On the (Im)plausibility of Public-Key Quantum Money from CRHFs

ASIACRYPT 2023

Prabhanjan Ananth
UCSB

Zihan Hu
Tsinghua University

Henry Yuen
Columbia University

eprint 2023/069

No-Cloning Theorem

This motivates many classically impossible primitives

- Copy Protection [Aar09, …]
- Unclonable Encryption [BL20, …]
- Quantum Money [Wie83, AC13, …]
- …

Public-Key Quantum Money (PKQM)

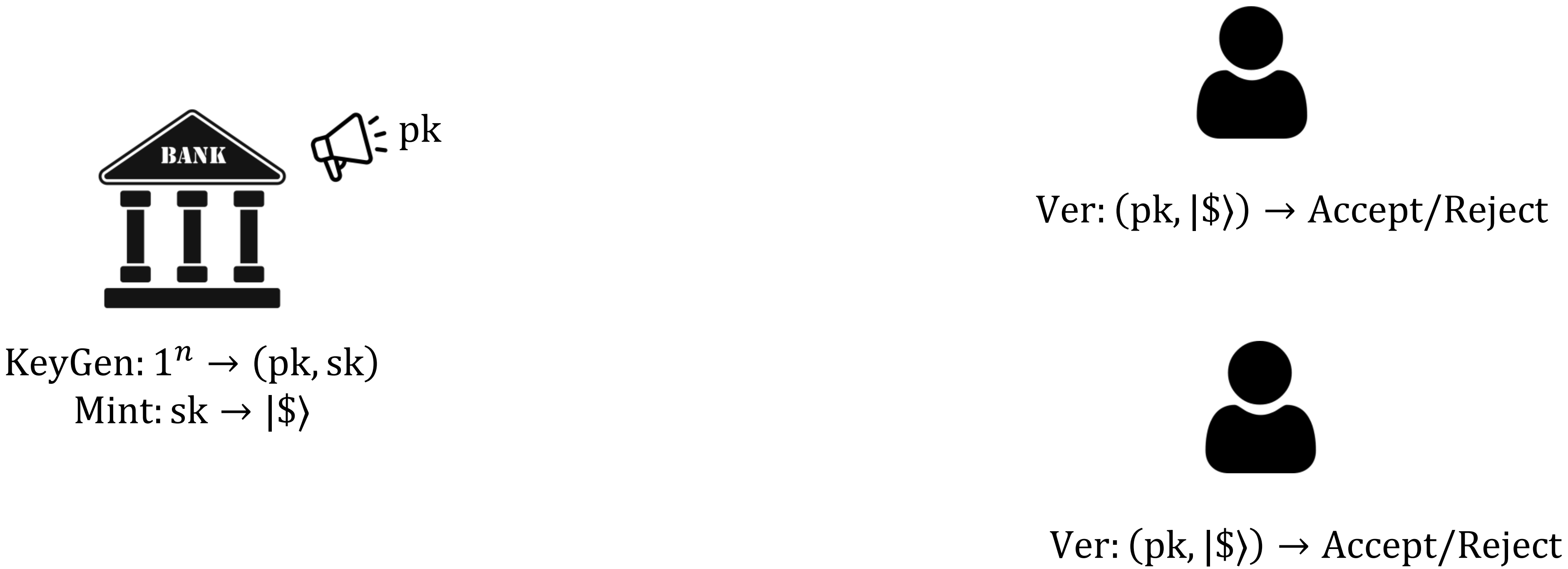
pk

KeyGen: $1^n \rightarrow (\mathrm{pk}, \mathrm{sk})$

Mint: $\mathrm{sk} \rightarrow |\$\rangle$

- Bank can generate the money state efficiently

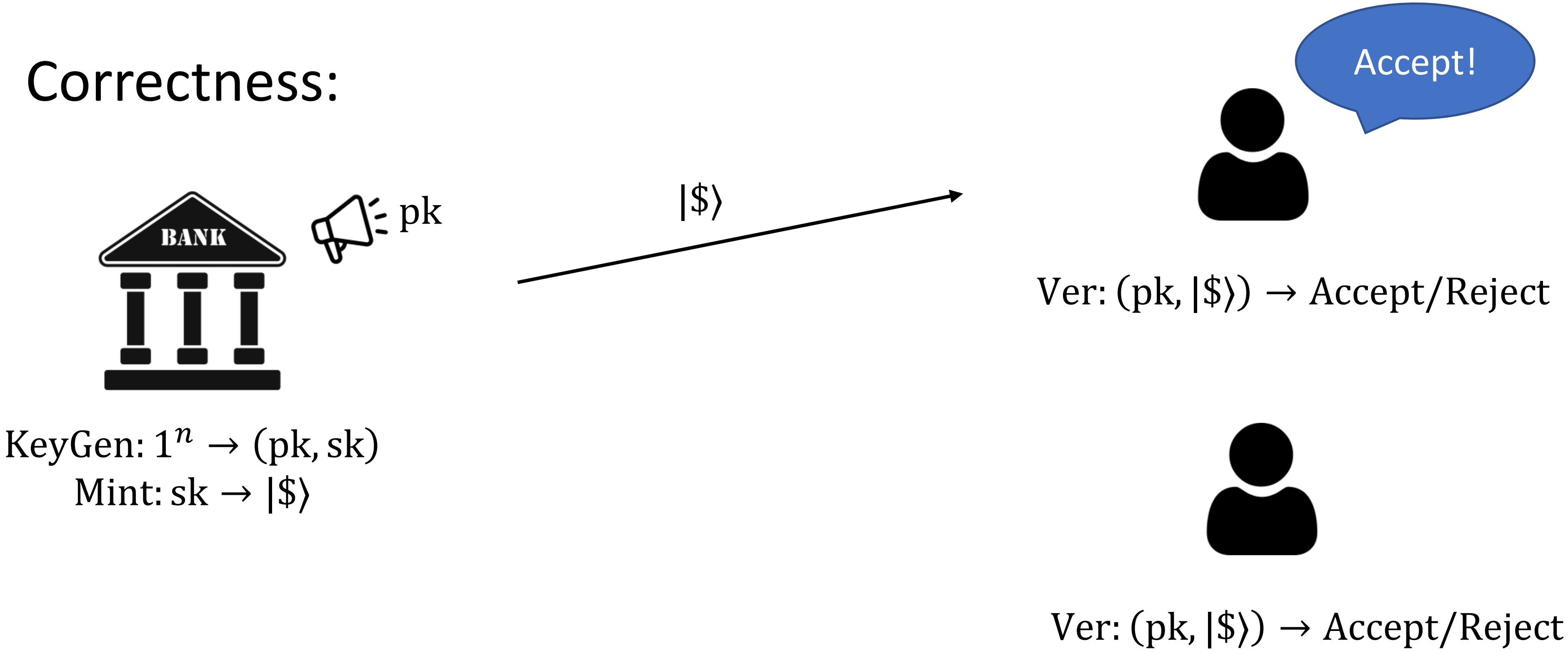
Public-Key Quantum Money (PKQM)



- Bank can generate the money state efficiently
- Honest parties can check whether the money state is valid

Public-Key Quantum Money (PKQM)

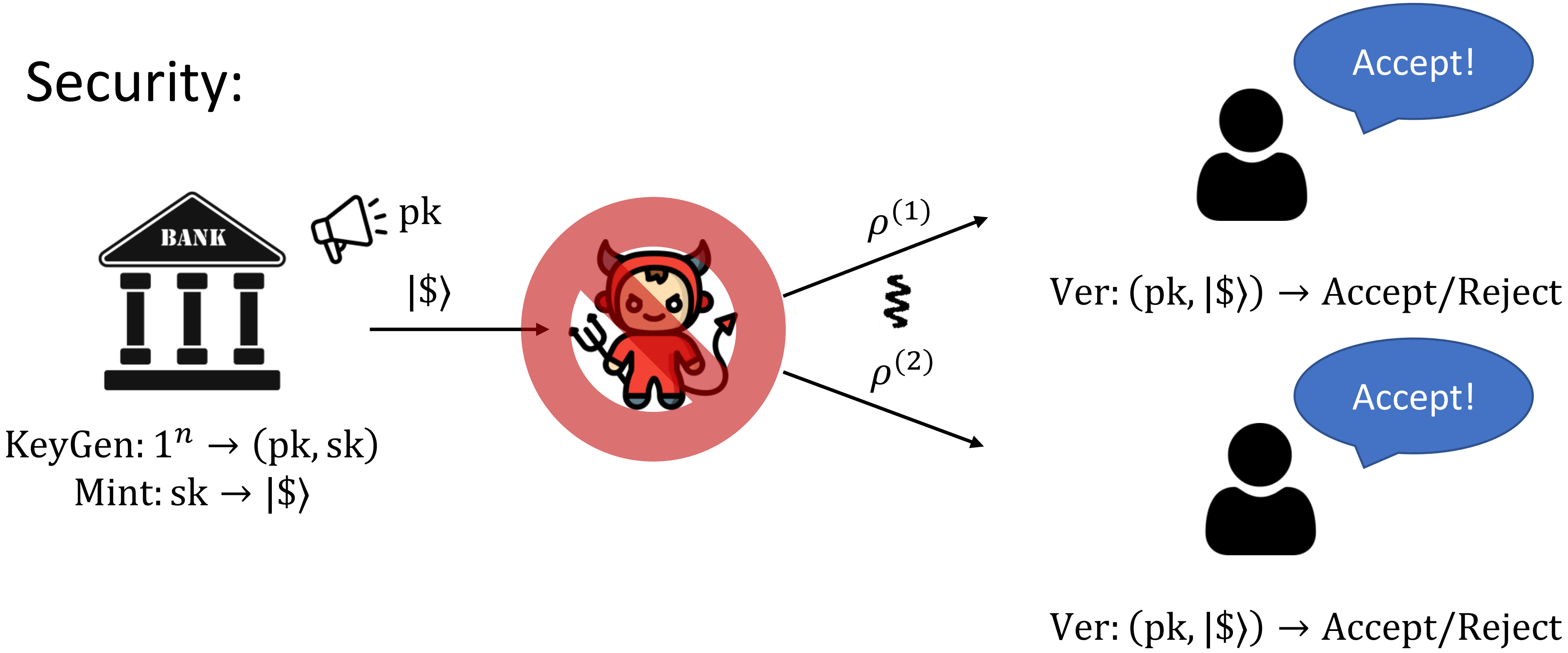
Correctness:



- Bank can generate the money state efficiently
- Honest parties can check whether the money state is valid

Public-Key Quantum Money (PKQM)

Security:



- Bank can generate the money state efficiently
- Honest parties can check whether the money state is valid
- It is difficult for a malicious party (adversary) to counterfeit

Constructions of PKQM

- Oracle Model: unconditional construction [AC13]

- Standard Model:
  - Constructions based on non-standard assumptions [FGH+12, KSS21, Zha21].
  - Constructions based on very strong cryptographic primitives, e.g. post-quantum indistinguishability obfuscation [Zha21].

Question: Is the difficulty in constructing PKQM from standard assumptions **inherent**?

When everyone can query the oracle at unit cost,

Primitive A

Primitive B

Our Result

When everyone can query a random oracle R and PSPACE oracle,

(by default, quantum queries)

CRHFs…

a class of PKQMs
where Ver only has classical access to R

Our Result

When everyone can query a random oracle R and PSPACE oracle,

(by default, quantum queries)

CRHFs…

a class of PKQMs
where Ver only has classical access to R

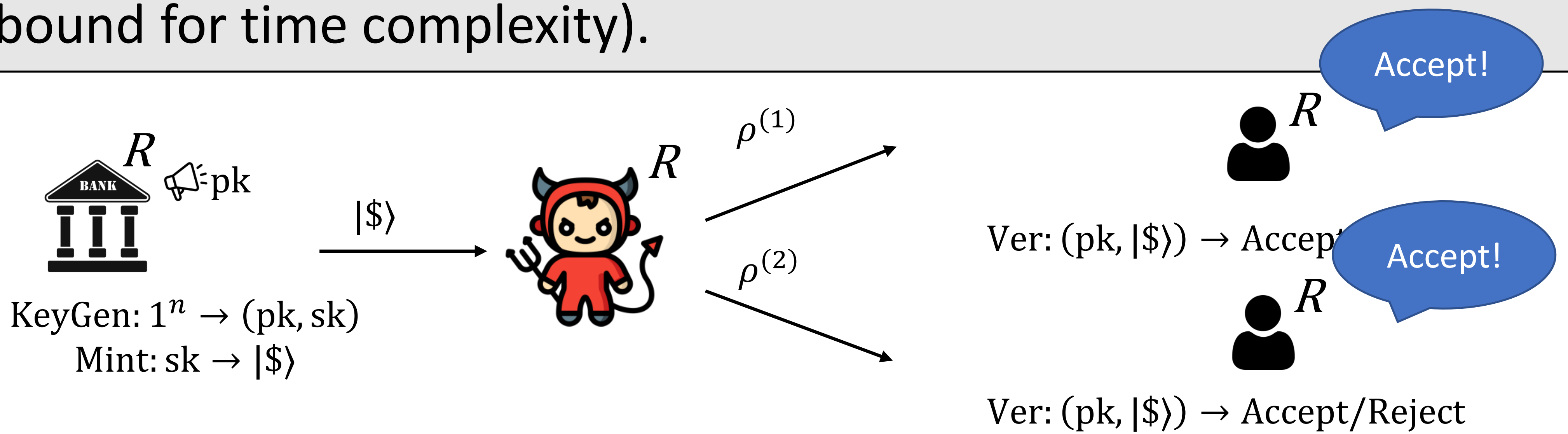
There does not exist reusable and secure public-key quantum money scheme $\left(\mathrm{KeyGen}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Mint}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Ver}^{R,|\mathrm{PSPACE}\rangle}\right)$ where R is a random oracle.

For Today,

There does not exist reusable and secure public-key quantum money scheme $\left(\mathrm{KeyGen}^R, \mathrm{Mint}^R, \mathrm{Ver}^R\right)$ where R is a random oracle, and we only require that the query complexity of every party is polynomial (no bound for time complexity).

For Today,

There does not exist reusable and secure public-key quantum money scheme $(\mathrm{KeyGen}^R, \mathrm{Mint}^R, \mathrm{Ver}^R)$ where R is a random oracle, and we only require that the query complexity of every party is polynomial (no bound for time complexity).



Technical Details: without R

Goal of : Find good money states.

Technical Details: without R

Goal of : Find good money states.

Just do brute-force search!

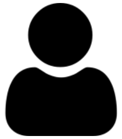
- Repeat the following for enough times:
  - Guess a random state
  - If it's good, output it
- Output $|0\rangle$ if we run out of time

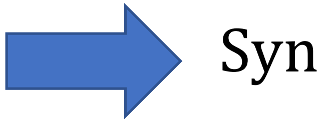
Technical Details: without R

Goal of  : Find good money states.

Just do brute-force search!

- Repeat the following for enough times:
  - Guess a random state
  - If it's good, output it
- Output $|0\rangle$ if we run out of time

We can easily synthesize a good money state for .



Technical Details: with R

- When Ver makes queries to R, the brute-force algorithm also needs R to work…

We need to query R here

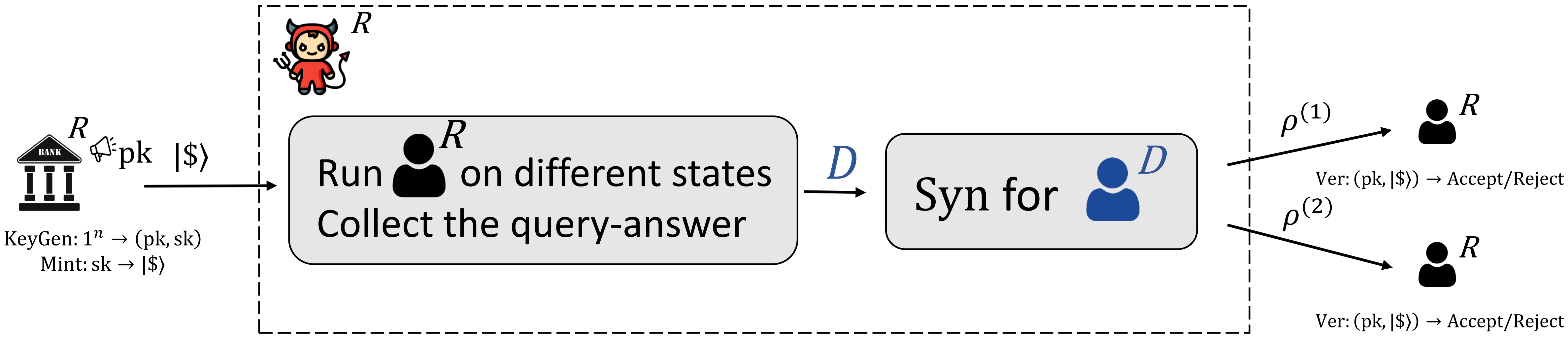
- Repeat the following for enough times:
  - Guess a random state
  - If it's good, output it
- Output $|0\rangle$ if we run out of time

Technical Details: with R

- The key idea: find a proper database D to replace the random oracle

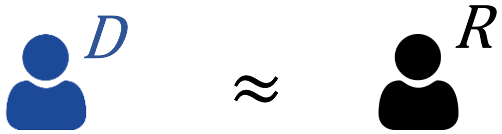
Technical Details: with R

- The key idea: find a proper database D to replace the random oracle



Technical Details: with R

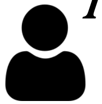
What's the requirement for D?

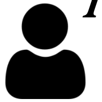


Technical Details: with R

What's the requirement for D?

On a money state $|\$\rangle$ and the synthesized state σ,

 $^{D}(|\$\rangle) \approx$  $^{R}(|\$\rangle)$

 $^{D}(\sigma) \approx$  $^{R}(\sigma)$

Technical Details: with R

What's the requirement for D?

On a money state $|\$\rangle$ and the synthesized state σ,

Technical Details: with R

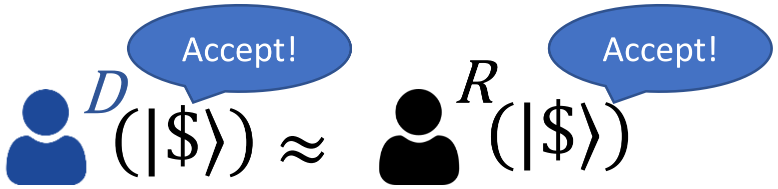
What's the requirement for D?

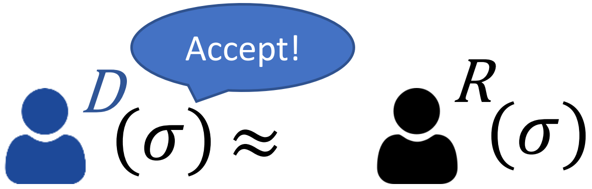
On a money state $|\$\rangle$ and the synthesized state σ,

Technical Details: with R

What's the requirement for D?

On a money state $|\$\rangle$ and the synthesized state σ,



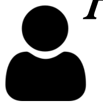
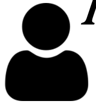


Technical Details: with R

What's the requirement for D?

On a money state $|\$\rangle$ and the synthesized state σ,

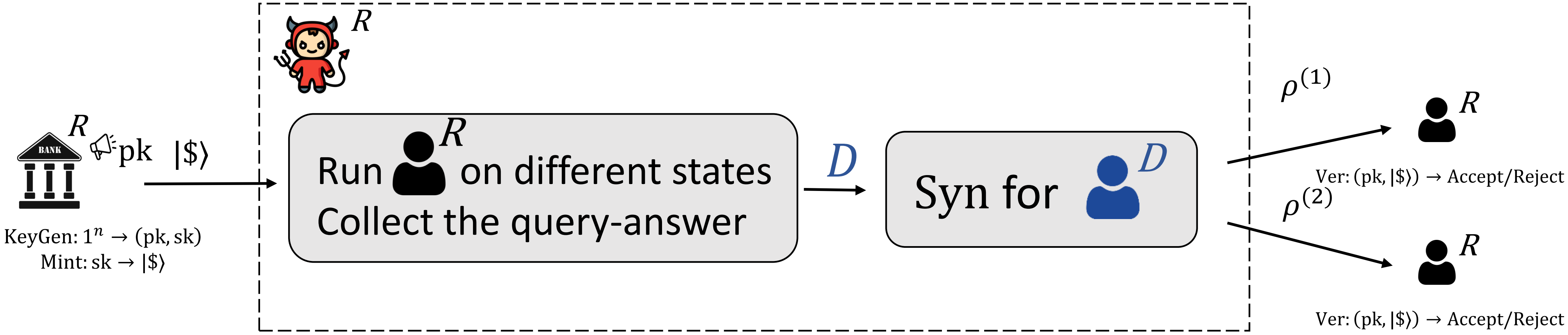
Technical Details: with R

How to get D such that $^{D}(|\$\rangle) \approx$ $^{R}(|\$\rangle)$ $^{D}(\sigma) \approx$ $^{R}(\sigma)$

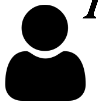
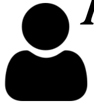
Technical Details: with R

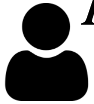
How to get D such that $\quad {}^{D}(|\$\rangle) \approx {}^{R}(|\$\rangle) \qquad {}^{D}(\sigma) \approx {}^{R}(\sigma)$

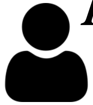
By lazy sampling, $\quad {}^{R} = {}^{D'}$ where $D' = D \cup D_{\mathrm{KeyGen}} \cup D_{\mathrm{Mint}}$



Technical Details: with R

How to get D such that  $^{D}(|\$\rangle) \approx$  $^{R}(|\$\rangle)$  $^{D}(\sigma) \approx$  $^{R}(\sigma)$

By lazy sampling,  ^{R} $=$  $^{D'}$ where $D' = D \cup D_{\mathrm{KeyGen}} \cup D_{\mathrm{Mint}}$

If we make a mistake:  $^{D}(\rho) \not\approx$  $^{R}(\rho)$

we make progress  ^{R} learns $D_{\mathrm{KeyGen}} \cup D_{\mathrm{Mint}}$ better

Technical Details: with R

How to get D such that $^{D}(|\$\rangle) \approx {}^{R}(|\$\rangle)$ $\quad$ $^{D}(\sigma) \approx {}^{R}(\sigma)$

By lazy sampling, $^{R} = {}^{D'}$ where $D' = D \cup D_{\text{KeyGen}} \cup D_{\text{Mint}}$

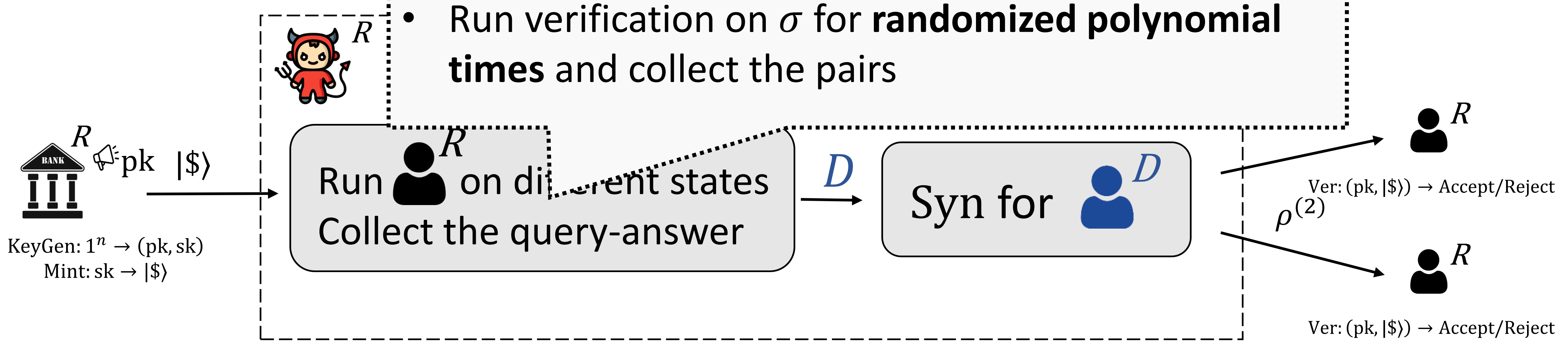
If we make a mistake: $^{D}(\rho) \not\approx {}^{R}(\rho)$

we make progress  ^{R} learns $D_{\text{KeyGen}} \cup D_{\text{Mint}}$ better

At most $|D_{\text{KeyGen}} \cup D_{\text{Mint}}| = \mathbf{poly}$ mistakes!

Technical Details: with R

- Run verification on $|\$\rangle$ for **randomized polynomial times** and collect the pairs
- Run verification on σ for **randomized polynomial times** and collect the pairs



Ideas for $(\mathrm{KeyGen}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Mint}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Ver}^{R,|\mathrm{PSPACE}\rangle})$

- From query-efficient to time-efficient with $|\mathrm{PSPACE}\rangle$
  - Use Marriott-Watrous technique (MW technique) [MW05]. The brute force algorithm actually runs in quantum polynomial space.

> "Quantum Brute-Force Search"
> - Repeat the following for exponential times:
>   - Start with maximally mixed state
>   - Apply MW technique on it to estimate the acceptance probability
>   - If the estimation is high enough, output the residual state
> - Output $|0\rangle$ if we run out of time

Ideas for $(\mathrm{KeyGen}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Mint}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Ver}^{R,|\mathrm{PSPACE}\rangle})$

- From query-efficient to time-efficient with $|\mathrm{PSPACE}\rangle$
  - Use Marriott-Watrous technique (MW technique) [MW05]. The brute force algorithm actually runs in quantum polynomial space.
  - Quantum states computable in quantum polynomial space can be synthesized by a quantum polynomial time algorithm with $|\mathrm{PSPACE}\rangle$ [RY21, MY23].

Ideas for $(\mathrm{KeyGen}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Mint}^{|R\rangle,|\mathrm{PSPACE}\rangle}, \mathrm{Ver}^{R,|\mathrm{PSPACE}\rangle})$

- From query-efficient to time-efficient with $|\mathrm{PSPACE}\rangle$
  - Use Marriott-Watrous technique (MW technique) [MW05]. The brute force algorithm actually runs in quantum polynomial space.
  - Quantum states computable in quantum polynomial space can be synthesized by a quantum polynomial time algorithm with $|\mathrm{PSPACE}\rangle$ [RY21, MY23].
- $\mathrm{KeyGen}, \mathrm{Mint}$: from R to $|R\rangle$
  - The same construction also works.
  - In the analysis, use Zhandry's compressed oracle technique [Zha18] to find analogue of $D_{\mathrm{KeyGen}} \cup D_{\mathrm{Mint}}$.

Wrap-up

- Take-Away Message

  Public-key quantum money schemes are difficult to construct.

  A classical access to a weak cryptographic primitive may not be enough.

- Open Problems
  - Quantum queries in verification?
  - Separations between other primitives?

Thank you!

eprint 2023/069