle

x

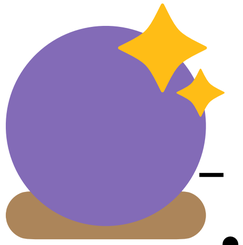
Step 0: *Classical* SEQ Random oracle

x →

← $H(x)$

or $\perp$ after 1st query

Step 0: *Classical* SEQ Random oracle



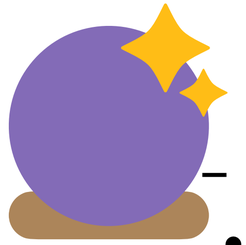
x

$H(x)$

or $\perp$ after 1st query

- Initialize empty database D

Step 0: *Classical* SEQ Random oracle



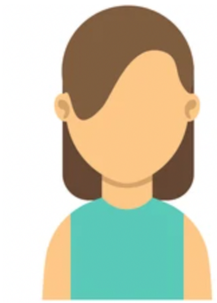
x →

← $H(x)$

or $\perp$ after 1st query

- Initialize empty database D
- On query x:

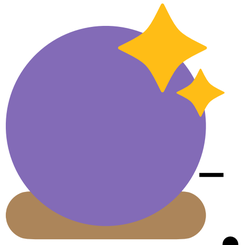
Step 0: *Classical* SEQ Random oracle



x

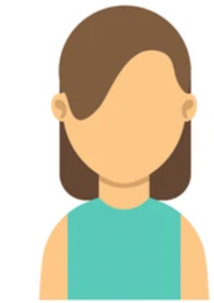
$H(x)$

or $\perp$ after 1st query



- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$

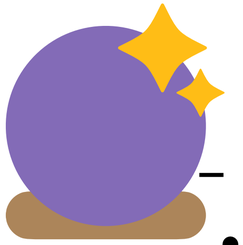
Step 0: *Classical* SEQ Random oracle



x

$H(x)$

or $\perp$ after 1st query



- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$
  - Else:
    - Output $\perp$

Step 1: Quantum SEQ Random oracle

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle$$

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

- Initialize empty database D

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

- Initialize empty database D
- On query x:

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$

How to record quantum query?

Step 1: Quantum SEQ Random oracle

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

$$\sum_x \alpha_x |x,\ H(x)\rangle \longleftarrow$$

$$\sum_x \alpha_x |x,0\rangle \longleftarrow$$

- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$

How to record quantum query?

Step 1: Quantum SEQ Random oracle

$\sum_x \alpha_x |x,0\rangle$ →

$\sum_x \alpha_x |x,\ H(x)\rangle$ ←

→

$\sum_x \alpha_x |x,0\rangle$ ←

Allow un-computation

- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$
  - If current query == last query
    - Output $H(x)$ and erase record

How to record quantum query?

Step 1: Quantum SEQ Random oracle

$\sum_x \alpha_x |x,0\rangle$ →

$\sum_x \alpha_x |x,\ H(x)\rangle$ ←

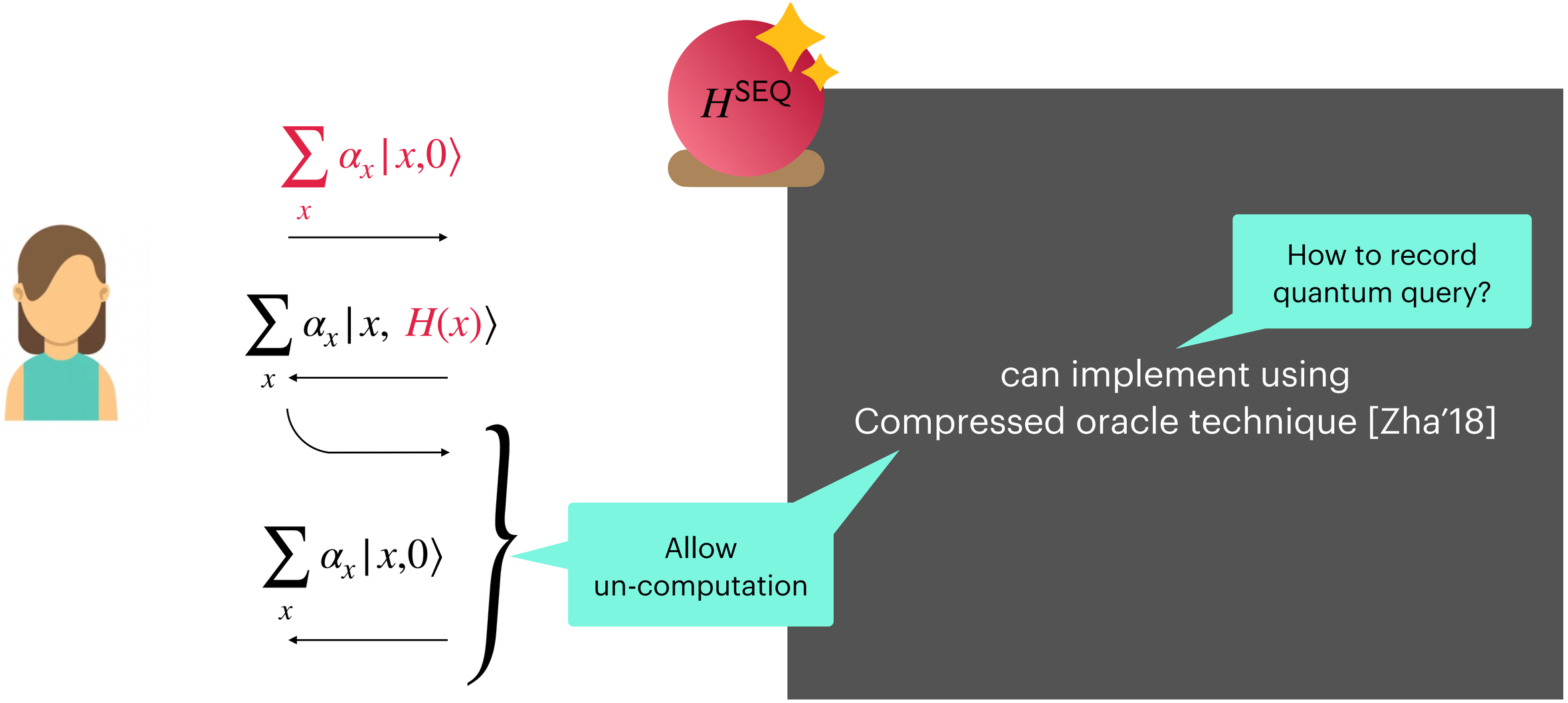
$\sum_x \alpha_x |x,0\rangle$ ←

Allow un-computation

- Initialize empty database D
- On query x:
  - If D is empty
    - Record x and output $H(x)$
  - If current query == last query
    - Output $H(x)$ and erase record
  - Else:
    - Output $\perp$

How to record quantum query?

Step 1: Quantum SEQ Random oracle

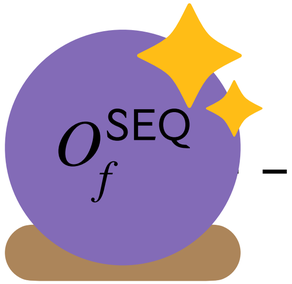


Step 2: SEQ oracle for $f : X \times R \to Y$

$$\sum_x \alpha_x |x,0\rangle \longrightarrow$$

Step 2: SEQ oracle for $f : X \times R \to Y$

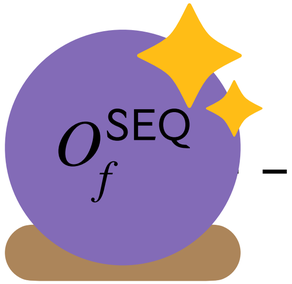
$$\sum_x \alpha_x |x,0\rangle$$



Use H^{SEQ} as a subroutine to generate randomness $H(x)$

Step 2: SEQ oracle for $f : X \times R \to Y$

$\sum_x \alpha_x |x,0\rangle$

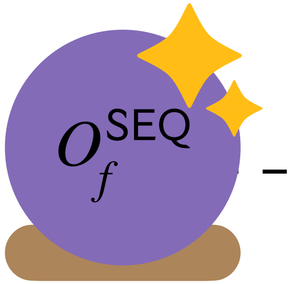


Use H^{SEQ} as a subroutine to generate randomness $H(x)$

$\sum_x \alpha_x |x,0\rangle |0\rangle$

Step 2: SEQ oracle for $f : X \times R \to Y$

$\sum_x \alpha_x |x,0\rangle$

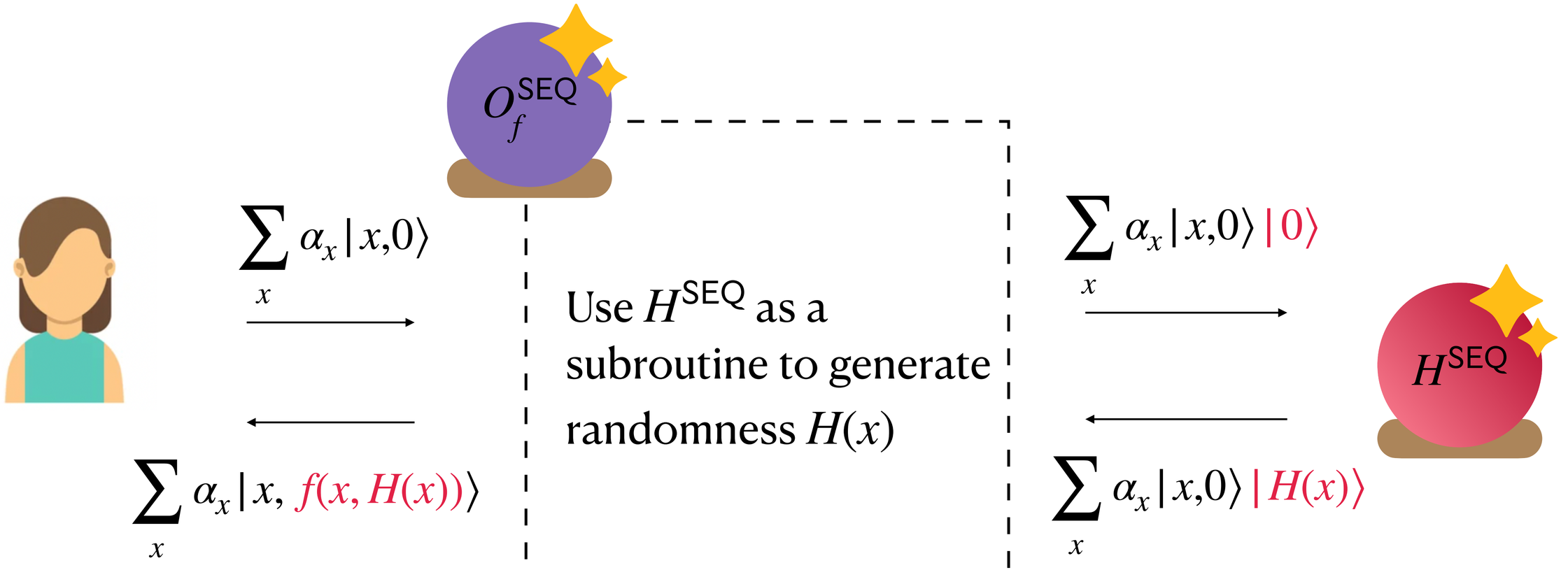


Use H^{SEQ} as a subroutine to generate randomness $H(x)$

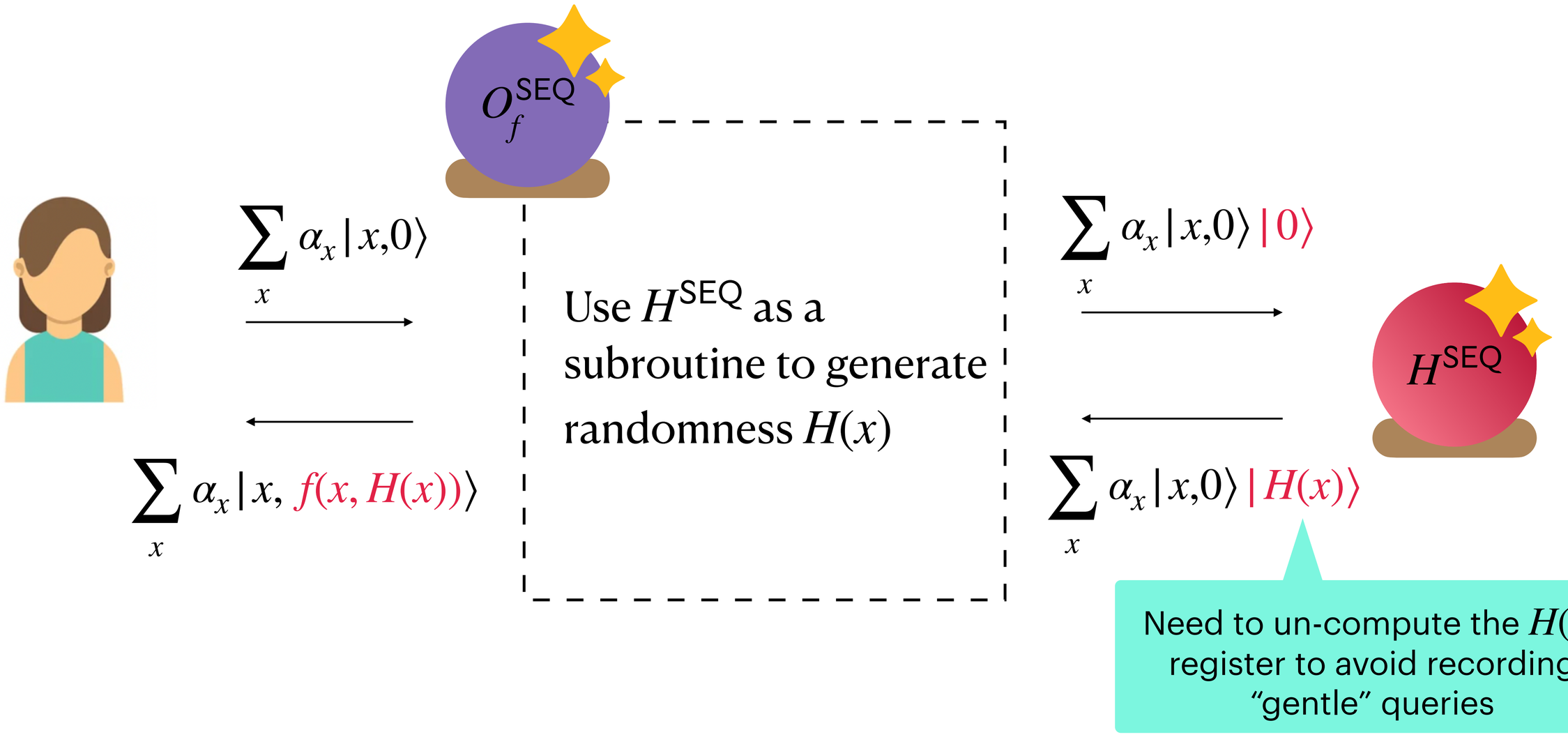
$\sum_x \alpha_x |x,0\rangle |0\rangle$

$\sum_x \alpha_x |x,0\rangle |H(x)\rangle$

Step 2: SEQ oracle for $f : X \times R \to Y$



Step 2: SEQ oracle for $f : X \times R \to Y$

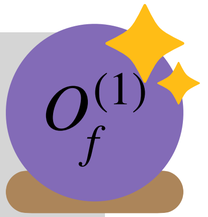


One-time security definitions

Simulation security

Operational security

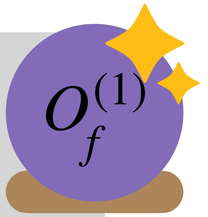
Attempt #1:
One-time oracle



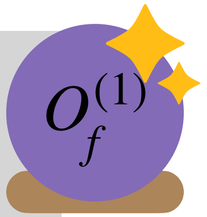
One-time security definitions

Simulation security

Attempt #1:
One-time oracle



Attempt #2:
Classical output adv

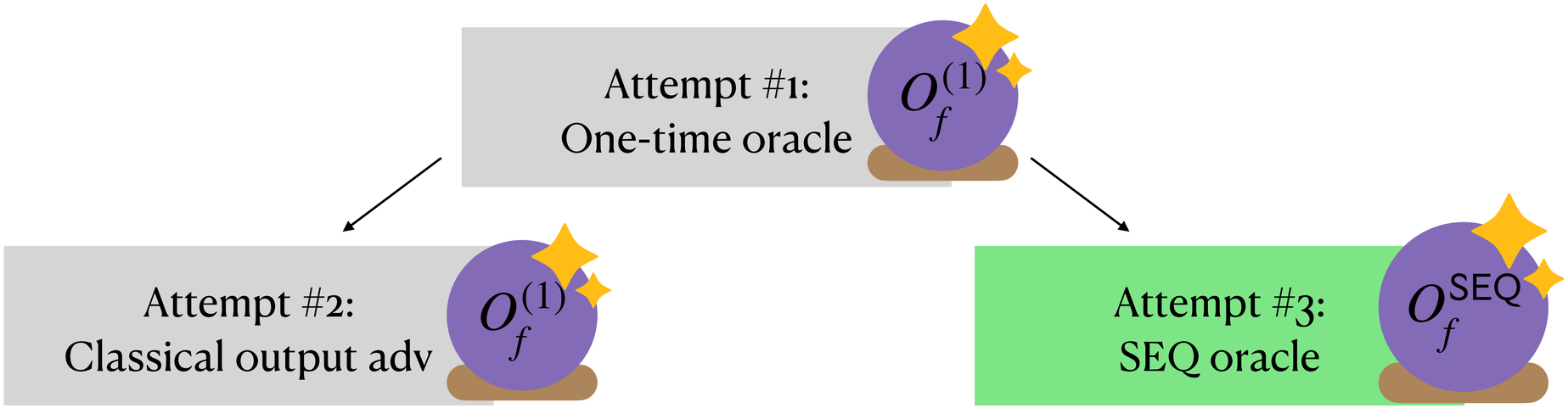


Operational security

One-time security definitions

Simulation security

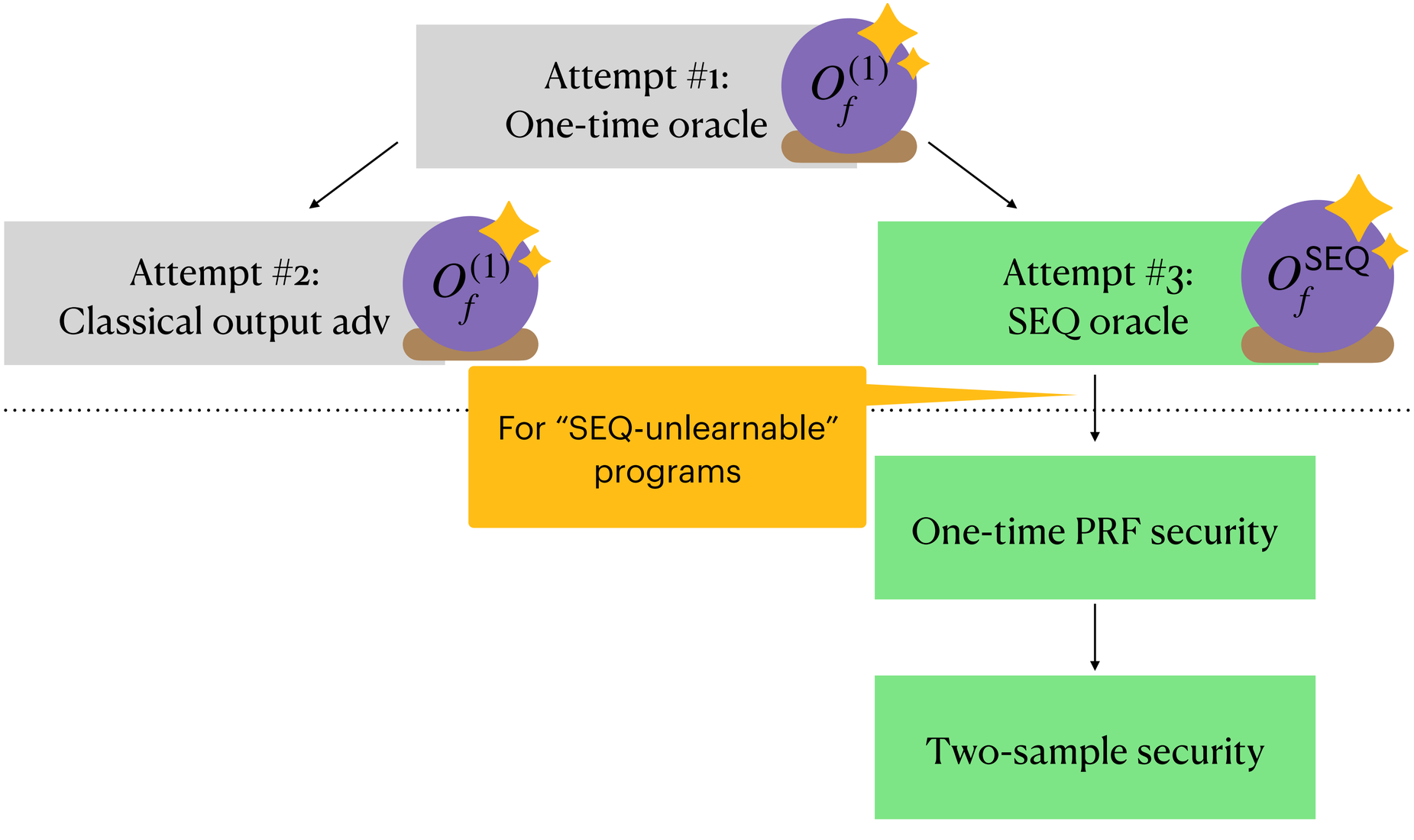
Operational security



One-time security definitions

Simulation security

Operational security



Instantiating OTPs in the Plain Model

Instantiating our OTP the Plain Model

- Use IO to heuristically instantiate oracles

Instantiating our OTP the Plain Model

- Use IO to heuristically instantiate oracles

Theorem [GLRRV'24]

Post-quantum IO + LWE $\Longrightarrow$ OTP for randomized constrained PRFs in plain model

Instantiating our OTP the Plain Model

- Use IO to heuristically instantiate oracles

Theorem [GLRRV'24]

Post-quantum IO + LWE $\Longrightarrow$ OTP for randomized constrained PRFs in plain model

Theorem [GLRRV'24]

$\exists$ SEQ-unlearnable programs that have no OTP in the plain model.

Randomized Constrained PRFs

[KPTZ13, BW13, BGI14]

Randomized Constrained PRFs

[KPTZ13, BW13, BGI14]

Randomized PRF

$$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$$

for PRF F

Randomized Constrained PRFs

[KPTZ13, BW13, BGI14]

Randomized PRF

$$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$$

for PRF F

Constrained PRFs. Given $k_C \leftarrow \mathsf{Constrain}(k, C)$,

Randomized Constrained PRFs

[KPTZ13, BW13, BGI14]

Randomized PRF

$$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$$

for PRF F

Constrained PRFs. Given $k_C \leftarrow \mathsf{Constrain}(k, C)$,

- Can evaluate $\mathsf{Eval}(k_C, x) = F_k(x)$ for all $C(x) = 1$

Randomized Constrained PRFs

[KPTZ13, BW13, BGI14]

Randomized PRF

$$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$$

for PRF F

Constrained PRFs. Given $k_C \leftarrow \mathsf{Constrain}(k, C)$,

- Can evaluate $\mathsf{Eval}(k_C, x) = F_k(x)$ for all $C(x) = 1$
- $F_k(x)$ pseudorandom for all $C(x) = 0$

One-time PRF security game

“Two-valid-outputs” game might not be secure enough for some programs, like PRFs

Randomized PRF

$$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$$

for PRF F

One-time PRF security game

“Two-valid-outputs” game might not be secure enough for some programs, like PRFs

$|\mathrm{OTP}_{F'_k}\rangle$ ←

Sample PRF key k

Randomized PRF

$$F'_k(x;r) = \left(r\,,\ F_k(x\|r)\right)$$

for PRF F

One-time PRF security game

“Two-valid-outputs” game might not be secure enough for some programs, like PRFs

$|\mathsf{OTP}_{F'_k}\rangle$

$(x_1, r_1), (x_2, r_2)$

Sample PRF key k

Randomized PRF

$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$

for PRF F

One-time PRF security game

“Two-valid-outputs” game might not be secure enough for some programs, like PRFs

$|\mathsf{OTP}_{F'_k}\rangle$

$(x_1, r_1), (x_2, r_2)$

y_1, y_2

Sample PRF key k

Sample $b_1, b_2 \leftarrow \{0,1\}$

If $b_i = 0$ then $y_i := F_k(x_i \| r_i)$

If $b_i = 1$ then $y_i \leftarrow$ random

Randomized PRF

$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$

for PRF F

One-time PRF security game

"Two-valid-outputs" game might not be secure enough for some programs, like PRFs

$|\mathrm{OTP}_{F'_k}\rangle$

$(x_1, r_1), (x_2, r_2)$

y_1, y_2

$\hat{b}_1, \hat{b}_2$

Sample PRF key k

Sample $b_1, b_2 \leftarrow \{0,1\}$

If $b_i = 0$ then $y_i := F_k(x_i \| r_i)$

If $b_i = 1$ then $y_i \leftarrow$ random

Adv wins if:

$b_1 = \hat{b}_1$ and $b_2 = \hat{b}_2$

Randomized PRF

$F'_k(x; r) = \left(r\,,\ F_k(x \| r)\right)$

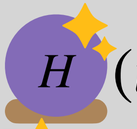
for PRF F

OTP for constrained PRF F'_k

Construction in plain model

Classical oracle construction: $\mathsf{OTP}_f := (|S\rangle,$  $)$

$\underline{O(x,u)}$:

$r \leftarrow$  (u)

If  $(u) = 1$: output $F'_k(x;r)$

Else: abort

OTP for constrained PRF F'_k

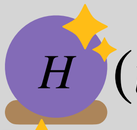
Construction in plain model

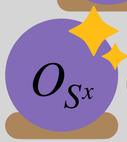
Plain model

~~Classical oracle~~ construction:

$\mathsf{OTP}_f := (|S\rangle,$  $)$

$\underline{O(x,u)}$:

$r \leftarrow$  (u)

If  $(u) = 1$: output $F'_k(x;r)$

Else: abort

OTP for constrained PRF F'_k

Construction in plain model

Plain model

~~Classical oracle~~ construction:

$\mathsf{OTP}_f := (|S\rangle, \mathsf{iO}(O))$

$\underline{O(x,u)}$:

$r \leftarrow H(u)$

If $O_{S^x}(u) = 1$: output $F'_k(x;r)$

Else: abort

OTP for constrained PRF F'_k

Construction in plain model

Plain model

~~Classical oracle~~ construction:

$\mathsf{OTP}_f := (|S\rangle, \mathsf{iO}(O))$

$\underline{O(x,u)}$:

$r \leftarrow H(u)$

If $\mathsf{iO}(O_{S^x})(u) = 1$: output $F'_k(x;r)$

Else: abort

OTP for constrained PRF F'_k

Construction in plain model

Plain model

~~Classical oracle~~ construction:

$$\mathsf{OTP}_f := (|S\rangle, \mathsf{iO}(O))$$

$\underline{O(x, u)}$:

$r \leftarrow \mathsf{InvertiblePRF}(u)$

If $\mathsf{iO}(O_{S^x})(u) = 1$: output $F'_k(x; r)$

Else: abort

OTP for constrained PRF F'_k

Construction in plain model

Plain model

~~Classical oracle~~ construction:

$$\mathsf{OTP}_f := (|S\rangle, \mathsf{iO}(O))$$

$\underline{O(x,u)}$:

$r \leftarrow \mathsf{InvertiblePRF}(u)$

If $\mathsf{iO}(O_{S^x})(u) = 1$: output $F'_k(x;r)$

Else: abort

Proof sketch:

- Constrain PRF F'_k to r whose preimages are valid subspace vectors

*Use coset states instead of subspace states for plain model security proof [CLLZ21, VZ21]

OTP for constrained PRF F'_k

Construction in plain model

Plain model

~~Classical oracle~~ construction:

$\mathsf{OTP}_f := (|S\rangle, \mathsf{iO}(O))$

$\underline{O(x,u)}$:

$r \leftarrow \mathsf{InvertiblePRF}(u)$

If $\mathsf{iO}(O_{S^x})(u) = 1$: output $F'_k(x;r)$

Else: abort

Proof sketch:

- Constrain PRF F'_k to r whose preimages are valid subspace vectors
- Then, adversary must have found $s \in S, s' \in S^\perp$

*Use coset states instead of subspace states for plain model security proof [CLLZ21, VZ21]

Applications

Applications

- **More signature tokens:**
  - Blind unforgeable signatures [AMRS20] $\Longrightarrow$ signature tokens

Applications

- **More signature tokens:**
  - Blind unforgeable signatures [AMRS20] $\Longrightarrow$ signature tokens
- **One-time NIZK proofs**
  - One-time proving token that allows a prover to prove only one statement

Applications

- **More signature tokens:**
  - Blind unforgeable signatures [AMRS20] $\Longrightarrow$ signature tokens
- **One-time NIZK proofs**
  - One-time proving token that allows a prover to prove only one statement
- **Quantum money**
  - One-time program for a signature scheme

Open Question #1

Strongest achievable notion of one-time security?

- We gave the SEQ simulation security definition
- Is it the strongest notion achievable?

Open Question #2

More applications for one-time **randomized** programs?

- We suggest a few: more signature tokens, one-time proofs, quantum money
- Flagship application of [Gunn-Movassagh'24] is Generative AI/LLMs
    - Heuristic security, difficult to prove

Open Question #3

Construction for one-time quantum channels?

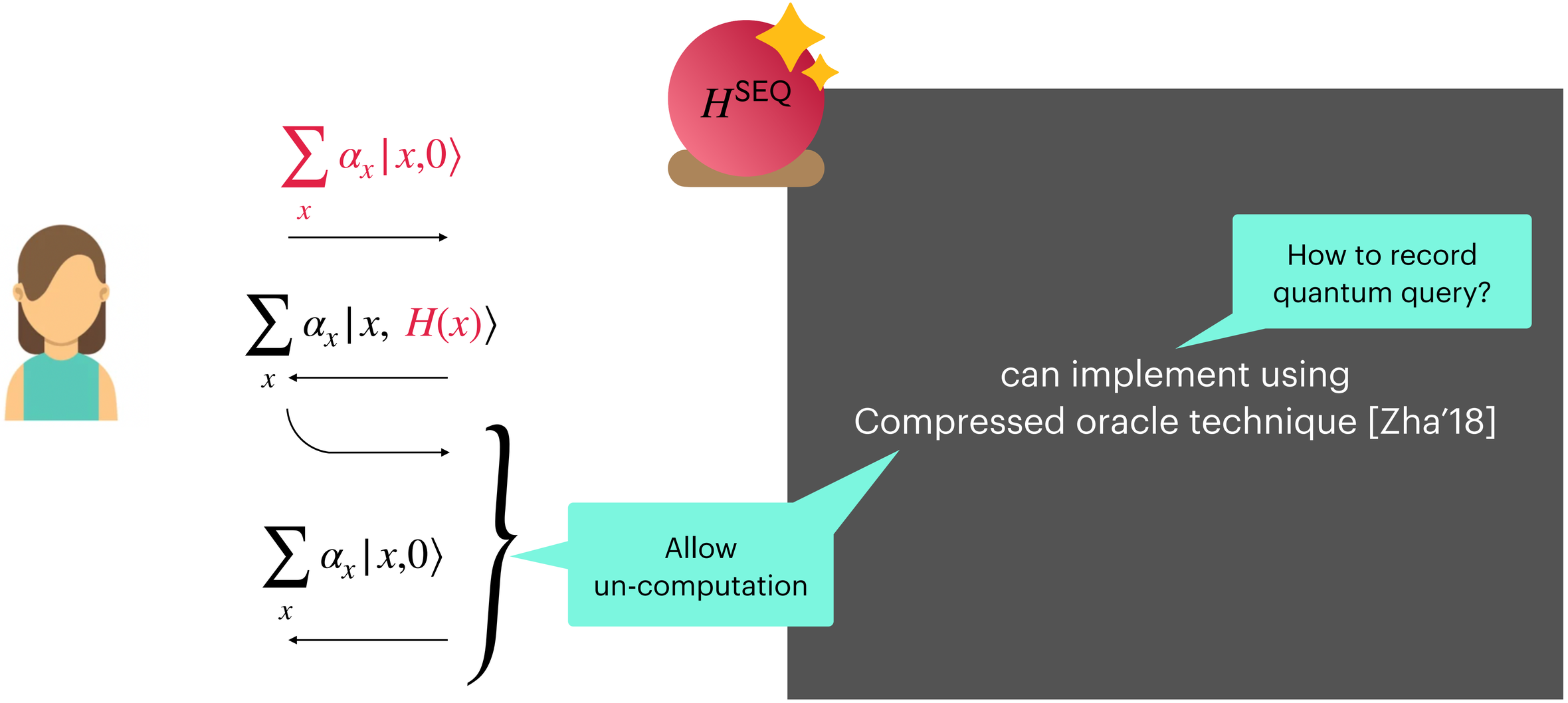
- [BGS13] give a construction assuming one-time memory hardware devices
- Not clear how to generalize our construction
  - We used unclonable subspace states to generate classical randomness
  - How to tamper with inherent quantum randomness?

Thank you!

Bonus slides I

Compressed Oracle Technique

Step 1: Quantum SEQ Random oracle



Compressed Oracle [Zha'18]

Compressed Oracle [Zha'18]

- Technique developed to record quantum queries to random oracle

Compressed Oracle [Zha'18]

- Technique developed to record quantum queries to random oracle
- Useful for
    - On-the-fly simulation of quantum-accessible random oracles

Compressed Oracle [Zha'18]

- Technique developed to record quantum queries to random oracle
- Useful for
    - On-the-fly simulation of quantum-accessible random oracles
    - (Re-)proving quantum query lower bounds

Compressed Oracle [Zha'18]

- Technique developed to record quantum queries to random oracle
- Useful for
    - On-the-fly simulation of quantum-accessible random oracles
    - (Re-)proving quantum query lower bounds

Quantum queries to random oracle H

Action on standard basis states: $|x, u\rangle \mapsto |x, u + H(x)\rangle$

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus H(x)\rangle \otimes |H\rangle$$

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus H(x)\rangle \otimes |H\rangle$$

- Writing $H : [N] \to R$ as a truth table,

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus H(x)\rangle \otimes |H\rangle$$

- Writing $H : [N] \to R$ as a truth table,

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$\sum_{r_x} |x, u \oplus r_x\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus H(x)\rangle \otimes |H\rangle$$

- Writing $H : [N] \to R$ as a truth table,

Measure output

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$|x, u \oplus r_x\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Oracle register collapses too.
Can detect this in the Fourier basis.

Compressed Oracle [Zha'18]

Purify the randomness of the RO to record adversary's quantum queries

- Purify oracle register $\sum_H |H\rangle$
- Oracle query entangles query and oracle registers; detect this entanglement just looking at oracle register

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus H(x)\rangle \otimes |H\rangle$$

- Writing $H : [N] \to R$ as a truth table,

Measure output

Oracle register collapses too.
Can detect this in the Fourier basis.

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$|x, u \oplus r_x\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Compressed oracle: Isometry to convert long truth table to short databases. [Zha'18]

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x,H(x))\rangle \otimes |H\rangle$$

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x,H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x, H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$\sum_{r_x} |x, u \oplus f(x; r_x)\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x, H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Measure output

$$|x, u \oplus f(x; r_x)\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x,H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Measure output

$$|x, u \oplus f(x;r_x)\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Destructive Query

- Highly random f, oracle register collapses
- Can detect in Fourier basis

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x,H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$|x, u \oplus f(x;r_x)\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Measure output

Gentle Query

- Eg. Deterministic program: $f(x;r)$ does not depend on r
- Oracle register does not collapse

Destructive Query

- Highly random f, oracle register collapses
- Can detect in Fourier basis

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to keep track of destructive queries

- SEQ oracle uses random oracle H for randomness

$$|x,u\rangle \otimes \sum_H |H\rangle \quad \mapsto \quad \sum_H |x, u \oplus f(x,H(x))\rangle \otimes |H\rangle$$

- Writing as a truth table

$$|x,u\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

$$\downarrow$$

$$|x, u \oplus f(x;r_x)\rangle \otimes \sum_{r_1} |r_1\rangle \otimes \cdots \otimes |r_x\rangle \otimes \cdots \otimes \sum_{r_N} |r_N\rangle$$

Measure output

Gentle Query

- Eg. Deterministic program: $f(x;r)$ does not depend on r
- Oracle register does not collapse

Destructive Query

- Highly random f, oracle register collapses
- Can detect in Fourier basis

Compressed oracle: Isometry to convert long truth table to short databases. [Zha'18]

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to maintain a database of queries made

Disclaimer: (Very) informal description of the SEQ oracle

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to maintain a database of queries made

- Initialize empty compressed oracle database for RO

Disclaimer: (Very) informal description of the SEQ oracle

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to maintain a database of queries made

- Initialize empty compressed oracle database for RO
- On query $|x, u\rangle$,
  - If database contains no entries, answer query

#recorded queries: 0 → 1

Disclaimer: (Very) informal description of the SEQ oracle

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to maintain a database of queries made

- Initialize empty compressed oracle database for RO
- On query $|x, u\rangle$,
  - If database contains no entries, answer query
  - If database contains x, answer query

#recorded queries: 0 → 1

Uncomputation: #recorded queries: 1 → 0

Disclaimer: (Very) informal description of the SEQ oracle

Single Effective Query (SEQ) Model

Use Zhandry's compressed oracle technique to maintain a database of queries made

- Initialize empty compressed oracle database for RO
- On query $|x, u\rangle$,
  - If database contains no entries, answer query
  - If database contains x, answer query
  - If database contains $x' \neq x$, don't answer query

#recorded queries: 0 → 1

Uncomputation: #recorded queries: 1 → 0

#recorded queries: 1 → 1

Disclaimer: (Very) informal description of the SEQ oracle

Bonus slides II

Plain model barrier

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input	Output
x	$f_{sk,\, a,\, b}\,(x\,;\,r)$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	**Output** $f_{sk, a, b}(x\,;\, r)$
a	$Enc_{sk}(b;\, r)$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	**Output** $f_{sk, a, b}(x\,;\, r)$
a	$Enc_{sk}(b;\, r)$
$x \neq a$	$Enc_{sk}(x;\, r)$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$Enc_{sk}(b;\,r)$
$x \neq a$	$Enc_{sk}(x;\,r)$

aux

- QFHE public key pk'
- $\mathrm{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$Enc_{sk}(b;\,r)$
$x \neq a$	$Enc_{sk}(x;\,r)$

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.
Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$\mathsf{Enc}_{sk}(b;\,r)$
$x \neq a$	$\mathsf{Enc}_{sk}(x;\,r)$

Plain model $|\mathsf{OTP}_f\rangle$ + aux

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.
Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$\mathsf{Enc}_{sk}(b;\,r)$
$x \neq a$	$\mathsf{Enc}_{sk}(x;\,r)$

Plain model $|\mathsf{OTP}_f\rangle$ + aux

↓ **QFHE evaluation** on $\mathsf{Enc}_{pk'}(a)$

$\mathsf{Enc}_{sk'}(\mathsf{Enc}_{sk}(b))$

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$\mathsf{Enc}_{sk}(b;\,r)$
$x \neq a$	$\mathsf{Enc}_{sk}(x;\,r)$

Plain model $|\mathsf{OTP}_f\rangle$ + aux

↓ **QFHE evaluation** on $\mathsf{Enc}_{pk'}(a)$

$\mathsf{Enc}_{sk'}(\mathsf{Enc}_{sk}(b)) \longrightarrow \hat{P} \longrightarrow (sk, sk')$

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$\mathsf{Enc}_{sk}(b;\,r)$
$x \neq a$	$\mathsf{Enc}_{sk}(x;\,r)$

Plain model $|\mathsf{OTP}_f\rangle$ + aux

↓ **QFHE evaluation** on $\mathsf{Enc}_{pk'}(a)$

$\mathsf{Enc}_{sk'}(\mathsf{Enc}_{sk}(b)) \longrightarrow \hat{P} \longrightarrow (sk, sk')$

↓

(a, b)

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$

Theorem [GLRRV'24]

∃ SEQ-unlearnable programs that have no OTP in the plain model.

Inspired by [ABDS20, AP21]

Input x	Output $f_{sk,\,a,\,b}(x\,;\,r)$
a	$\mathsf{Enc}_{sk}(b;\,r)$
$x \neq a$	$\mathsf{Enc}_{sk}(x;\,r)$

Plain model $|\mathsf{OTP}_f\rangle$ + aux

Plain model OTP allows you to do non-black box things like QFHE evaluation

QFHE evaluation on $\mathsf{Enc}_{pk'}(a)$

$$\mathsf{Enc}_{sk'}(\mathsf{Enc}_{sk}(b)) \longrightarrow \hat{P} \longrightarrow (sk, sk') \longrightarrow (a, b)$$

aux

- QFHE public key pk'
- $\mathsf{Enc}_{pk'}(a)$
- Obfuscation $\hat{P}$ of P

$$P(x) = \begin{cases} (sk, sk') & \text{if } \mathsf{Dec}_{sk}(\mathsf{Dec}_{sk'}(x)) = b \\ 0 & \text{otherwise} \end{cases}$$