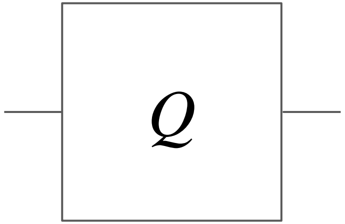
Obfuscation of Unitary Quantum Programs

Er-Cheng Tang
University of Washington

joint work with Miryam Huang

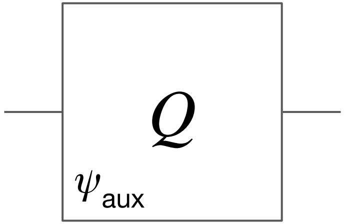
Quantum Programs

Main focus



Quantum circuit

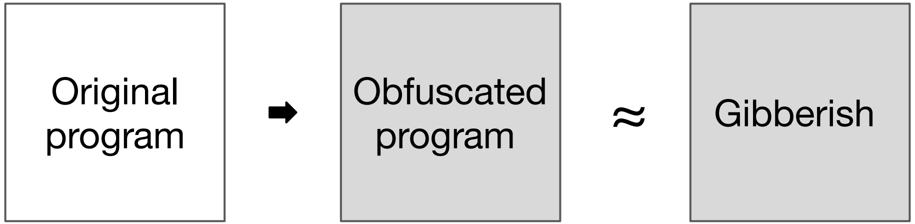
classically described



Quantum program

quantumly described

Obfuscation of Quantum Program



- Ideally, we would like to obfuscate all kinds of quantum programs.

Past Explorations

Obfuscating quantum programs is highly non-trivial:

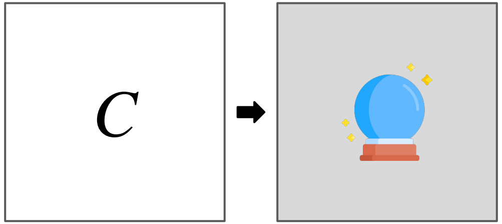
- Quantum circuits with log-many T gates (iO) [BK21]

  in the plain model

- Dream: obfuscate all kinds of quantum programs.

Classical Oracle Model

- Everyone can turn efficient classical circuits into classical oracles for free
- These classical oracles can be queried in superposition

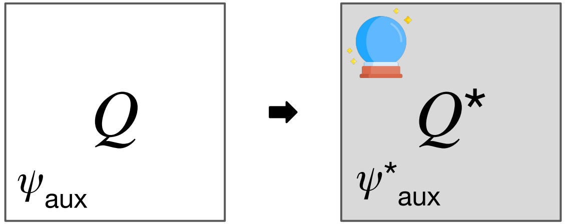


- We can obfuscate more programs in the classical oracle model
- The classical oracle can be heuristically instantiated with post-quantum iO

Past Explorations

Obfuscating quantum programs is highly non-trivial:

- Quantum circuits with log-many T gates (iO) [BK21]
- Quantum circuits for deterministic classical functions (VBB) [BKNY23]
- Quantum programs for deterministic classical functions (ideal) [BBV24]



- Dream: obfuscate all kind of quantum programs.

Past Explorations

Obfuscating quantum programs is highly non-trivial:

- Quantum circuits with log-many T gates (iO) [BK21]
- Quantum circuits for deterministic classical functions (VBB) [BKNY23]
- Quantum programs for deterministic classical functions (ideal) [BBV24]

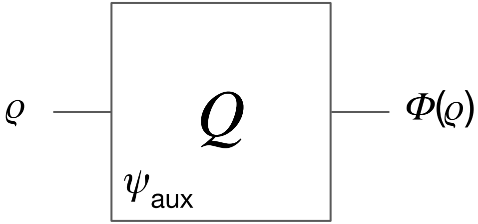
This work: Quantum programs for unitary transformations (ideal)

- Dream: obfuscate all kind of quantum programs.

Quantum Functionality

Φ maps quantum state to quantum states

Program that implements
a classical function F

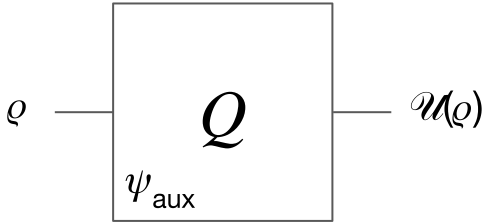


Program that implements
a quantum mapping Φ
(Embedding + Unitary + Partial Trace)

Quantum Functionality

Φ maps quantum state to quantum states

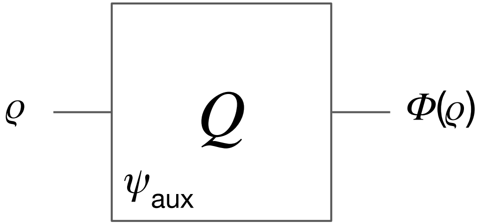
$\mathscr{U}(\varrho) = U \varrho U^{\dagger}$ for some unitary operator U



Program that implements
a unitary transformation $\mathscr{U}$

Examples

- Quantum Fourier transform
- Pseudorandom unitary



Program that implements
a quantum mapping Φ
(Embedding + Unitary + Partial Trace)

- State preparation
- Quantum error correction

Our Results

$\mathscr{U}(\varrho) = U\varrho U^\dagger$ for some unitary operator U

For unitary functionalities $\mathscr{U}$

- We define the notion of ideal obfuscation for programs with unitary functionalities
- We construct an ideal obfuscation scheme for all quantum programs with unitary functionalities in the classical oracle model
- Our obfuscated programs are reusable (for multiple evaluations)

Outline

I. Defining Ideal Obfuscation for Unitary Functionalities

II. Construction

Part I: Defining Ideal Obfuscation

for Unitary Functionalities

Philosophy Behind Ideal Obfuscation

- The only way to use an obfuscated program is to run it on some input
- Is this true in the quantum setting?
  - One can run the program
  - One can rewind the program
  - One can perform a quantum-controlled execution of the program

- If we have a program that computes a classical function F, these operations are all (black-box) *equivalent* to computing F

- If we have a program that computes a unitary transformation $\mathscr{U}$, these operations enable *more power* than computing $\mathscr{U}$

What Are These Additional Powers?

Given a program that computes a unitary transformation $\mathscr{U}$

- Rewinding of the program computes $\mathscr{U}^{-1}$
- Quantum-controlled execution can compute ctrl-$(U^{-1}AU)$ for every efficient A.
  (but not ctrl-U)

One can use the program to compute $\mathscr{U}$, $\mathscr{U}^{-1}$, and ctrl-$(U^{-1}AU)$ for every efficient A.

In fact, all of them can be computed with black-box access to ctrl-$(U^{-1}A_0U)$,
for a fixed unitary A_0 (eg. A_0 = SWAP on $2n$ qubits)

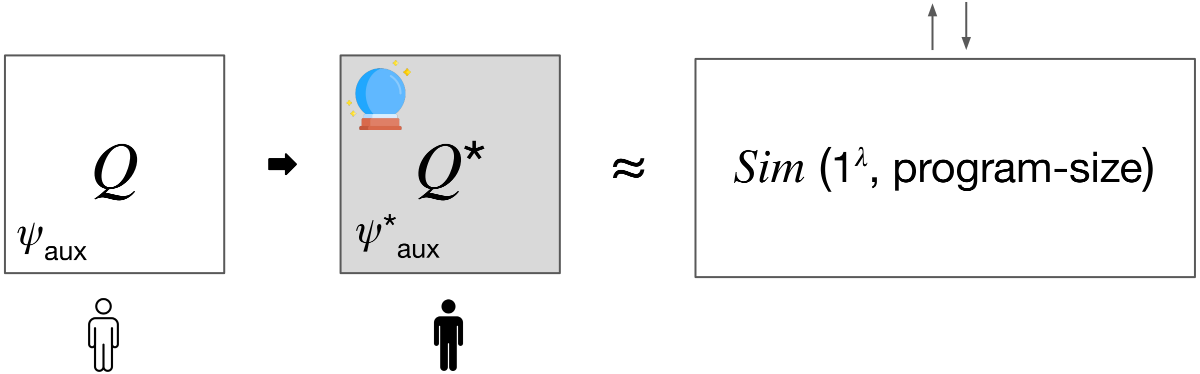
Defining Ideal Obfuscation

Obfuscator

Evaluator

If (Q, ψ_{aux}) computes $\mathscr{U}(\varrho) = U \varrho U^{-1}$

ctrl-$(U^{-1} A_0 U)$



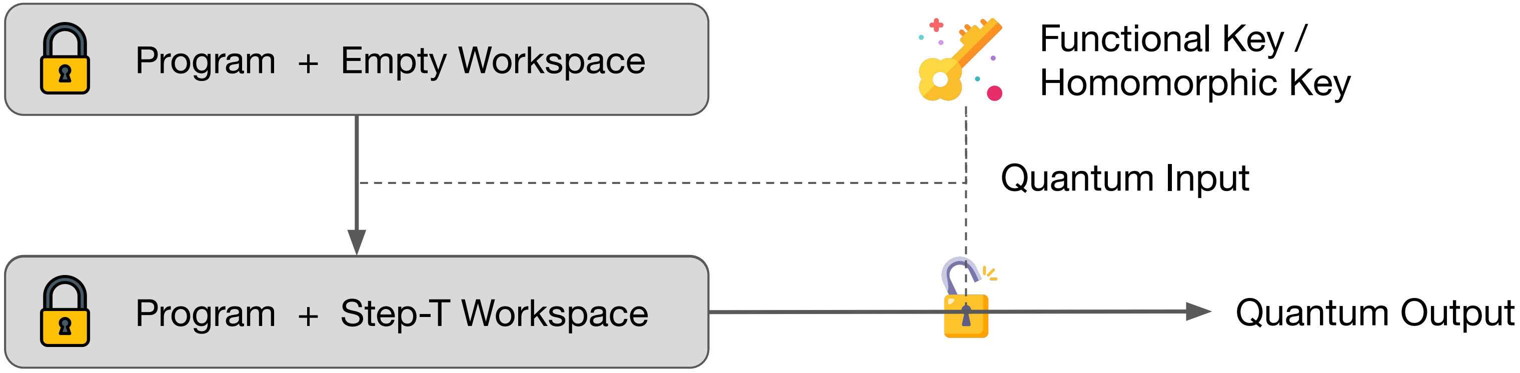
- When $\mathscr{U}$ corresponds to computing a classical function, i.e. $U|x,y\rangle = |x, y \oplus F(x)\rangle$

ctrl-$(U^{-1} A_0 U)$ ⇔ $\mathscr{U}$ ⇔ F

Our definition recovers the standard definition of ideal obfuscation

Part II: Construction

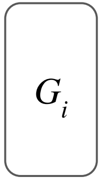
A Template for Constructing Obfuscation Schemes



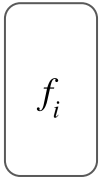
1. Start with a convenient model of quantum computation
2. Apply the quantum computation homomorphically + Final decryption
   (Encryption and authentication are both needed to ensure privacy and integrity)

1. A Model of Quantum Computation [BBV24]

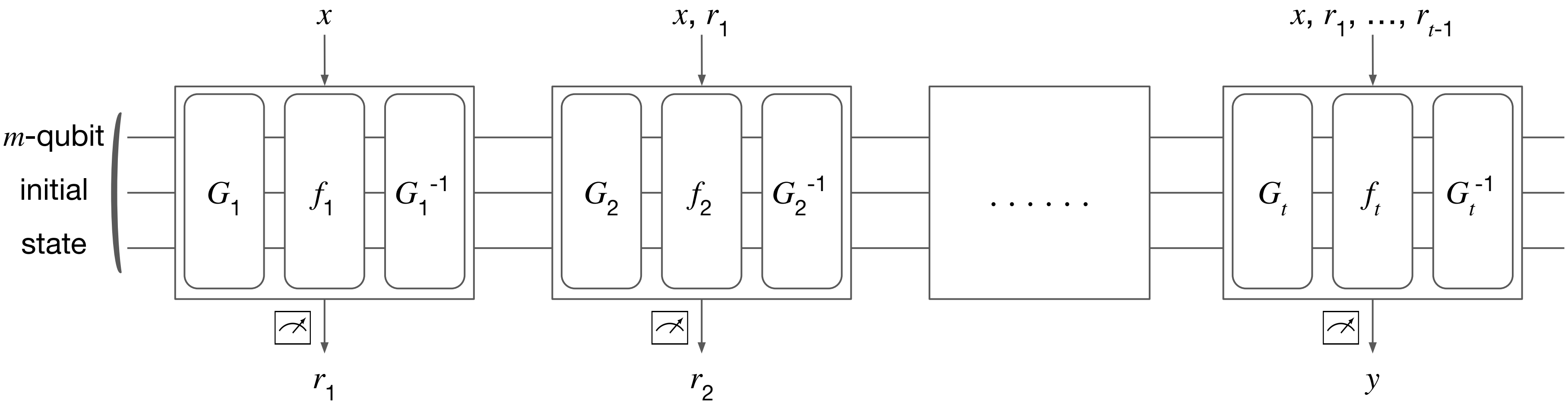
- Classical input $x \in \{0,1\}^n$ and classical output $y \in \{0,1\}^{n'}$



"CNOT+H" gate

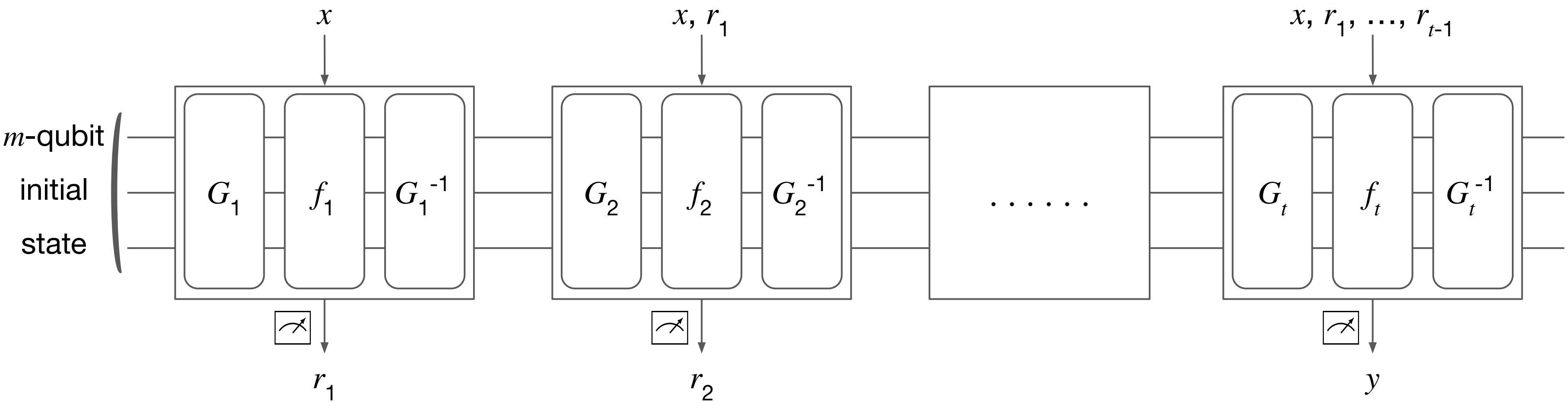


A classical function: $\{0,1\}^m \to \{0,1\}$ parametrized by $x, r_1, \ldots, r_{i-1}$

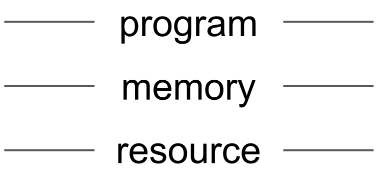


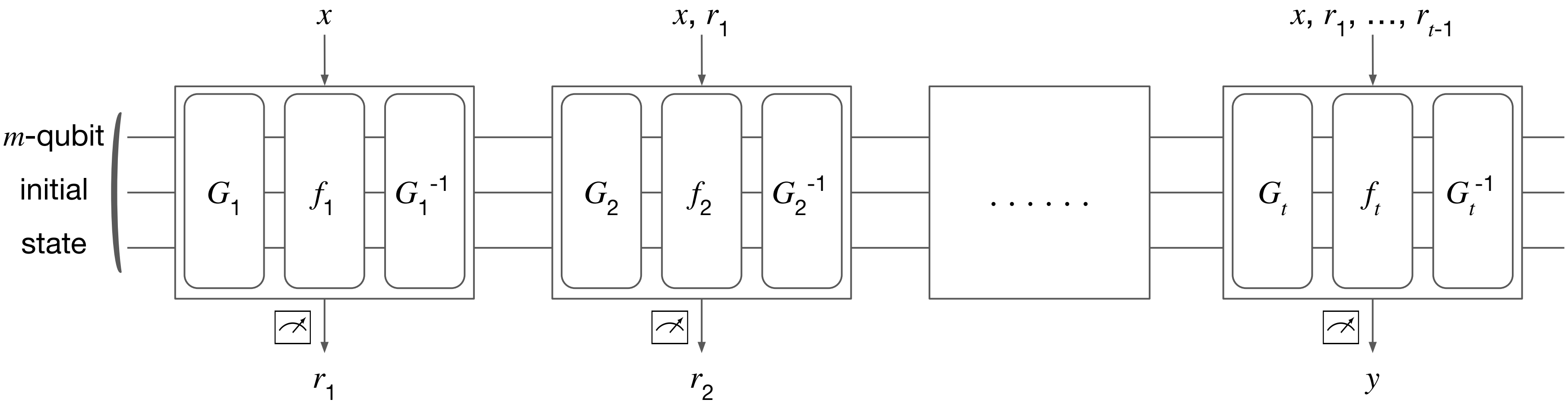
1. A Model of Quantum Computation [BBV24]

- The computation is
  - Quantum: because the measurements are across different “CNOT+H” bases
  - Universal: because adaptively-chosen functions are sufficiently expressive



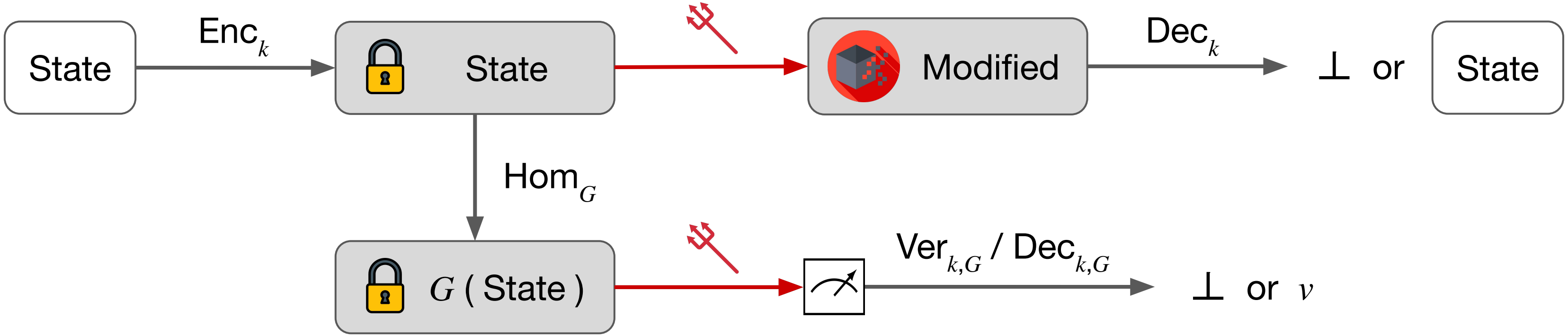
1. A Model of Quantum Computation [BBV24]

- Every efficient program can be efficiently compiled into this form with efficient f_i
- The compiler outputs a state  and descriptions of G_i, f_i



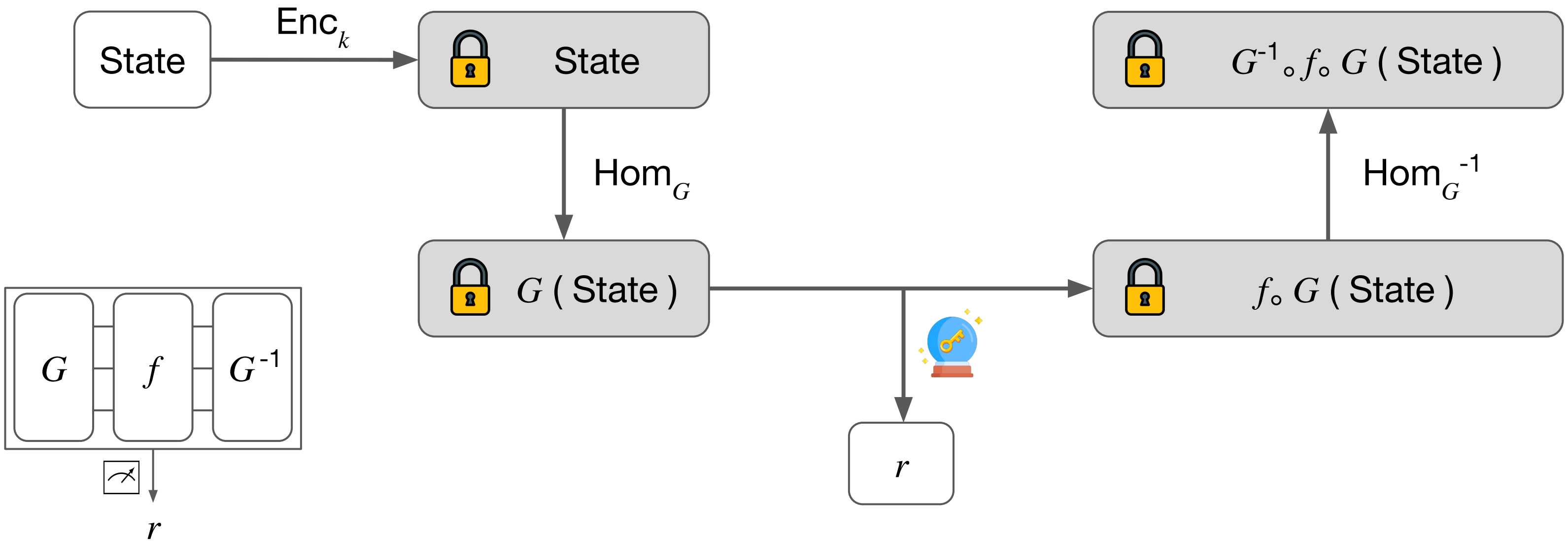
2. Quantum Authentication with Classical Decodability [BBV24]

- Quantum authentication with homomorphic measurement in “CNOT+H” bases
- Classical oracle  (eg. $f \circ \mathrm{Dec}_{k,G}$) would help with homomorphic computation



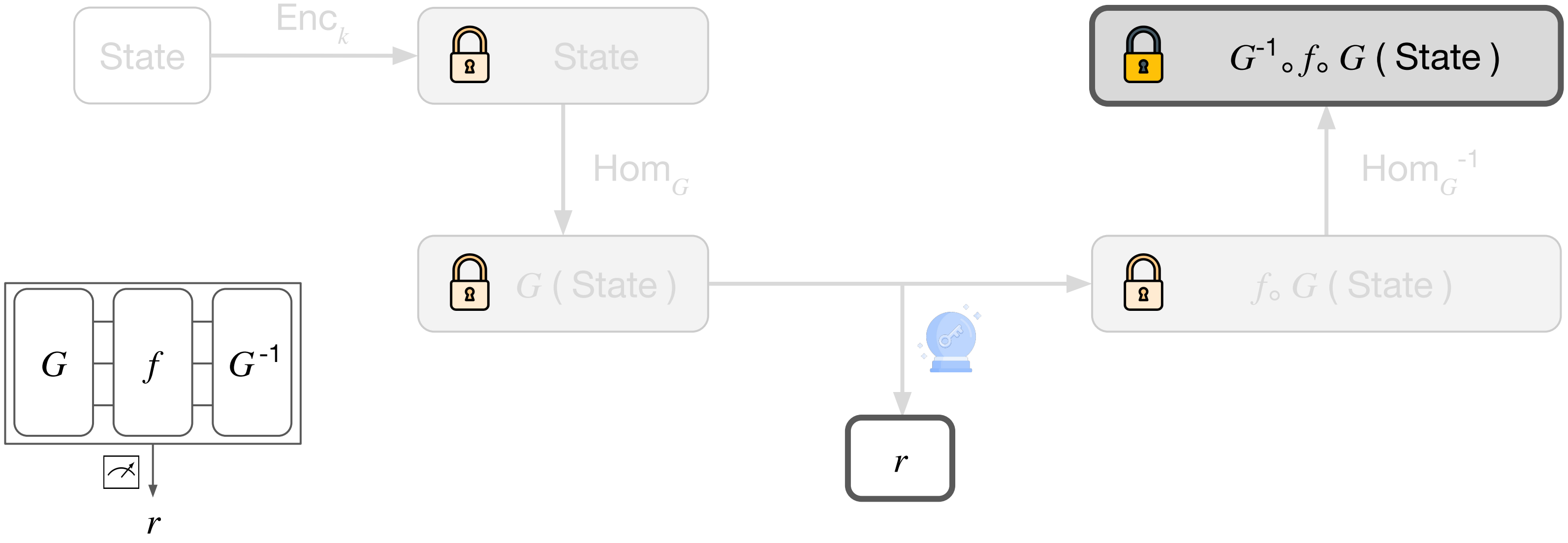
2. Quantum Authentication with Classical Decodability [BBV24]

- Classical oracle (eg. $f \circ \mathrm{Dec}_{k,G}$) would help with homomorphic computation
- Make superposition query to



2. Quantum Authentication with Classical Decodability [BBV24]

- Classical oracle (eg. $f \circ \mathrm{Dec}_{k,G}$) would help with homomorphic computation
- Make superposition query to



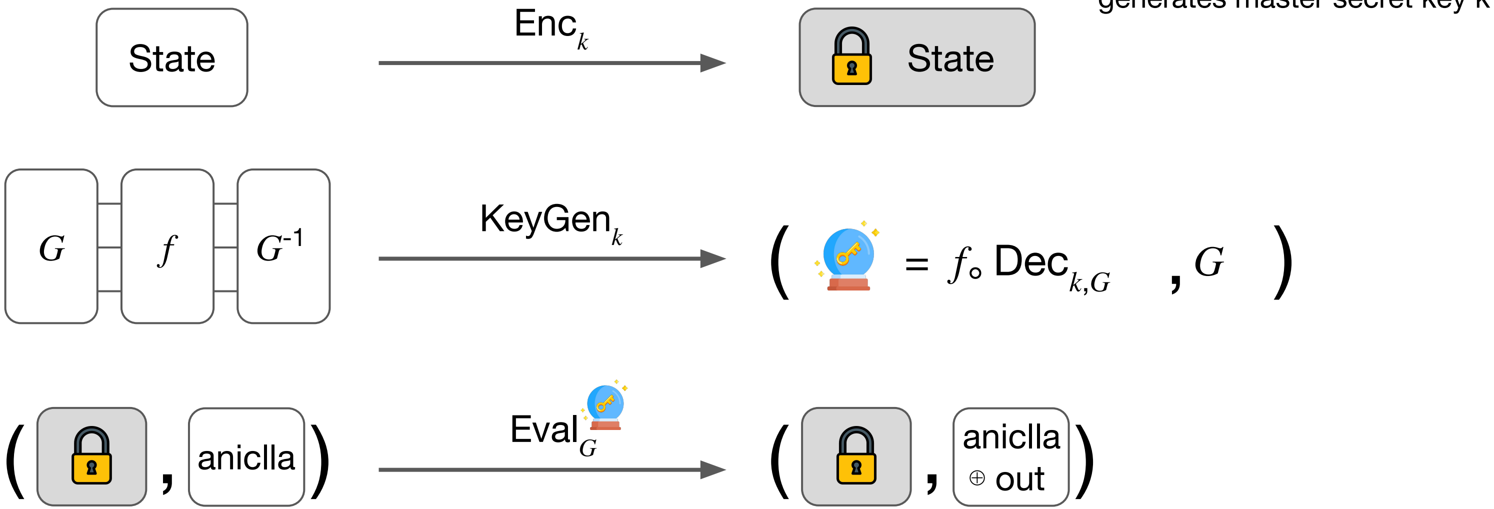
2. Quantum Auth with Classical Decodability: Security

- [BBV24] proves some property-based security (privacy, integrity, public verifiability)
- [BBV24] does not fully handle security with partial decoding power

- To remedy this, we propose and achieve simulation security of their scheme, where the adversary can have partial decoding power
- We can reinterpret the scheme as a functional quantum authentication scheme

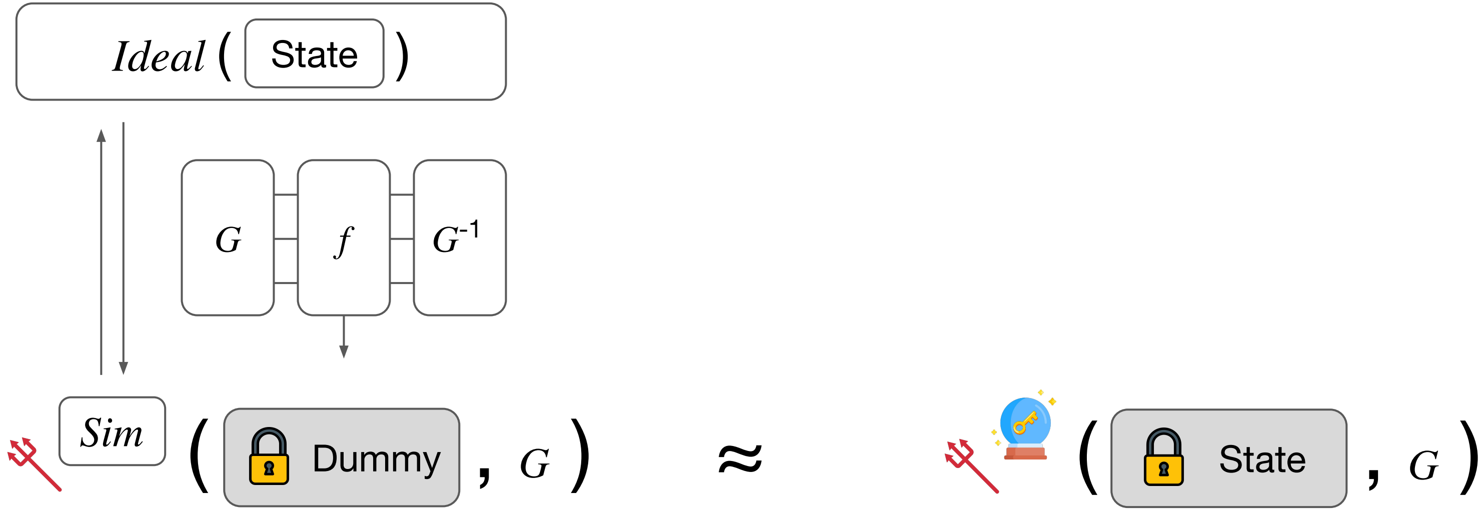
2. Functional Quantum Authentication Scheme

- We can view each oracle $= f \circ \mathrm{Dec}_{k,G}$ as a functional key
- Functional quantum authentication scheme consists of a setup and the following algorithms:



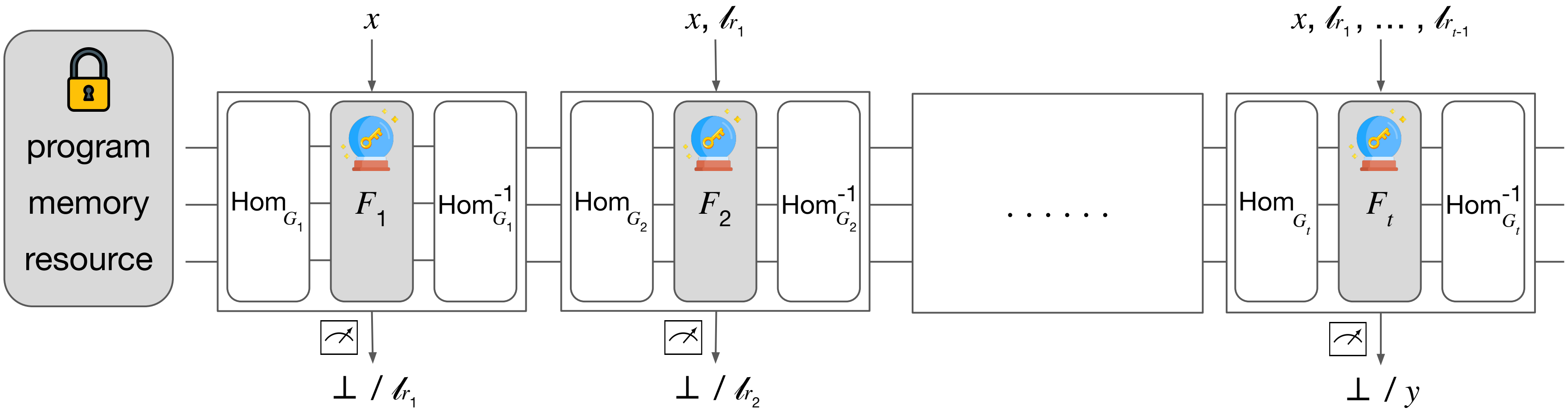
2. Functional Quantum Authentication Scheme

- We can view each oracle  $= f \circ \mathrm{Dec}_{k,G}$ as a functional key
- Functional quantum authentication scheme satisfies simulation security:



Obfuscation Scheme with Classical Inputs & Outputs

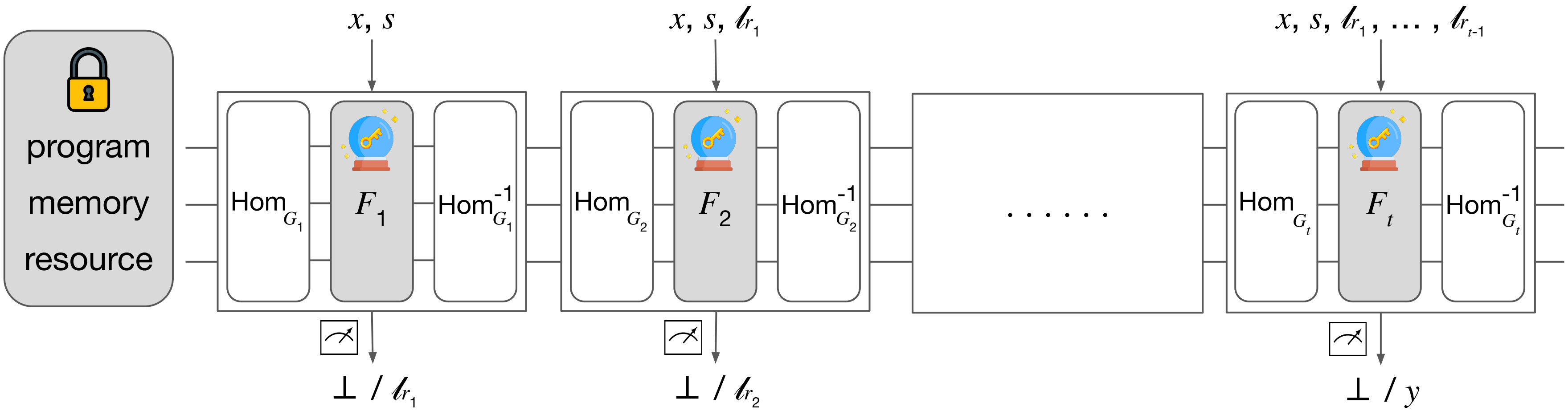
- The obfuscator creates an encrypted state along with classical oracles $F_1, \ldots, F_t$ as the obfuscated program



- Need to ensure that the input x is consistently used across oracle calls

Obfuscation Scheme with Classical Inputs & Outputs

- The obfuscator creates an encrypted state along with classical oracles $F_1, \ldots, F_t$ as the obfuscated program



- Use one-time tokenized signature [BS16] to ensure input consistency
- This is essentially the construction of [BBV24]

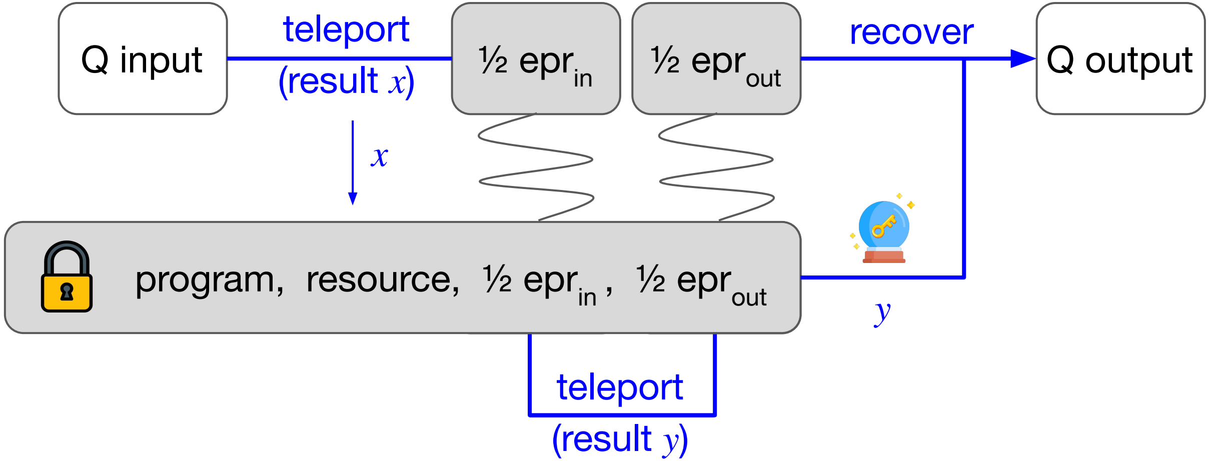
Supporting Quantum Inputs & Outputs

- Obstacles of the previous method
  - The model of quantum computation only deals with classical inputs & outputs
  - Several tasks were accomplished through **classical** oracles
    - Interpret the classical input
    - Check for input consistency across oracle calls
    - Deliver the classical output

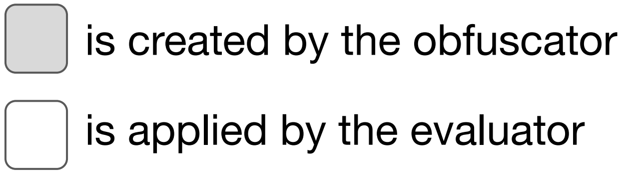
- How could the model of quantum computation support quantum inputs & outputs?
- How could classical oracles handle the same tasks for quantum inputs & outputs?

Solution for Supporting Quantum Inputs & Outputs

- We incorporate quantum teleportation into the framework

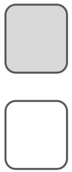


Our Obfuscation Scheme

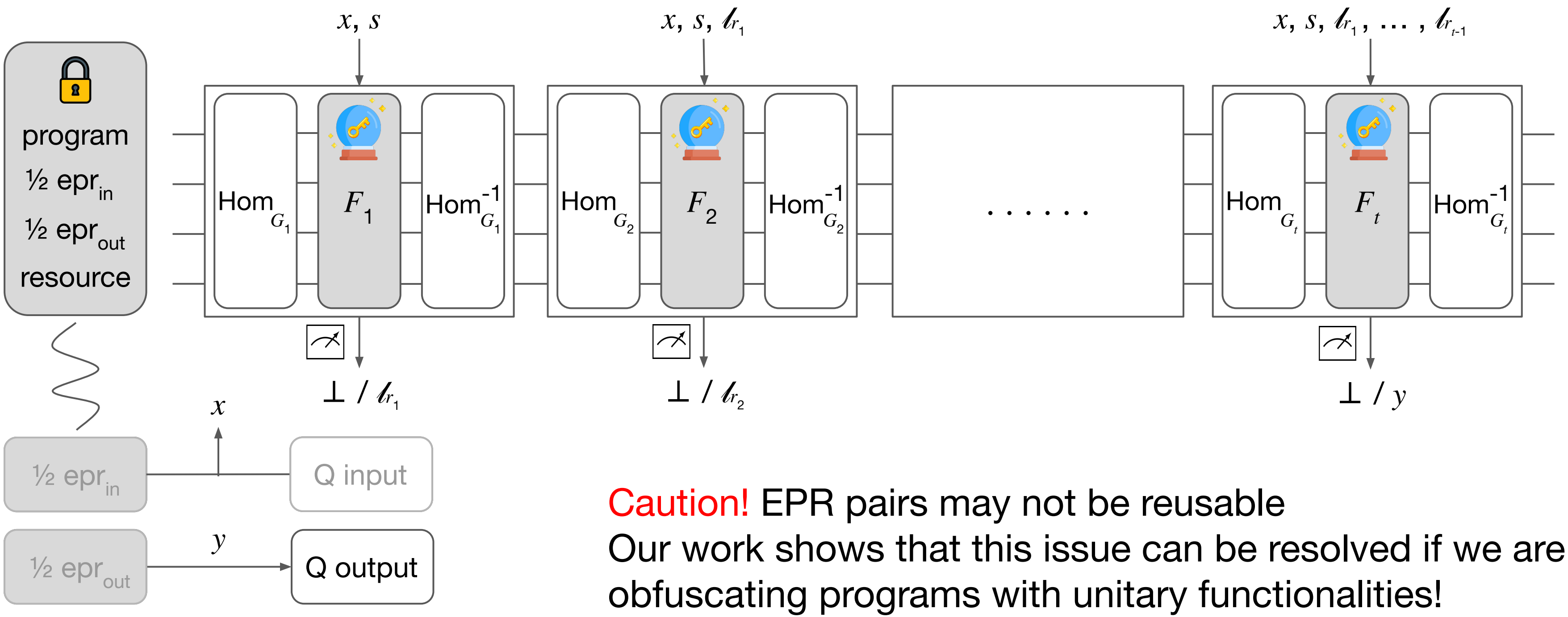


- The obfuscator produces the following obfuscated program

Our Obfuscation Scheme



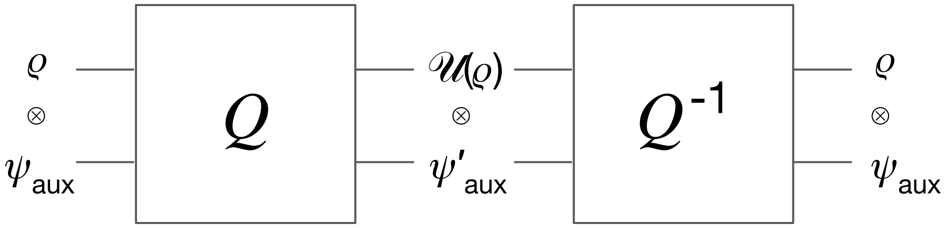
- Execution of the obfuscated program



Caution! EPR pairs may not be reusable
Our work shows that this issue can be resolved if we are obfuscating programs with unitary functionalities!

Addressing Reusability

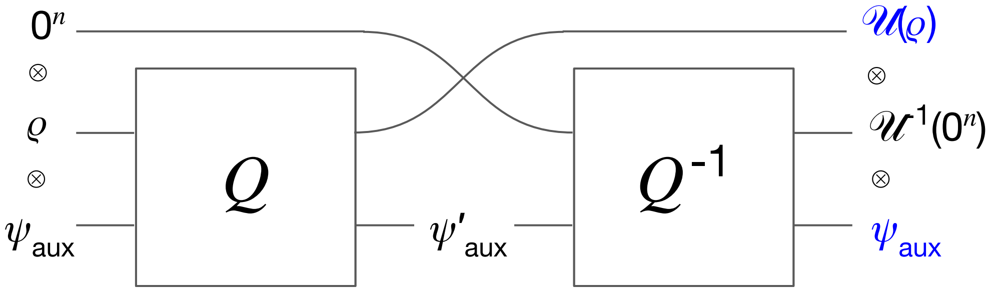
- $(\mathcal{Q}, \psi_{\text{aux}})$ implements a unitary functionality $\mathscr{U}$



- Crucially, the state ψ'_{aux} is independent of ϱ
- $(\mathcal{Q}^{-1}, \psi'_{\text{aux}})$ implements the unitary functionality $\mathscr{U}^{-1}$

Addressing Reusability

- $(\mathcal{Q}, \psi_{\text{aux}})$ implements a unitary functionality $\mathscr{U}$



- The state ψ_{aux} can be recovered after program execution
- One can evaluate $\mathscr{U}$ and $\mathscr{U}^{-1}$ multiple times

To Sum Up

For unitary functionalities

- We properly defined the notion of ideal obfuscation ✓
- We constructed ideal obfuscation for all quantum programs ✓
- Our obfuscated programs are reusable (for multiple evaluations) ✓

We also obtained

- A functional quantum authentication scheme with simulation security ✓

Open Problems & Future Directions

- Construct obfuscation schemes for
  - Isometry
  - Isometry + partial trace (which includes randomized classical functions)
- Obfuscate quantum circuits into quantum circuits (without auxiliary quantum states)
- Weaken the assumption to indistinguishability obfuscation

- Applications to quantum complexity theory
  - Instantiate quantum oracles from classical oracles plus quantum states

Thank you

Slides are prepared jointly by Miryam Huang and Er-Cheng Tang