Classical obfuscation for quantum circuits

Aparna Gupte
MIT

James Bartusek
Columbia

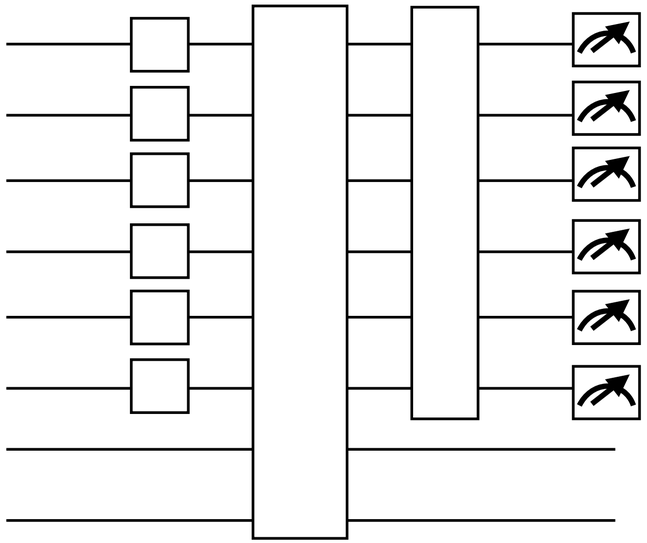
Saachi Mutreja
Columbia

Omri Shmueli
NTT

A mystery: in all previous work*, obfuscator must be quantum

* with the exception of obfuscation for null circuits.

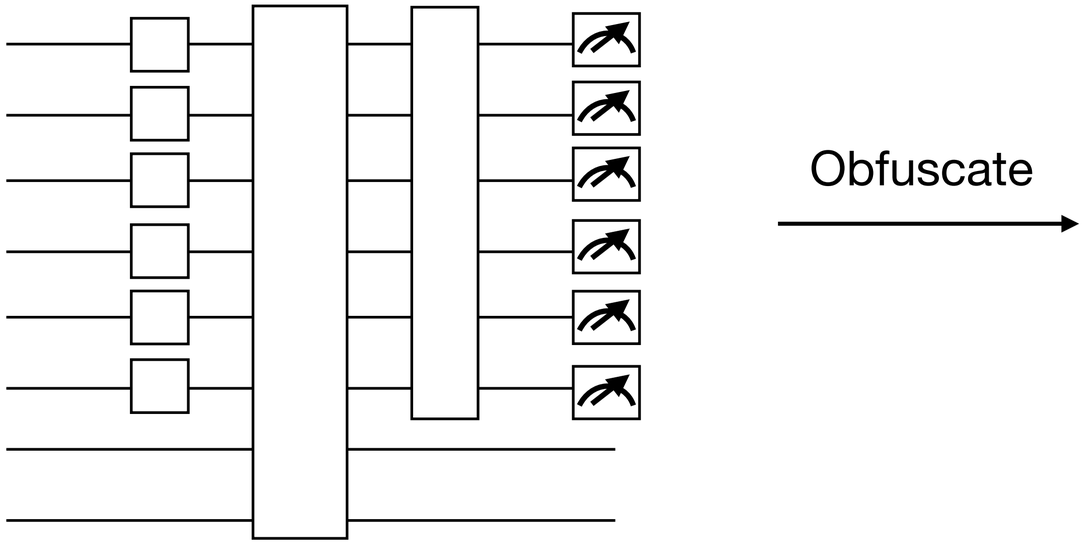
A mystery: in all previous work*, obfuscator must be quantum



Even if we start with a quantum circuit,
with classical description of gates

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum

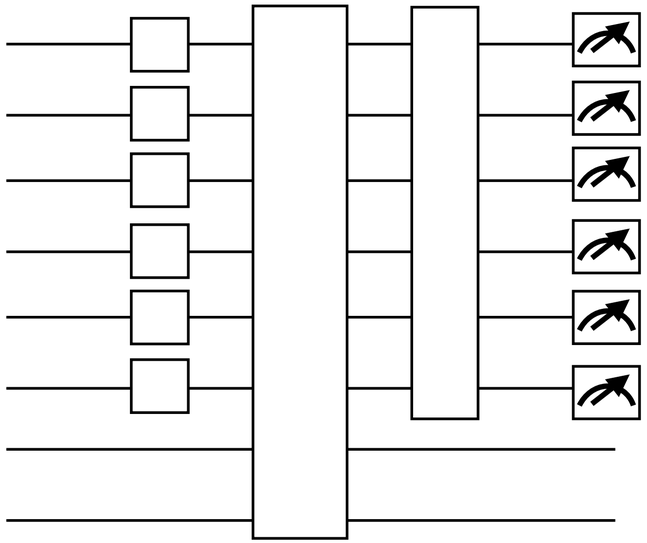


Obfuscate

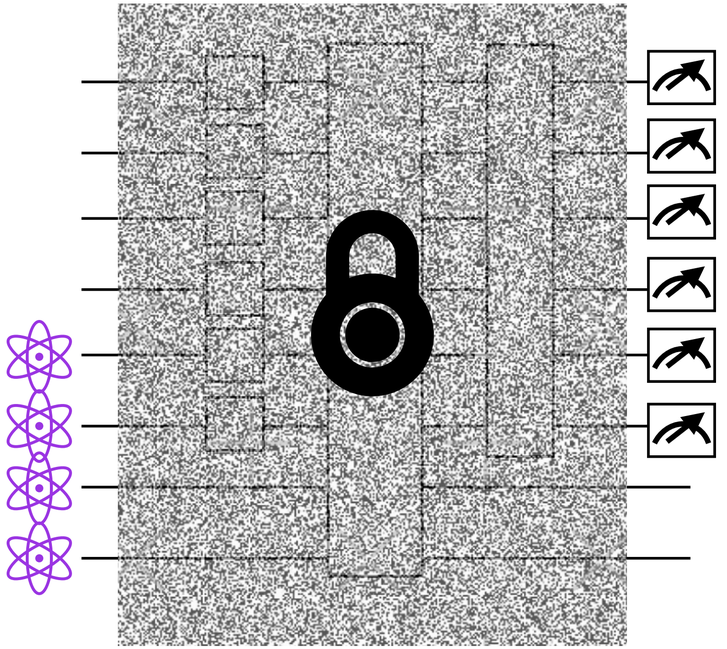
Even if we start with a quantum circuit,
with classical description of gates

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum



Obfuscate

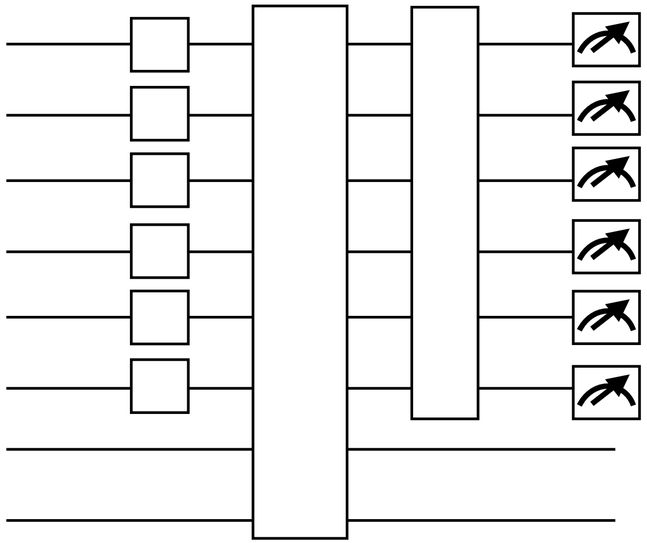


Even if we start with a quantum circuit,
with classical description of gates

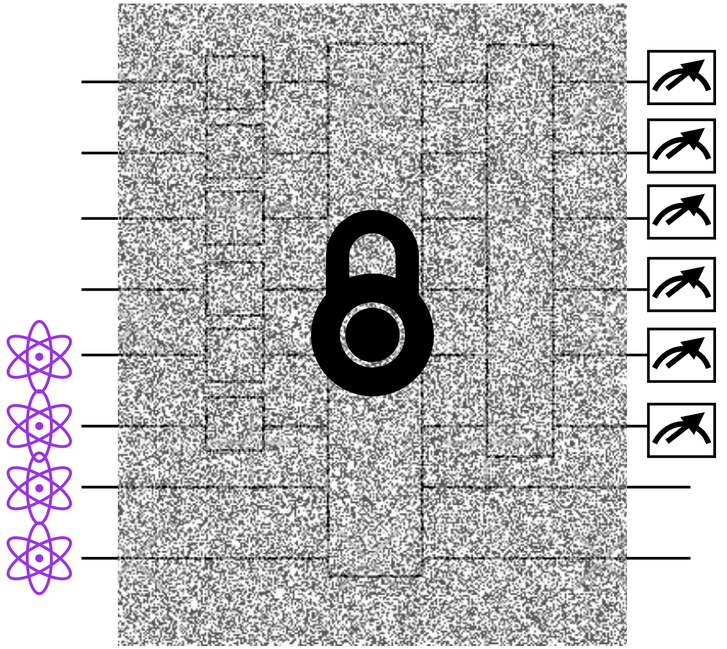
the obfuscation is a quantum program,
with auxiliary quantum state required to evaluate.

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum



Obfuscate



Even if we start with a quantum circuit,
with classical description of gates

the obfuscation is a quantum program,
with auxiliary quantum state required to evaluate.

Note: For this talk, I will not distinguish between “quantum obfuscator” and
“output of the obfuscator being quantum”.

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum

Pros

- Stronger security thanks to the quantum no-cloning theorem
  - copy-protection [CG23, Aar09]
  - one-time programs [BGS13, **G**LRRV24, GM24, **G**LQRRZ25]

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum

Pros

- Stronger security thanks to the quantum no-cloning theorem
  - copy-protection [CG23, Aar09]
  - one-time programs [BGS13, **G**LRRV24, GM24, **G**LQRRZ25]

Cons

- Need quantum networks to send obfuscated quantum programs

* with the exception of obfuscation for null circuits.

A mystery: in all previous work*, obfuscator must be quantum

Pros

- Stronger security thanks to the quantum no-cloning theorem
  - copy-protection [CG23, Aar09]
  - one-time programs [BGS13, **G**LRRV24, GM24, **G**LQRRZ25]

Cons

- Need quantum networks to send obfuscated quantum programs
- Heuristic proofs of quantumness (eg. from obfuscation of “peaked” quantum circuits) need classical verifier to perform obfuscation [AZ24]

* with the exception of obfuscation for null circuits.

Does obfuscation of quantum circuits require quantum obfuscator?

Does obfuscation of quantum circuits require quantum obfuscator?

Our main result:
No, at least for some quantum circuits.

Does obfuscation of quantum circuits require quantum obfuscator?

Our main result:
No, at least for some quantum circuits.

Theorem [B**G**MS]**.** Assuming LWE, there is a compiler from classical ideal obfuscation → ideal obfuscation for quantum circuits that compute classical functions.

*All these results are in the classical oracle model, and achieve ideal obfuscation

** Table omits BK’21 (circuits with logarithmic T-depth) and BM’22 (null circuits) for simplicity

Obfuscation for quantum circuits that implement:	**Classical functions** (e.g. Shor's algorithm)	**Unitary**	**General quantum maps**

*All these results are in the classical oracle model, and achieve ideal obfuscation

** Table omits BK'21 (circuits with logarithmic T-depth) and BM'22 (null circuits) for simplicity

Obfuscation for quantum circuits that implement:	**Classical functions** (e.g. Shor's algorithm)	**Unitary**	**General quantum maps**
Quantum obfuscator	BKNY'23 , BBV'24	HT'25	HT'26
Classical obfuscator	B**G**MS'26	?	?

*All these results are in the classical oracle model, and achieve ideal obfuscation

** Table omits BK'21 (circuits with logarithmic T-depth) and BM'22 (null circuits) for simplicity

Techniques

Techniques

Two distinct approaches to quantum obfuscation:

Techniques

Two distinct approaches to quantum obfuscation:

1. Homomorphic authentication scheme [BGS13, BBV24, HT25, HT26]

Techniques

Two distinct approaches to quantum obfuscation:

1. Homomorphic authentication scheme [BGS13, BBV24, HT25, HT26]
2. FHE + (public) verification of FHE evaluation [BKNY23]

Techniques

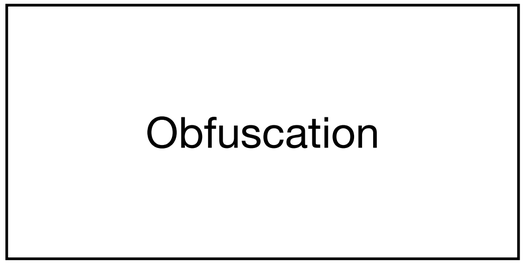
Two distinct approaches to quantum obfuscation:

1. Homomorphic authentication scheme [BGS13, BBV24, HT25, HT26]
2. FHE + (public) verification of FHE evaluation [BKNY23]

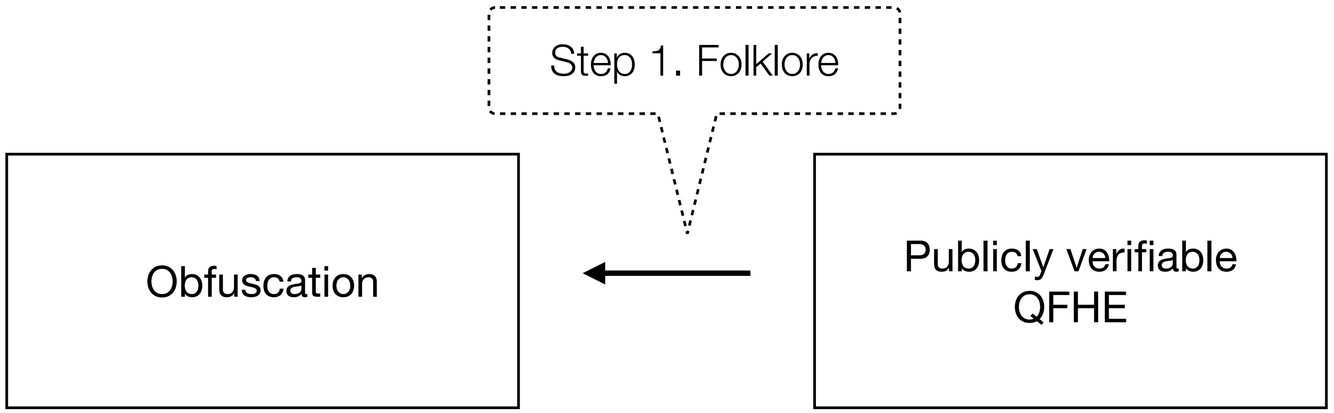
We take the second approach

Roadmap

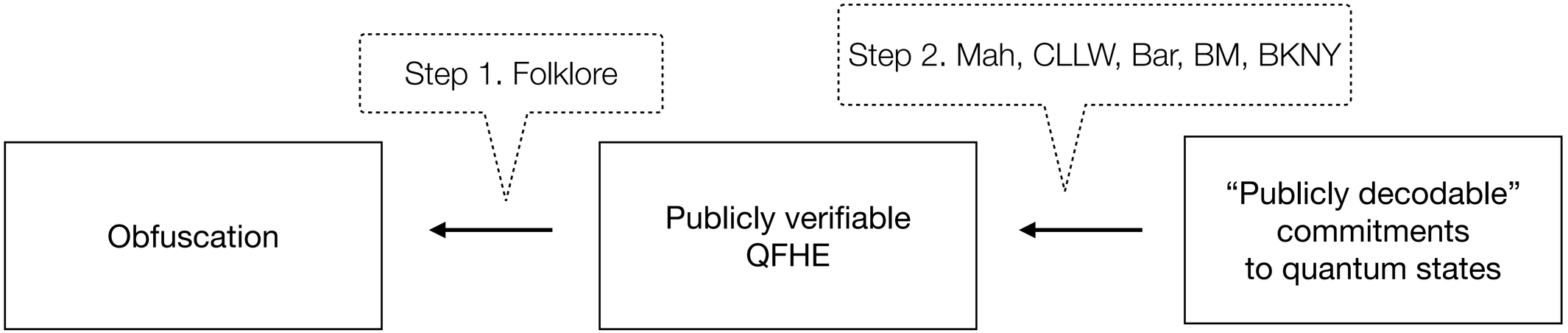
Roadmap



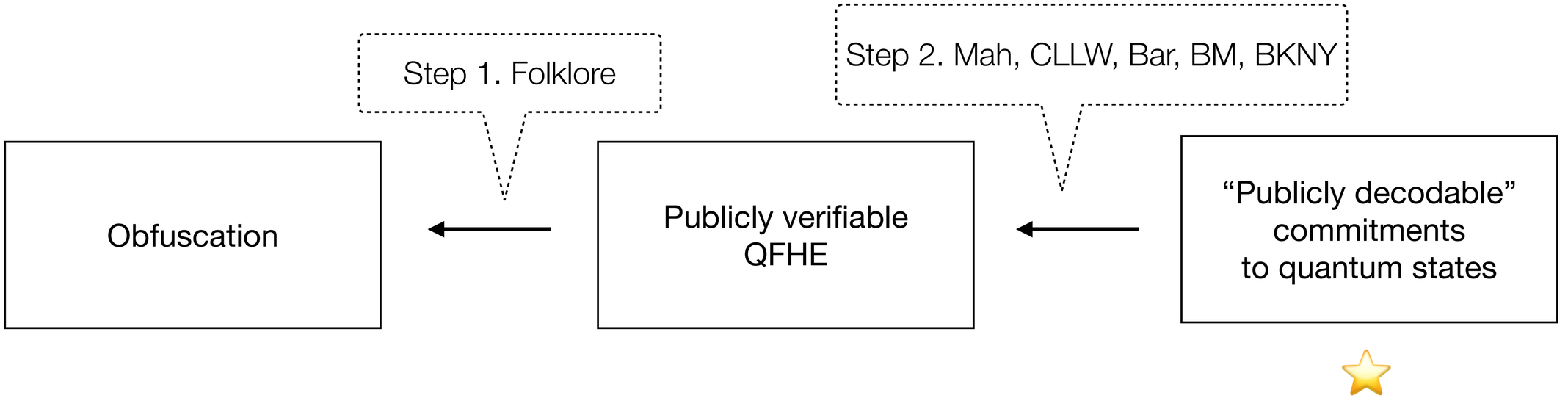
Roadmap



Roadmap

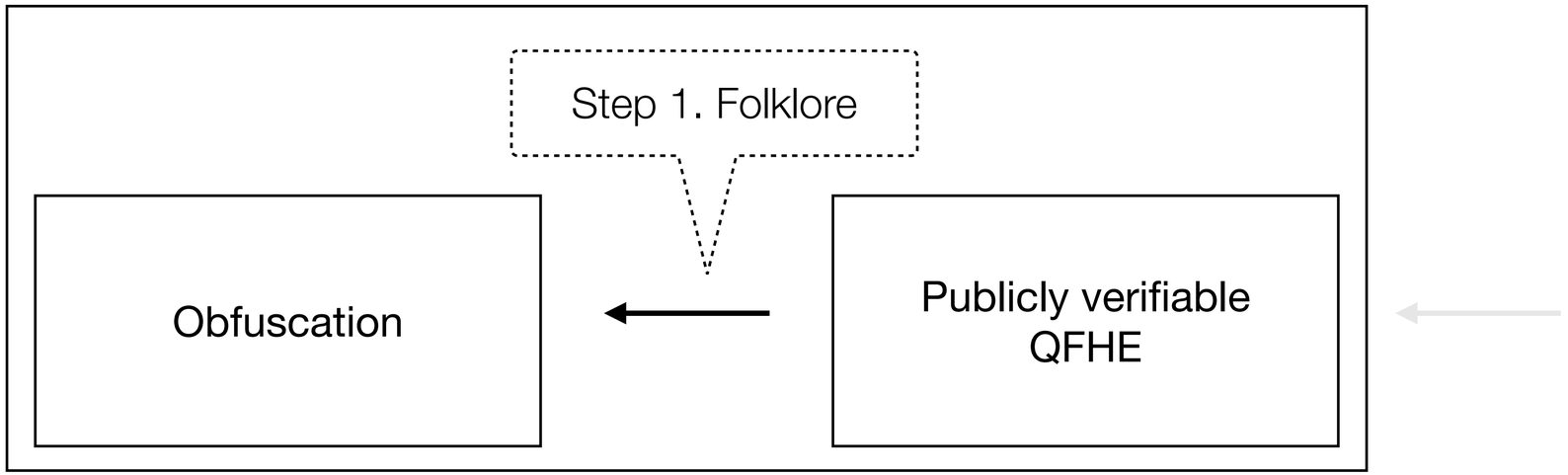

Step 1. Folklore
Step 2. Mah, CLLW, Bar, BM, BKNY
Obfuscation
Publicly verifiable
QFHE
“Publicly decodable”
commitments
to quantum states

Roadmap



- Roadmap laid out in BKNY, with **quantum** commitment keys.
- Our main technical contribution: constructing publicly decodable commitments to quantum states with **classical** commitment keys.

Roadmap



~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

Classical client

Quantum server

~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x

<u>Quantum server</u>

- Public quantum circuit Q

~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x
- $\mathsf{Enc}(\mathsf{pk}, x) \rightarrow \mathsf{ct}$

<u>Quantum server</u>

- Public quantum circuit Q

~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x
- Enc(pk, x) $\rightarrow$ ct

ct $\longrightarrow$

<u>Quantum server</u>

- Public quantum circuit Q

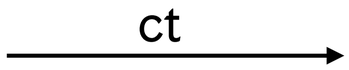
~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x
- $\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$



<u>Quantum server</u>

- Public quantum circuit Q
- $\mathsf{Eval}(\mathsf{evk}, \mathsf{ct}, Q) \to \widetilde{\mathsf{ct}}$

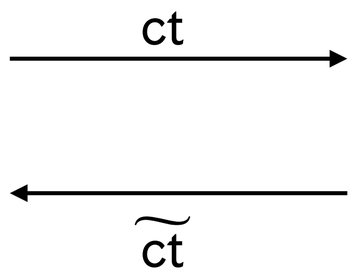
~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x
- $\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$



<u>Quantum server</u>

- Public quantum circuit Q
- $\mathsf{Eval}(\mathsf{evk}, \mathsf{ct}, Q) \to \widetilde{\mathsf{ct}}$

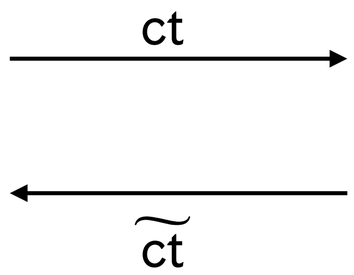
~~Publicly verifiable~~ Quantum FHE

Classical client can delegate quantum computation while keeping inputs private

LWE $\Longrightarrow$ QFHE [Mah18]

<u>Classical client</u>

- Input x
- Enc(pk, x) $\rightarrow$ ct
- Dec(sk, ct) $\rightarrow$ $Q(x)$



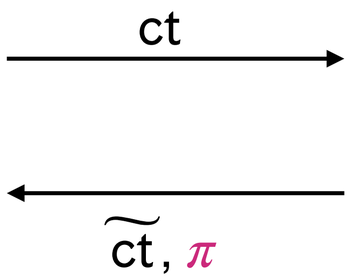
<u>Quantum server</u>

- Public quantum circuit Q
- Eval(evk, ct, Q) $\rightarrow$ $\widetilde{\text{ct}}$

Publicly verifiable Quantum FHE

Classical client

- Input x
- Enc(pk, x) $\rightarrow$ ct
- Dec(sk, ct) $\rightarrow$ $Q(x)$



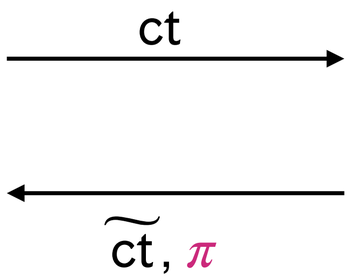
Quantum server

- Public quantum circuit Q
- Eval(evk, ct, Q) $\rightarrow$ $\widetilde{\text{ct}}$, π

Publicly verifiable Quantum FHE

Classical client

- Input x
- $\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$



- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to Q(x)$

Quantum server

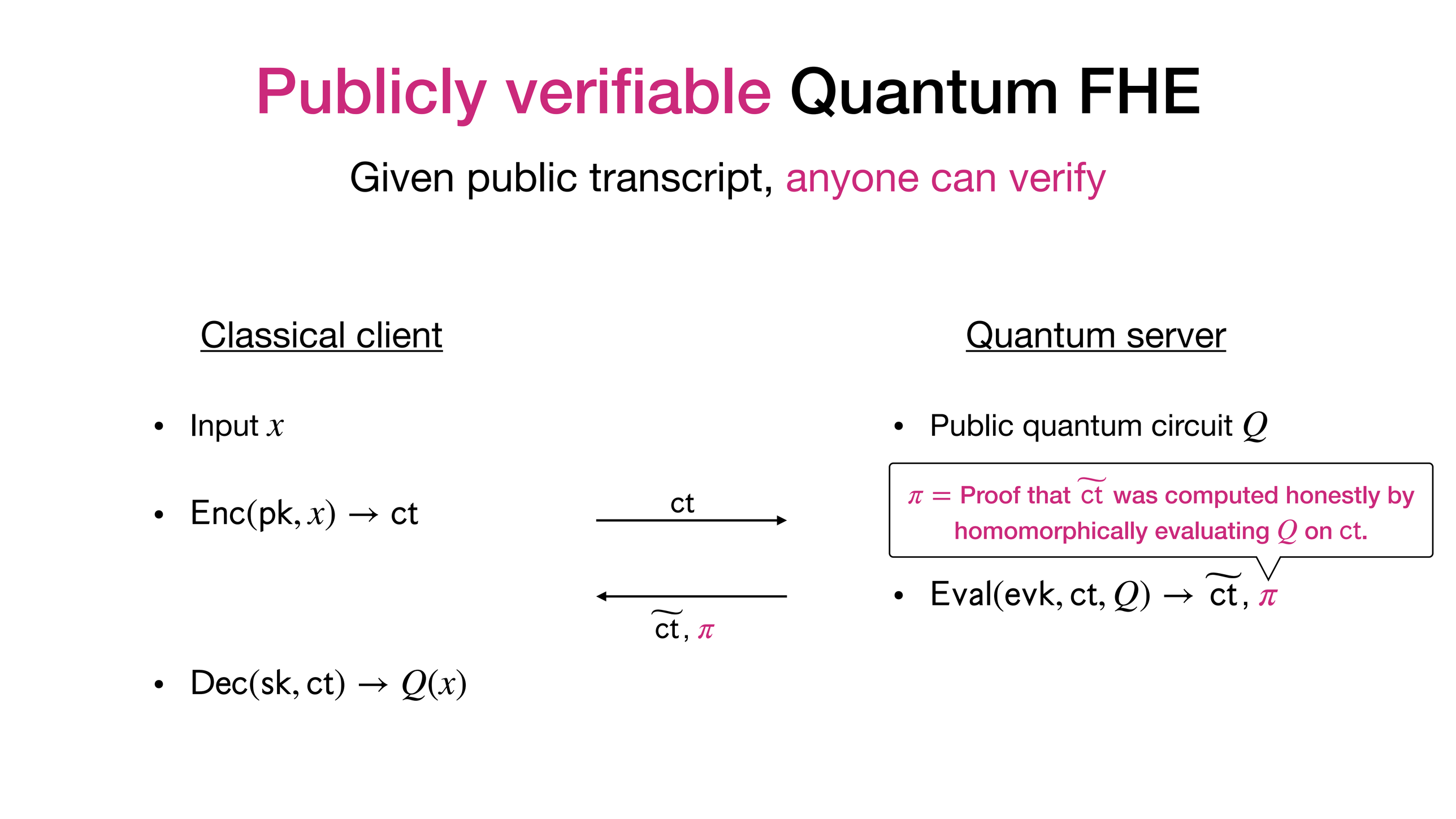
- Public quantum circuit Q

π = Proof that $\widetilde{\mathsf{ct}}$ was computed honestly by homomorphically evaluating Q on ct.

- $\mathsf{Eval}(\mathsf{evk}, \mathsf{ct}, Q) \to \widetilde{\mathsf{ct}}, \pi$

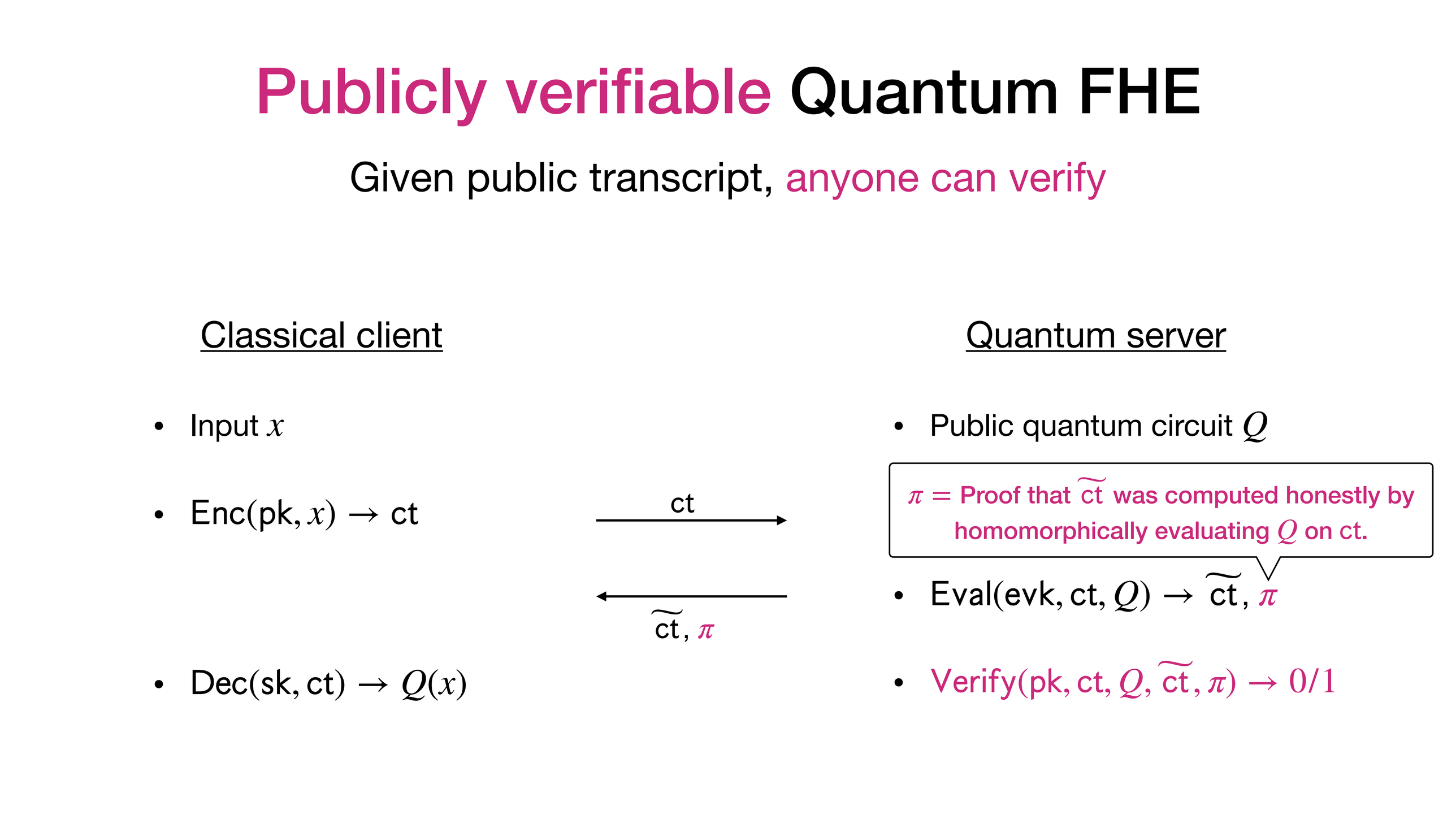
Publicly verifiable Quantum FHE

Given public transcript, anyone can verify



Publicly verifiable Quantum FHE

Given public transcript, anyone can verify



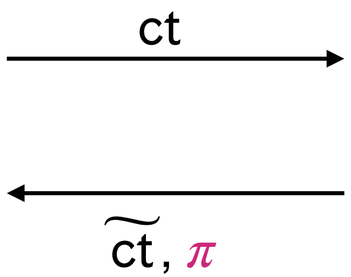
Publicly verifiable Quantum FHE

Given public transcript, anyone can verify

Remains sound even if malicious quantum server is given oracle access to Verify

Classical client

- Input x
- $\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to Q(x)$



Quantum server

- Public quantum circuit Q

π = Proof that $\widetilde{\mathsf{ct}}$ was computed honestly by homomorphically evaluating Q on ct.

- $\mathsf{Eval}(\mathsf{evk}, \mathsf{ct}, Q) \to \widetilde{\mathsf{ct}}, \pi$
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{ct}, Q, \widetilde{\mathsf{ct}}, \pi) \to 0/1$

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct

Evaluate($\hat{Q}, x$):

1. Eval(evk, Univ$_x$, ct) $\rightarrow$ $\widetilde{\text{ct}}, \pi$

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct
2. Define oracle Verify-Decrypt(x, $\widetilde{\text{ct}}$, π):
   If Verify(pk, ct, Univ_x, $\widetilde{\text{ct}}$, π) $= 1$, output Dec(sk, $\widetilde{\text{ct}}$)

Evaluate($\hat{Q}$, x):

1. Eval(evk, Univ_x, ct) $\rightarrow$ $\widetilde{\text{ct}}$, π

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct
2. Define oracle Verify-Decrypt(x, $\widetilde{\text{ct}}$, π):
   If Verify(pk, ct, Univ$_x$, $\widetilde{\text{ct}}$, π) $= 1$, output Dec(sk, $\widetilde{\text{ct}}$)
3. Output $\hat{Q}$ = (ct, Verify-Decrypt)

Classical ideal obfuscation (aka classical oracles) is free

Evaluate($\hat{Q}$, x):

1. Eval(evk, Univ$_x$, ct) $\rightarrow$ $\widetilde{\text{ct}}$, π

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct
2. Define oracle Verify-Decrypt(x, $\widetilde{\text{ct}}$, π):
   If Verify(pk, ct, Univ$_x$, $\widetilde{\text{ct}}$, π) $= 1$, output Dec(sk, $\widetilde{\text{ct}}$)
3. Output $\hat{Q}$ = (ct, Verify-Decrypt)

Classical ideal obfuscation (aka classical oracles) is free

Evaluate($\hat{Q}$, x):

1. Eval(evk, Univ$_x$, ct) $\rightarrow$ $\widetilde{\text{ct}}$, π
2. Run Verify-Decrypt(x, $\widetilde{\text{ct}}$, π) $\rightarrow$ $Q(x)$

Publicly verifiable QFHE $\Longrightarrow$ Obfuscation

Verify-then-decrypt

Obfuscate(Q):

1. Enc(pk, Q) $\rightarrow$ ct
2. Define oracle Verify-Decrypt(x, $\widetilde{\text{ct}}$, π):
   If Verify(pk, ct, Univ$_x$, $\widetilde{\text{ct}}$, π) $= 1$, output Dec(sk, $\widetilde{\text{ct}}$)
3. Output $\hat{Q}$ = (ct, Verify-Decrypt)

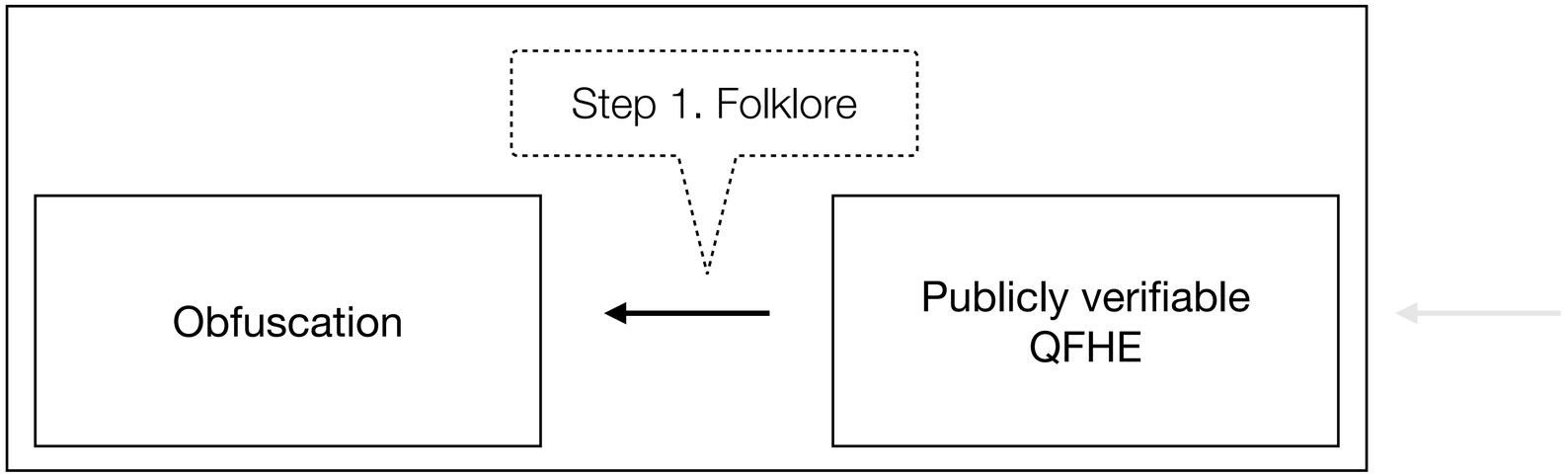
Evaluate($\hat{Q}$, x):

1. Eval(evk, Univ$_x$, ct) $\rightarrow$ $\widetilde{\text{ct}}$, π
2. Run Verify-Decrypt(x, $\widetilde{\text{ct}}$, π) $\rightarrow$ $Q(x)$

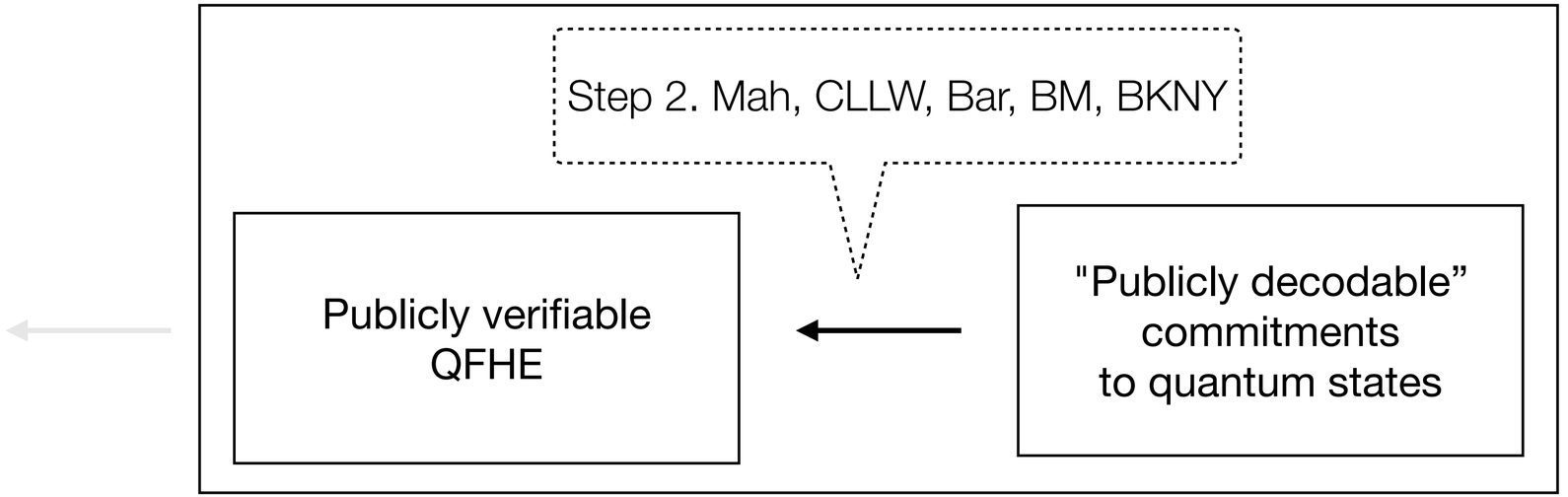
Why public verifiability?

- **If not, could find bad proof π^* after repeated queries.**
- **This can leak illegal information about Q, e.g. Verify-Decrypt(x, ct, π^*) reveals Q if π^* is accepted.**

Roadmap



Roadmap


Step 2. Mah, CLLW, Bar, BM, BKNY
Publicly verifiable
QFHE
"Publicly decodable”
commitments
to quantum states

Constructing Publicly Verifiable QFHE

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG
- Standard approach to **verifiable delegated quantum computation**

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

- Standard approach to **verifiable delegated quantum computation**

    1. Quantum server/prover prepares quantum “history state” (aka trace/tableau of the computation)

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

- Standard approach to **verifiable delegated quantum computation**

    1. Quantum server/prover prepares quantum “history state” (aka trace/tableau of the computation)

    2. This is the “ground state” of the “Feynman-Kitaev Hamiltonian” (aka satisfying assignment for Cook-Levin CSP)

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

- Standard approach to **verifiable delegated quantum computation**

    1. Quantum server/prover prepares quantum “history state” (aka trace/tableau of the computation)

    2. This is the “ground state” of the “Feynman-Kitaev Hamiltonian” (aka satisfying assignment for Cook-Levin CSP)

    3. To verify correctness of computation, measure qubits in standard and Hadamard bases (aka checking that CSP clauses are satisfied)

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

- Standard approach to **verifiable delegated quantum computation**

    1. Quantum server/prover prepares quantum “history state” (aka trace/tableau of the computation)

    2. This is the “ground state” of the “Feynman-Kitaev Hamiltonian” (aka satisfying assignment for Cook-Levin CSP)

    3. To verify correctness of computation, measure qubits in standard and Hadamard bases (aka checking that CSP clauses are satisfied)

    4. Need a way to force/bind the quantum server to “measure state honestly”

Constructing Publicly Verifiable QFHE

- Publicly verifiable *classical* FHE is “straightforward”: FHE + SNARG

- Standard approach to **verifiable delegated quantum computation**

    1. Quantum server/prover prepares quantum “history state” (aka trace/tableau of the computation)
    2. This is the “ground state” of the “Feynman-Kitaev Hamiltonian” (aka satisfying assignment for Cook-Levin CSP)
    3. To verify correctness of computation, measure qubits in standard and Hadamard bases (aka checking that CSP clauses are satisfied)
    4. Need a way to force/bind the quantum server to “measure state honestly”
    5. This is what commitments to quantum states enable!

Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

Classical verifier

Quantum committer

- Quantum state $|\psi\rangle$

Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

<u>Classical verifier</u>

- Gen $\rightarrow$ ck, dk

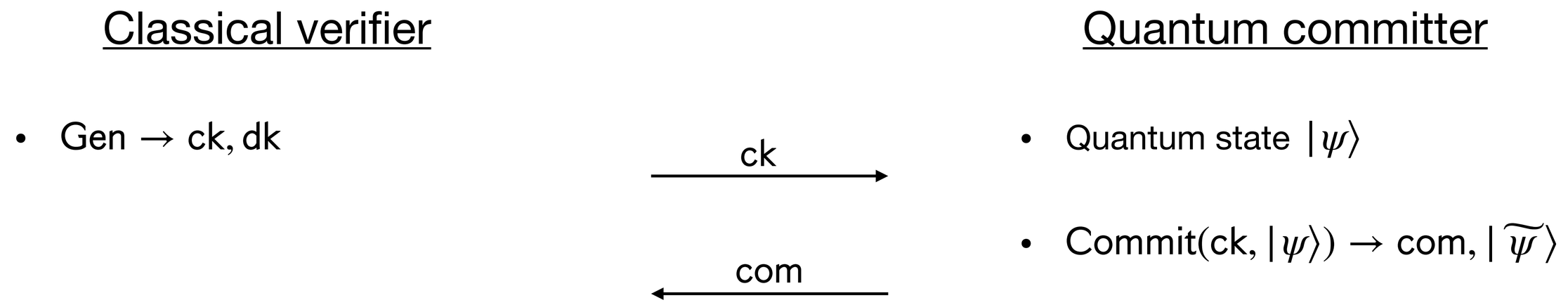
ck $\longrightarrow$

<u>Quantum committer</u>

- Quantum state $|\psi\rangle$

Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”



Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

Classical verifier

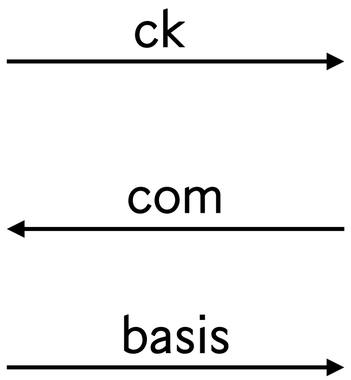
- Gen $\rightarrow$ ck, dk

- basis $\in \{Z, X\}$

Quantum committer

- Quantum state $|\psi\rangle$

- Commit(ck, $|\psi\rangle$) $\rightarrow$ com, $|\widetilde{\psi}\rangle$



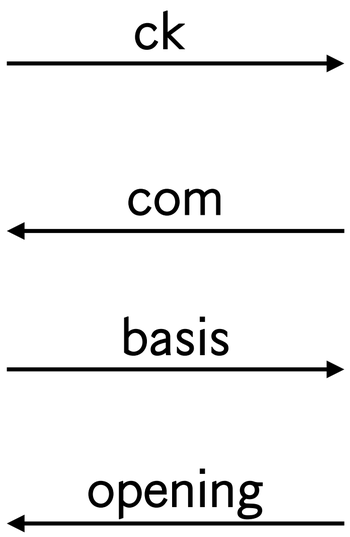
Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

Classical verifier

- Gen $\rightarrow$ ck, dk

- basis $\in \{Z, X\}$



Quantum committer

- Quantum state $|\psi\rangle$

- Commit(ck, $|\psi\rangle$) $\rightarrow$ com, $|\widetilde{\psi}\rangle$

- Open($|\tilde{\psi}\rangle$, basis) $\rightarrow$ opening

Commitments to quantum states

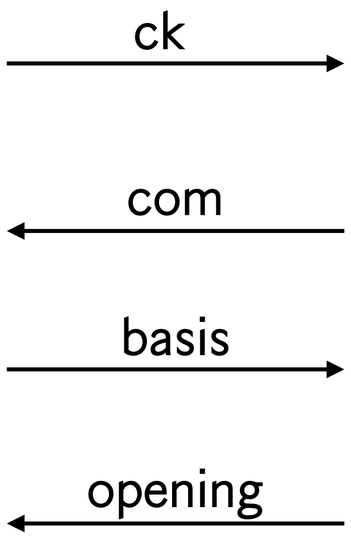
Force malicious quantum committer to “measure its state honestly”

Classical verifier

- Gen $\rightarrow$ ck, dk

- basis $\in \{Z, X\}$

- Decode(dk, com, opening)
  $\rightarrow$ measurement outcome



Quantum committer

- Quantum state $|\psi\rangle$

- Commit(ck, $|\psi\rangle$) $\rightarrow$ com, $|\widetilde{\psi}\rangle$

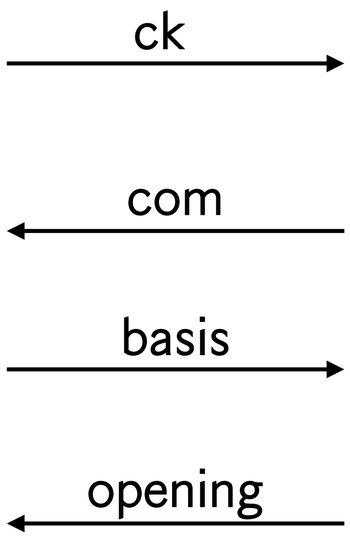
- Open($|\tilde{\psi}\rangle$, basis) $\rightarrow$ opening

Commitments to quantum states

Force malicious quantum committer to “measure its state honestly”

Classical verifier

- Gen $\rightarrow$ ck, dk
- basis $\in \{Z, X\}$
- Decode(dk, com, opening) $\rightarrow$ measurement outcome



Quantum committer

- Quantum state $|\psi\rangle$
- Commit(ck, $|\psi\rangle$) $\rightarrow$ com, $|\widetilde{\psi}\rangle$
- Open($|\tilde{\psi}\rangle$, basis) $\rightarrow$ opening

In standard constructions, committer does not know what it’s opening to!!!

Classical verifier uses decoding key dk to map opening to a measurement outcome

Constructing Publicly Verifiable QFHE

Standard approach to **verifiable delegated quantum computation**

1. Quantum server/prover commits “history state”, and is forced to measure honestly.

2. Verifier can check measurement outcomes correspond to “low energy” state (aka satisfying assignment)

Constructing Publicly Verifiable QFHE

Standard approach to **verifiable delegated quantum computation**

1. Quantum server/prover commits “history state”, and is forced to measure honestly.

2. Verifier can check measurement outcomes correspond to “low energy” state (aka satisfying assignment)

Public verifiability (quantum server has verification oracle)

Constructing Publicly Verifiable QFHE

Standard approach to **verifiable delegated quantum computation**

1. Quantum server/prover commits “history state”, and is forced to measure honestly.

2. Verifier can check measurement outcomes correspond to “low energy” state (aka satisfying assignment)

Public verifiability (quantum server has verification oracle)

$\Longrightarrow$ Server effectively sees measurement outcomes

Constructing Publicly Verifiable QFHE

Standard approach to **verifiable delegated quantum computation**

1. Quantum server/prover commits “history state”, and is forced to measure honestly.

2. Verifier can check measurement outcomes correspond to “low energy” state (aka satisfying assignment)

Public verifiability (quantum server has verification oracle)

$\Longrightarrow$ Server effectively sees measurement outcomes

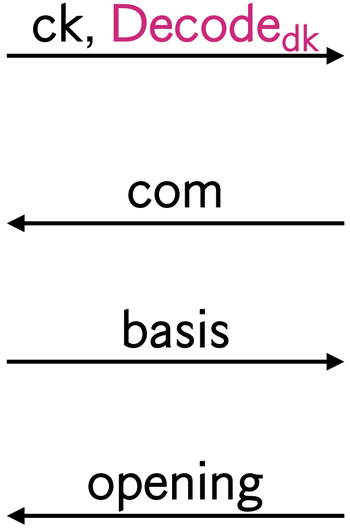
$\Longrightarrow$ Need “publicly decodable” commitment to quantum states

Publicly decodable commitments

Remains sound even if malicious quantum server knows what it's opening to
(i.e. is given oracle access to Decode_{dk})

Classical verifier

- Gen → ck, dk
- basis ∈ {Z, X}
- Decode(dk, com, basis, opening)
  → measurement outcome



Quantum committer

- Quantum state $|\psi\rangle$
- Commit(ck, $|\psi\rangle$) → com, $|\widetilde{\psi}\rangle$
- Open($|\tilde{\psi}\rangle$, basis) → opening

Publicly decodable commitments

Remains sound even if malicious quantum server knows what it's opening to
(i.e. is given oracle access to Decode_{dk})

Classical verifier

- Gen → ck, dk

- basis ∈ {Z, X}

- Decode(dk, com, basis, opening)
  → measurement outcome

ck, Decode_{dk} →

← com

basis →

← opening

Quantum committer

- Quantum state $|\psi\rangle$

- Commit(ck, $|\psi\rangle$) → com, $|\widetilde{\psi}\rangle$

- Open($|\tilde{\psi}\rangle$, basis) → opening

Impossible :(

Weakly publicly decodable commitments suffice [BKNY]

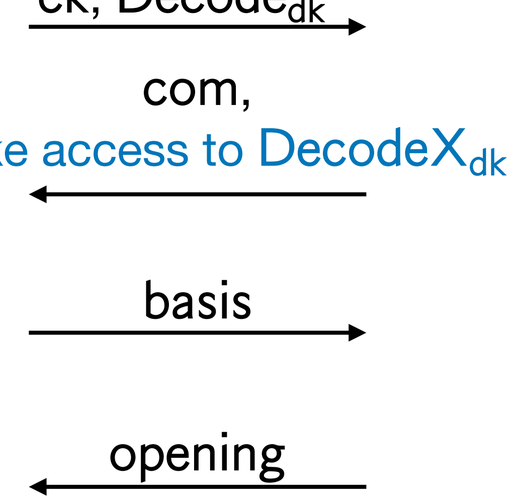
Remains sound even if malicious quantum server is given oracle access to Decode_{dk},
but access to DecodeX_{dk} is revoked after commitment

Classical verifier

- Gen $\rightarrow$ ck, dk

- basis $\in \{Z, X\}$

- Decode(dk, com, basis, opening)
  $\rightarrow$ measurement outcome



Quantum committer

- Quantum state $|\psi\rangle$

- Commit(ck, $|\psi\rangle$) $\rightarrow$ com, $|\widetilde{\psi}\rangle$

- Open($|\tilde{\psi}\rangle$, basis) $\rightarrow$ opening

BKNY commitments

$$\text{Commit}\big(\alpha\,|0\rangle + \beta\,|1\rangle\big) \longrightarrow \alpha\,|0\rangle\,|S\rangle + \beta\,|1\rangle\,|S+v\rangle$$

BKNY commitments

$$\text{Commit}\left(\alpha\,|0\rangle + \beta\,|1\rangle\right) \longrightarrow \alpha\,|0\rangle\,|S\rangle + \beta\,|1\rangle\,|S+v\rangle$$

- Quantum commitment key $|S\rangle = \sum_{s\in S} |s\rangle$

BKNY commitments

$$\text{Commit}\big(\alpha\,|0\rangle + \beta\,|1\rangle\big) \longrightarrow \alpha\,|0\rangle\,|S\rangle + \beta\,|1\rangle\,|S+v\rangle$$

- Quantum commitment key $|S\rangle = \sum_{s\in S} |s\rangle$

- Random $n/2$-dimensional subspace S
- DecodeX $= S^{\perp}$

BKNY commitments

$$\text{Commit}\left(\alpha\,|0\rangle + \beta\,|1\rangle\right) \longrightarrow \alpha\,|0\rangle\,|S\rangle + \beta\,|1\rangle\,|S+v\rangle$$

- Quantum commitment key $|S\rangle = \sum_{s\in S} |s\rangle$

- To get classical commitment keys, need a way for committer to prepare $|S\rangle$ without learning S

- Random $n/2$-dimensional subspace S
- DecodeX $= S^{\perp}$

BKNY commitments

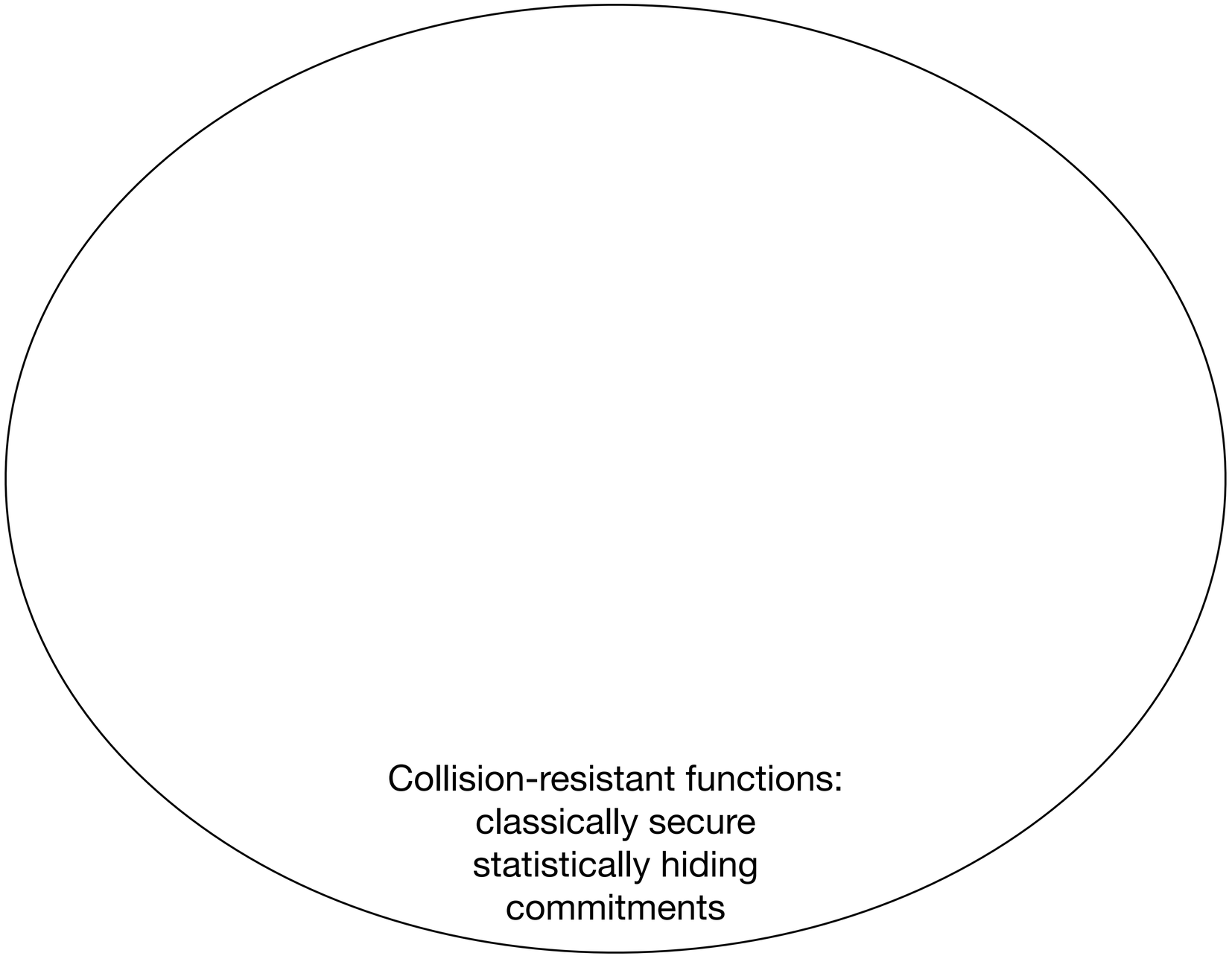
$$\text{Commit}\big(\alpha\,|0\rangle + \beta\,|1\rangle\big) \longrightarrow \alpha\,|0\rangle\,|S\rangle + \beta\,|1\rangle\,|S+v\rangle$$

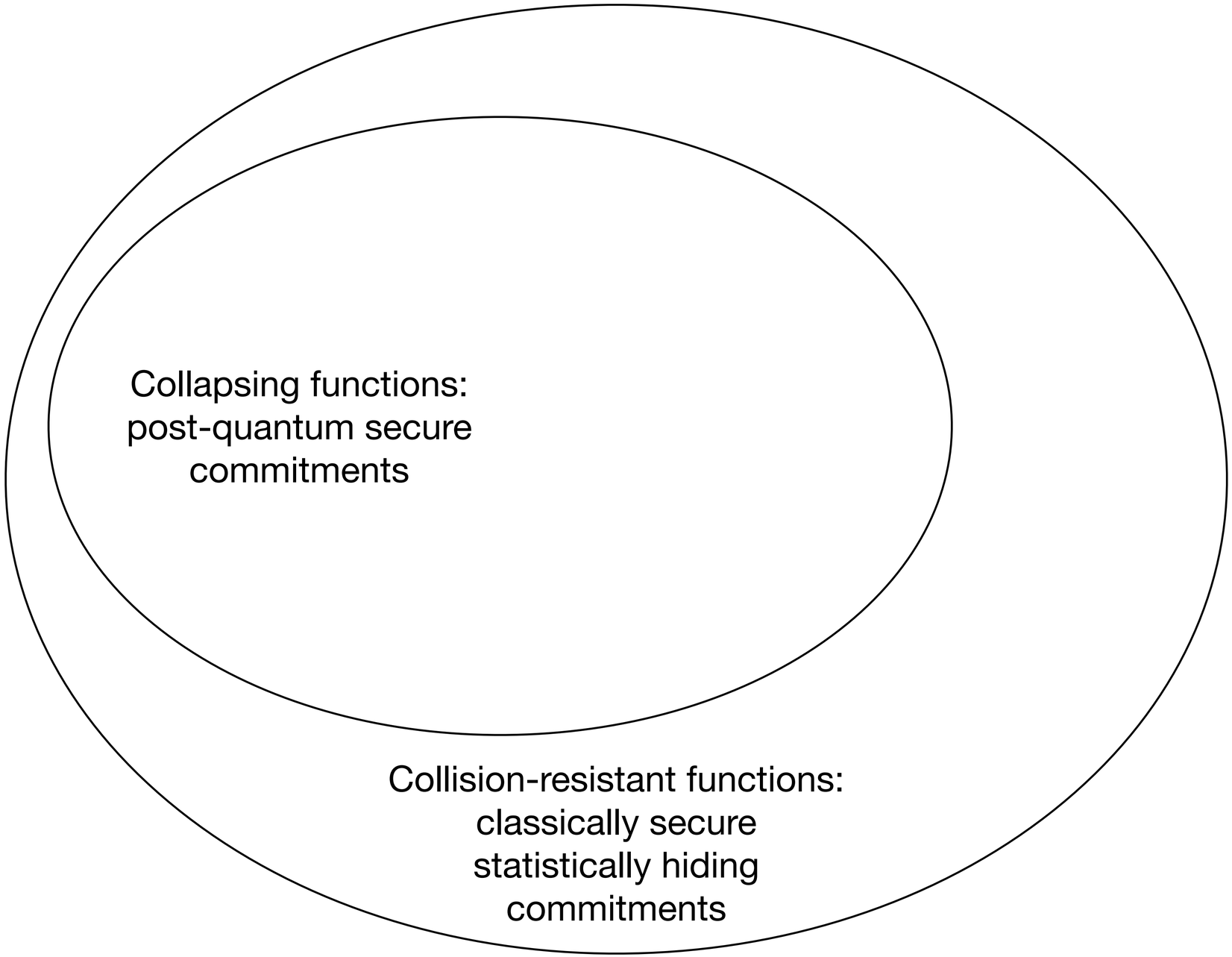
- Quantum commitment key $|S\rangle = \sum_{s\in S} |s\rangle$

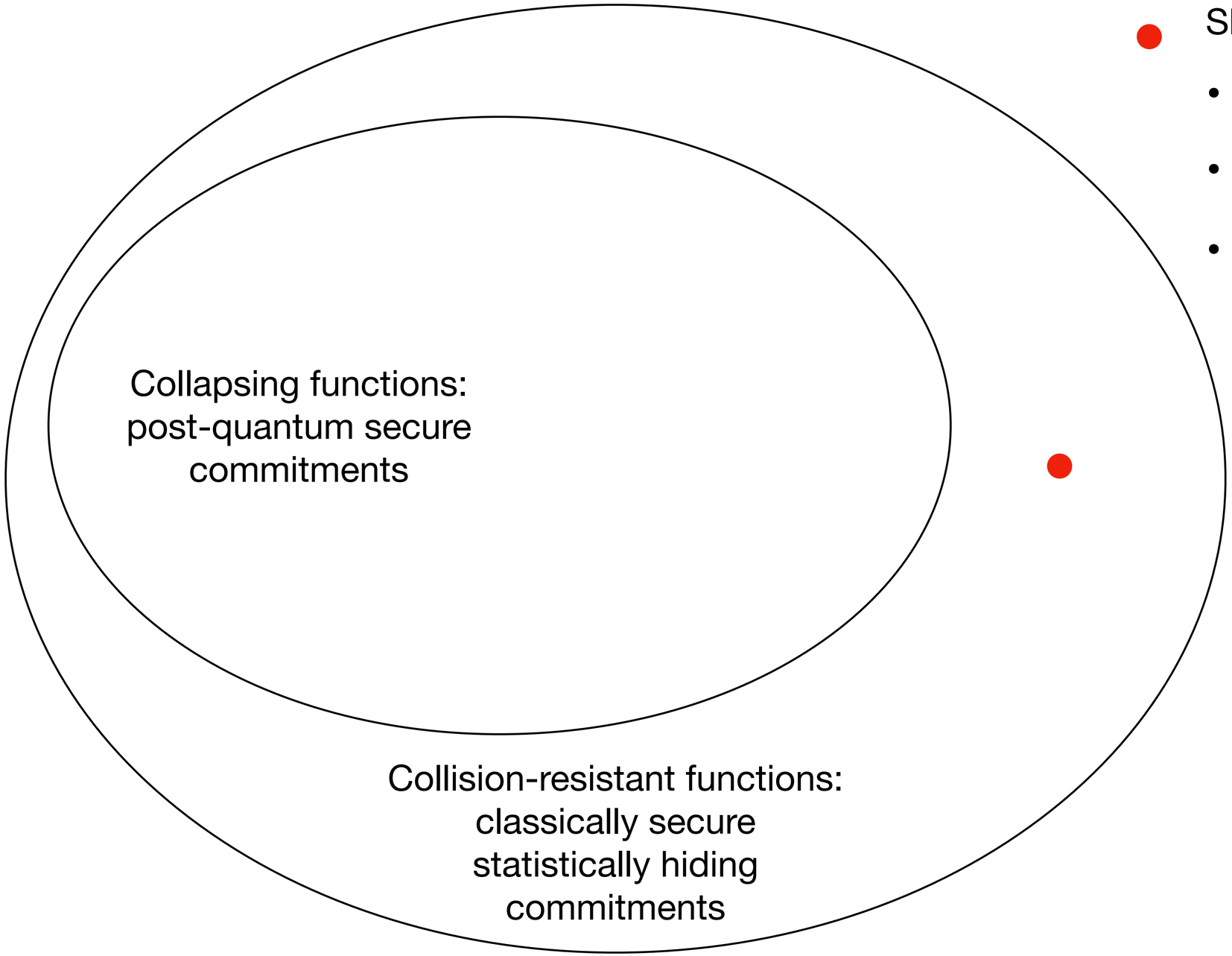
- To get classical commitment keys, need a way for committer to prepare $|S\rangle$ without learning S

- Shmueli-Zhandry function suggests a way to do this, but this gives an quantum-insecure commitment scheme!

- Random $n/2$-dimensional subspace S
- DecodeX $= S^{\perp}$

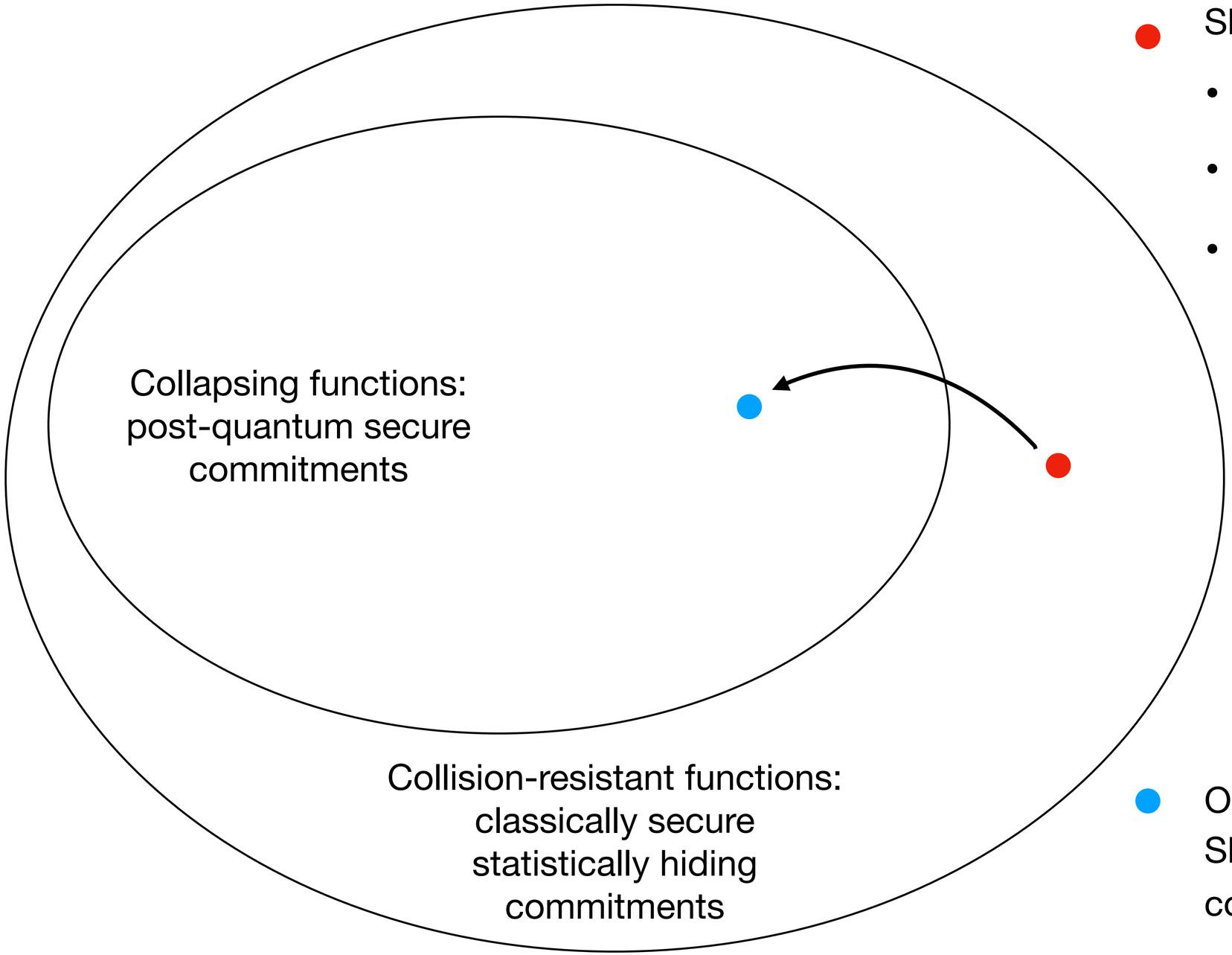

Collision-resistant functions:
classically secure
statistically hiding
commitments


Collapsing functions:
post-quantum secure
commitments
Collision-resistant functions:
classically secure
statistically hiding
commitments



Shmueli-Zhandry ’25

- Collision-resistant function f
- Preimage of $y \leftrightarrow$ subspaces S_y
- Oracle $S_y^{\perp}$ makes it non-collapsing, but preserves collision-resistance



- Shmueli-Zhandry ’25
  - Collision-resistant function f
  - Preimage of $y \leftrightarrow$ subspaces S_y
  - Oracle $S_y^{\perp}$ makes it non-collapsing, but preserves collision-resistance

- Our main contribution:
  Show that revoking $S_y^{\perp}$ makes f collapsing (aka a secure commitment)

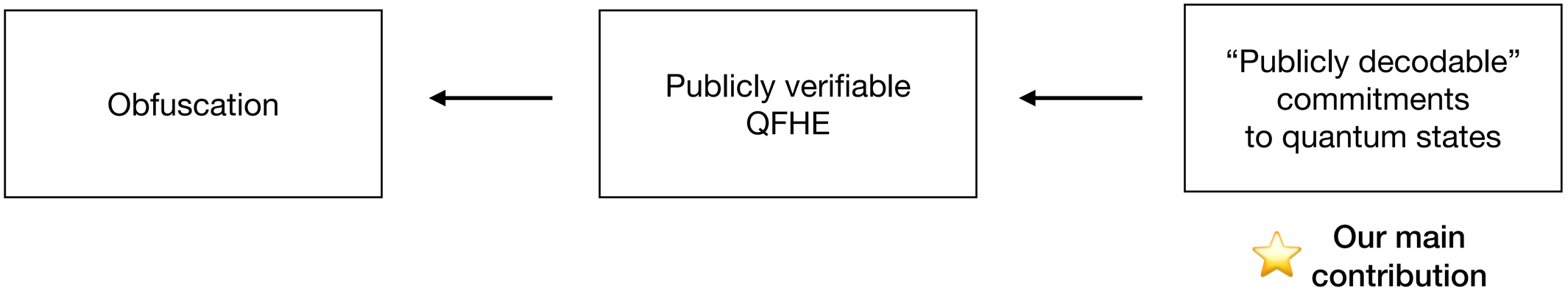
Summary

Summary

Theorem [B**G**MS]**.** Assuming LWE, there is a compiler from classical ideal obfuscation → quantum ideal obfuscation for quantum circuits that compute classical functions.

Summary

Theorem [BGMS]. Assuming LWE, there is a compiler from classical ideal obfuscation → quantum ideal obfuscation for quantum circuits that compute classical functions.



Open questions

Open questions

Can improve our construction along two axes, both would resolve major open questions:

Open questions

Can improve our construction along two axes, both would resolve major open questions:

1. Classical obfuscation for more **more expressive circuits**? (E.g. unitary, general quantum maps)
    - Due to HT'26, classical obfuscation for unitaries $\Longrightarrow$ obfuscation for general quantum maps
    - Implies verifiable delegated quantum computation for these circuits, currently out of reach

Open questions

Can improve our construction along two axes, both would resolve major open questions:

1. Classical obfuscation for more **more expressive circuits**? (E.g. unitary, general quantum maps)
    - Due to HT'26, classical obfuscation for unitaries $\Longrightarrow$ obfuscation for general quantum maps
    - Implies verifiable delegated quantum computation for these circuits, currently out of reach

2. Compiler in the **plain model (no classical ideal obfuscation)** that takes
   Classical iO $\rightarrow$ iO for quantum circuits that compute classical functions
    - Implies publicly verifiable delegated quantum computation in the plain model, also currently out of reach