新世代暗号の設計・評価ワークショップ＠九州大学

様々な格子ベースのゼロ知識証明を QROM安全にするシンプルな手法について

2021/11/16

勝又秀一（産総研/AIST）

産総研

*CRYPTO21の結果より

Our Result

A Simple Semi-Generic Method to Construct QROM Secure Lattice-based ZK PoKs*

*In this talk, we do not differentiate between "proofs" and "arguments"

- New tool: Extractable Linear Homomorphic Commitment (ExtLinHC)

- Semi-Generic Transform:

Many **Classically** Secure Lattice-based **Public-Coin Interactive Protocol** + ExtLinHC ⇒ QROM Secure NIZK (w/ Online-Extractability)

Agenda

1. Background and Motivation

2. More on Lattice-based QROM NIZKs

3. Our Result: ExtLinHC

4. Constructing ExtLinHC

1. Background and Motivation

Preparation: Non-Interactive Zero-Knowledge

$x \in L$

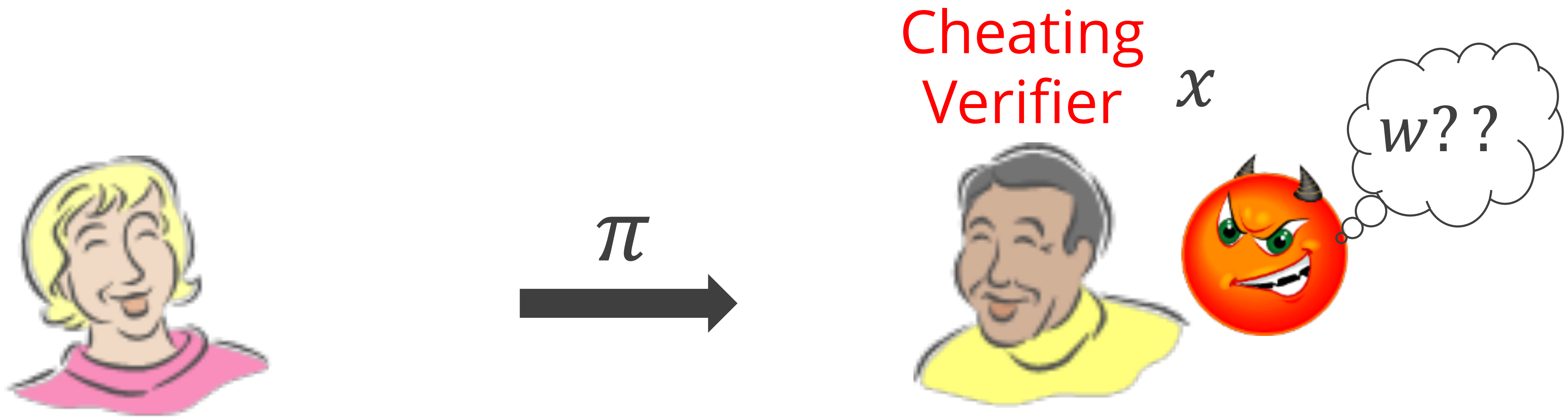
Prover(x, w)

Verifier x

π →

✓ **Completeness**: If $(x, w) \in R_L$, then Verifier is convinced.

Zero-Knowledge



- ✓ **Completeness:** If $(x, w) \in R_L$, then Verifier is convinced.
- ✓ **Zero-Knowledge:** If $x \in L$, Verifier only learns that $x \in L$.

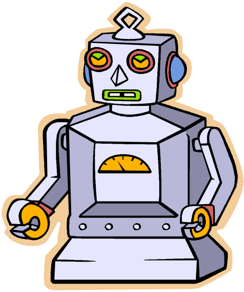
Proof of Knowledge

Cheating Prover

$x \in L$

π

Extractor x



w s.t. $(x, w) \in R_L$

✓ **Completeness:** If $(x, w) \in R_L$, then Verifier is convinced.

✓ **Zero-Knowledge:** If $x \in L$, Verifier only learns that $x \in L$.

✓ **Proof of Knowledge:** There exists an efficient extractor Ext s.t., if a cheating Prover outputs a valid π, then <u>Ext</u> outputs w s.t. $(x, w) \in R_L$.

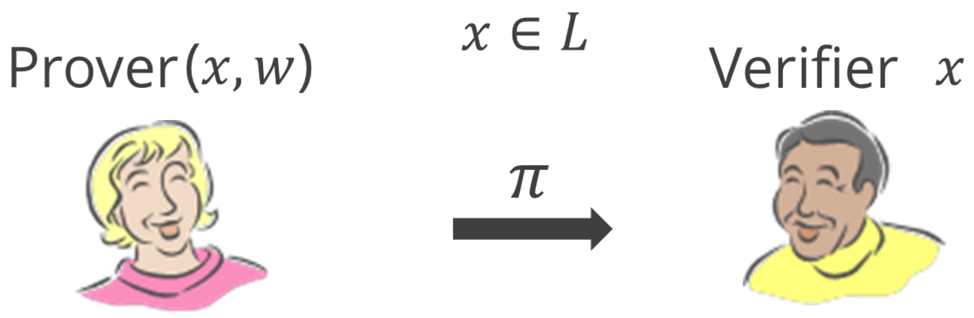
(*w/ extra capabilities)

*Implies soundness

Models

- Standard Model

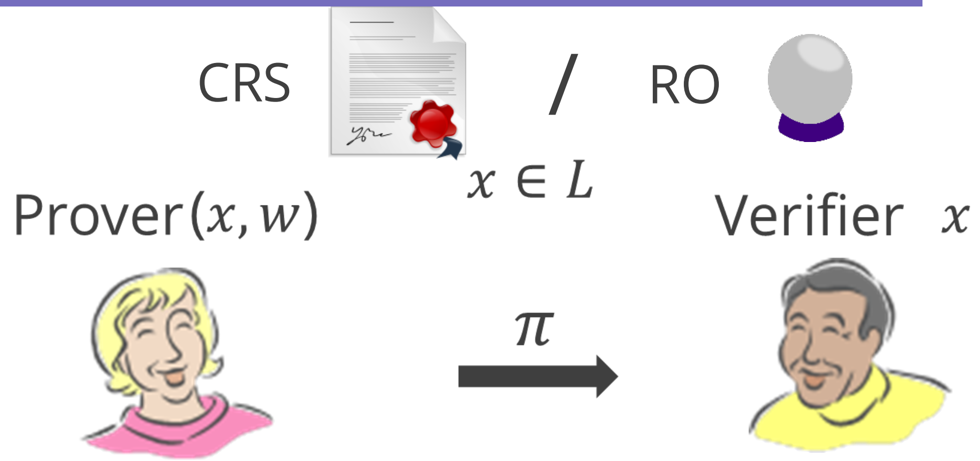
Only exist for trivial languages [GO94]



Models

- Standard Model

  Only exist for trivial languages [GO94]



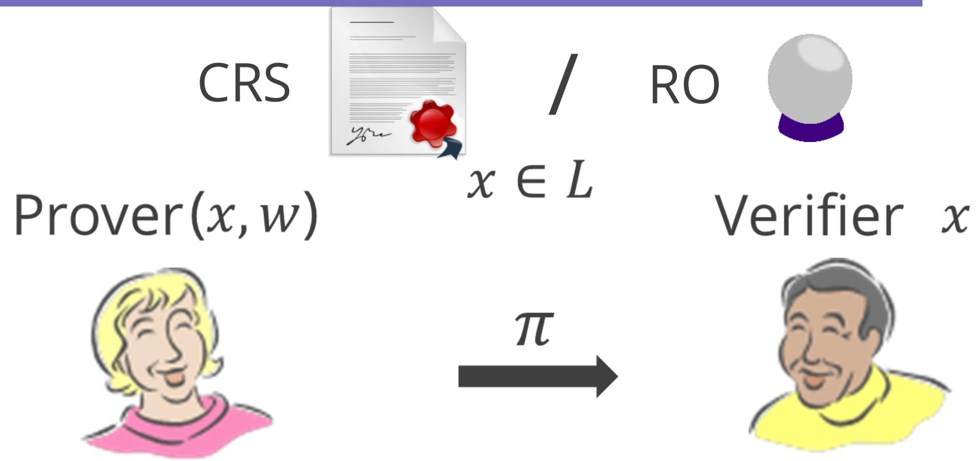
- Common Reference String Model

  Users are given a CRS generated by a trusted authority.

- Random Oracle Model

  Users are given access to a RO.

Models



- Standard Model

  Only exist for trivial languages [GO94]

- Common Reference String Model

  Users are given a CRS generated by a trusted authority.

  ➡ Provably secure but could be impractical.
  Also, trusted setup is required.

- Random Oracle Model

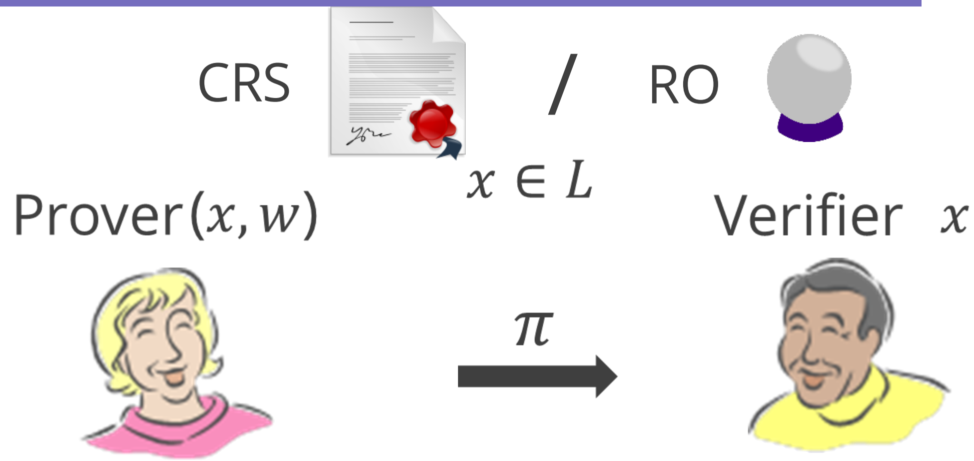
  Users are given access to a RO.

  ➡ Only secure in the ROM but typically most efficient and practical.

Models

- Standard Model

  Only exist for trivial languages [GO94]



- Common Reference String Model

  Users are given a CRS generated by a trusted authority.

  ➡ Provably secure but could be impractical.
  Also. trusted setup is required.

- Random Oracle Model

  Users are given access to a RO.

  ➡ Only secure in the ROM but typically most efficient and practical.

This Talk

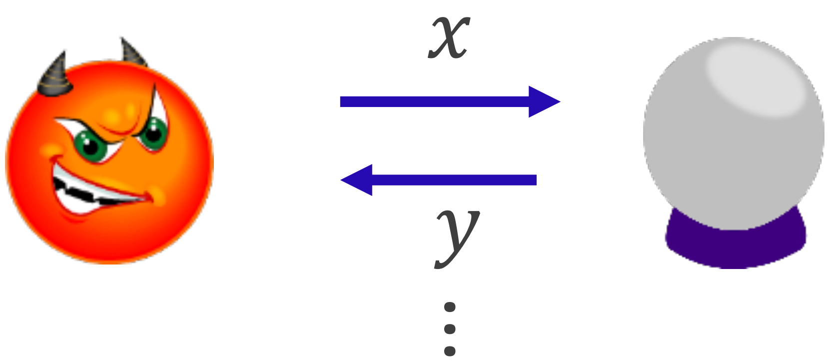
Classical vs Quantum ROM

A **quantum adversary** can evaluate hash function over qbits in real-world.

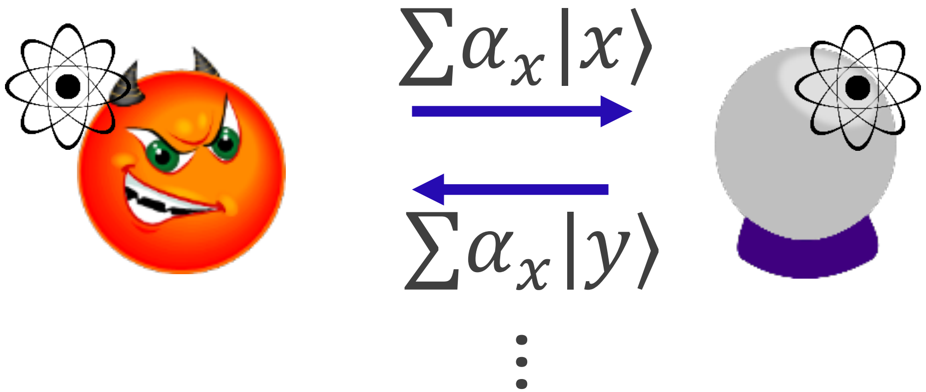
$$\sum_x \alpha_x |x\rangle \rightarrow \sum_x \alpha_x |x, \mathrm{H}(x)\rangle$$

QROM should model this capability!

Classical ROM



Quantum ROM

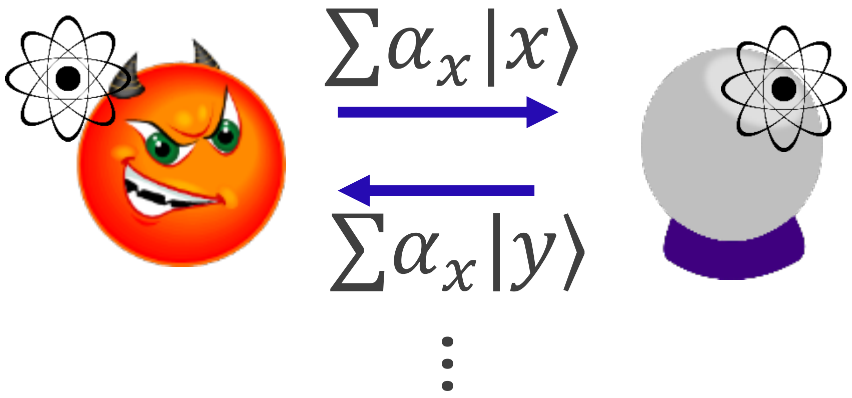


Some Difficulty in QROM

Typical CROM proof that "seems" hard to import to QROM.

① Observe the adversary's input query

② Know the corresponding output



Why? May disturb adversary's quantum state.

Some Difficulty in QROM

Typical CROM proof that "seems" hard to import to QROM.

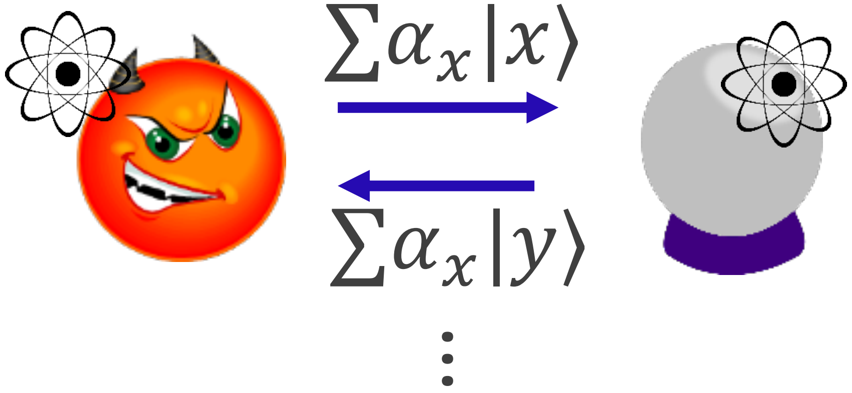
① Observe the adversary's input query

② Know the corresponding output

Why? May disturb adversary's quantum state.

③ Adaptively program the RO

Why? The adversary may query on the entire input space in superposition.



Some Difficulty in QROM

Typical CROM proof that "seems" hard to import to QROM.

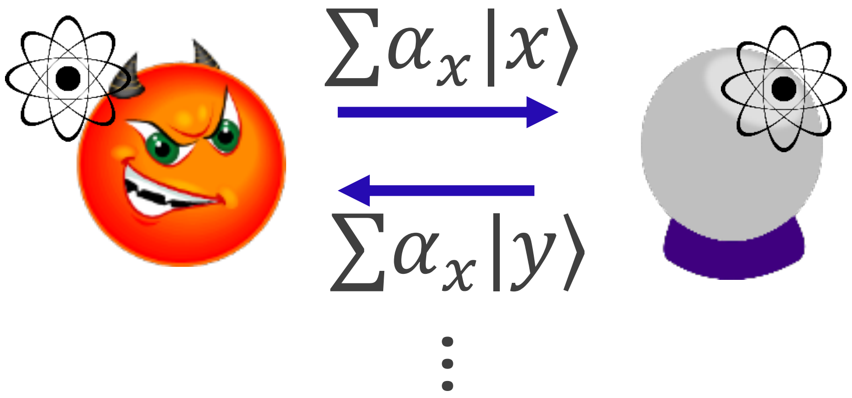
① Observe the adversary's input query

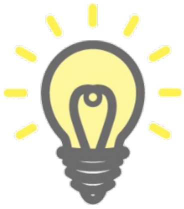
② Know the corresponding output

Why? May disturb adversary's quantum state.

③ Adaptively program the RO

Why? The adversary may query on the entire input space in superposition.



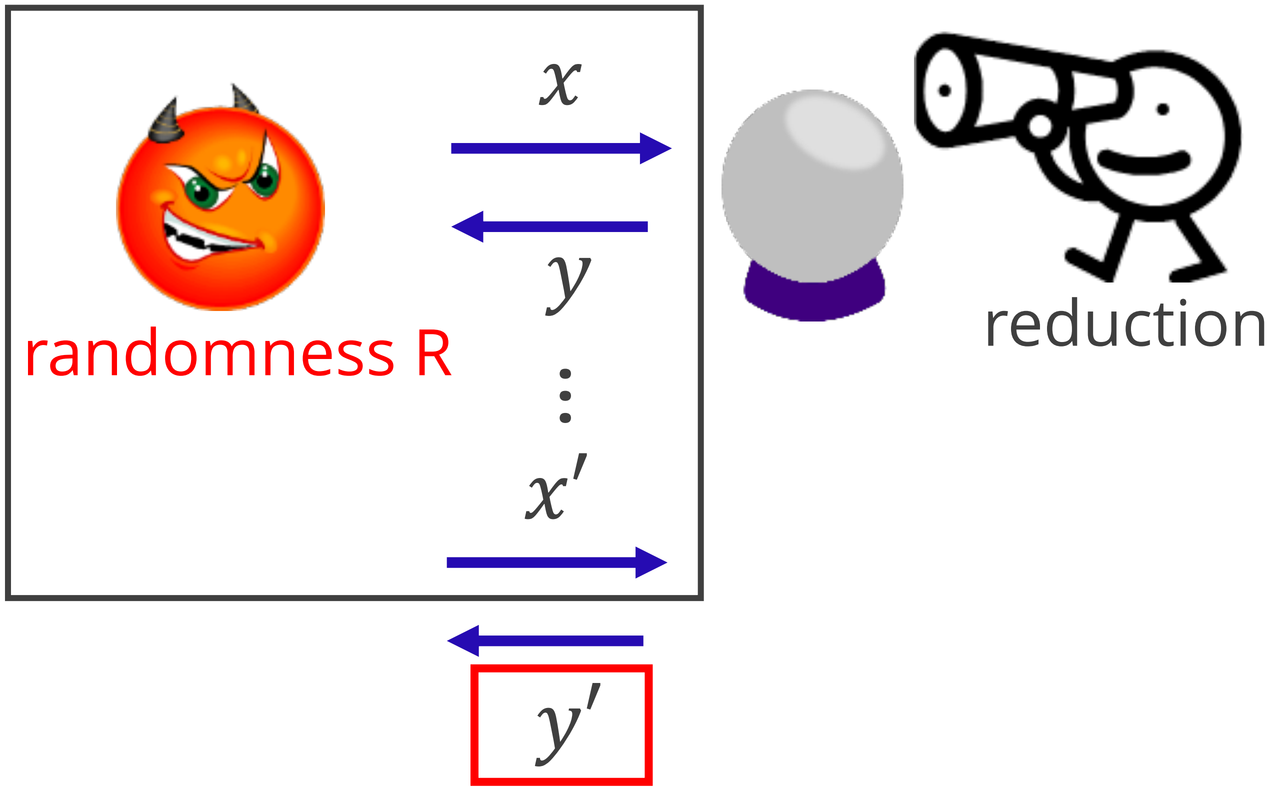


We now know many ways to overcome these seeming hardness but it is not as "free" as in the classical setting.

Some Other Difficulties in Quantum Setting

Handling quantum adversaries is difficult regardless of ROM being classical or quantum.

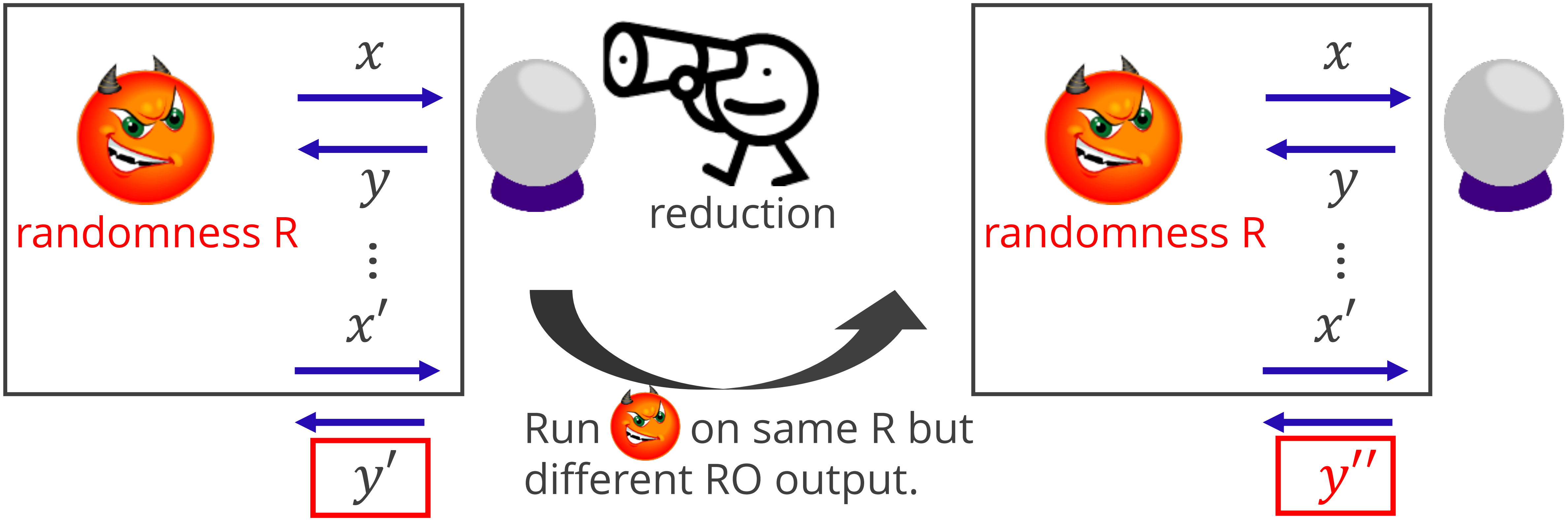
<u>Representative Example) Rewinding</u>



Some Other Difficulties in Quantum Setting

Handling quantum adversaries is difficult regardless of ROM being classical or quantum.

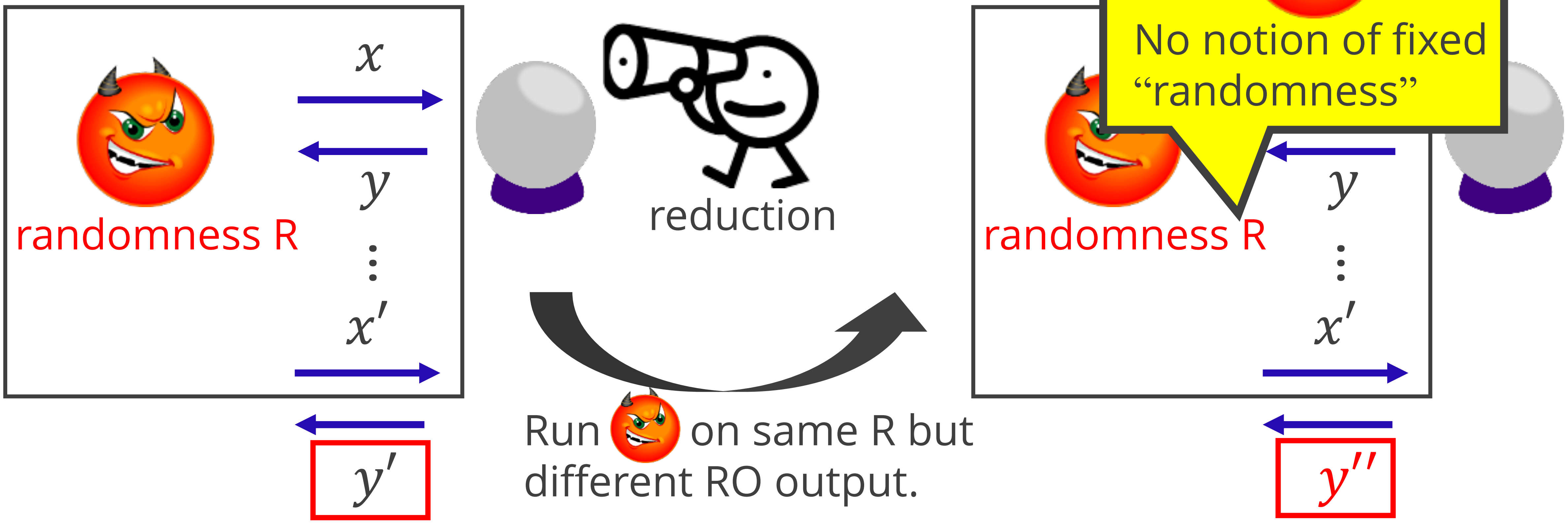
Representative Example) Rewinding



Some Other Difficulties in Quantum Setting

Handling quantum adversaries is difficult regardless of ROM being classical or quantum.

Representative Example) Rewinding



State-of-the-Affair for Lattice-based NIZKs

- **CRS-NIZK** (w/ quantum adversary)

  Correlation Intractable hash approach: [CCHLRRW19] and [PS19]

- **CROM-NIZK** (w/ classical adversary)

inefficient

efficient

State-of-the-Affair for Lattice-based NIZKs

- **CRS-NIZK** (w/ quantum adversary)

  Correlation Intractable hash approach: [CCHLRRW19] and [PS19]

- **CROM-NIZK** (w/ classical adversary)

  Stern protocol approach [Ste94, KTX08]

  - Combinatorial method and easy to understand.

inefficient

efficient

State-of-the-Affair for Lattice-based NIZKs

- **CRS-NIZK** (w/ quantum adversary)

  Correlation Intractable hash approach: [CCHLRRW19] and [PS19]

- **CROM-NIZK** (w/ classical adversary)

  Stern protocol approach [Ste94, KTX08]
  - Combinatorial method and easy to understand.

  Fiat-Shamir w/ Abort approach [Lyu09,Lyu12]
  - [Lyu09,Lyu12] is an analog of Schnorr's protocol.
  - Many tricks exploiting lattice structure for better efficiency.
  - Efficiency increased drastically in the past few years: [BLS19,YAZXYW19,ESLL19,ALS20...].

inefficient

↓

efficient

State-of-the-Affair for Lattice-based NIZKs

- **CRS-NIZK** (w/ quantum adversary)

  Correlation Intractable hash approach

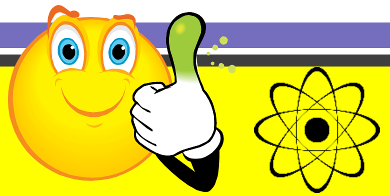
- **CROM-NIZK** (w/ classical adversary)

  Stern protocol approach [Ste94, KTX08]

  - Combinatorial method and easy to understand.

  Fiat-Shamir w/ Abort approach [Lyu09,Lyu12]

  - [Lyu09,Lyu12] is an analog of Schnorr's protocol.
  - Many tricks exploiting lattice structure for better efficiency.
  - Efficiency increased drastically in the past few years: [BLS19,YAZXYW19,ESLL19,ALS20...].



Due to its commit-and-open nature, QROM security is known.

inefficient

efficient

State-of-the-Affair for Lattice-based NIZKs

- **CRS-NIZK** (w/ quantum adversary)

  Correlation Intractable hash approach

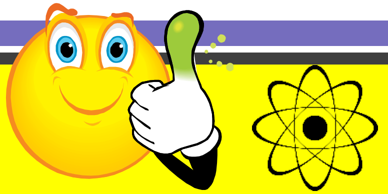
- **CROM-NIZK** (w/ classical adversary

  Stern protocol approach [Ste94, KTX08]

  - Combinatorial method and easy to understa

  Fiat-Shamir w/ Abort approach [Lyu09,Lyu12]

  - [Lyu09,Lyu12] is an analog of Schnorr's pro
  - Many tricks exploiting lattice structure for be
  - Efficiency increased drastically in the past few years: [BLS19,YAZXYW19,ESLL19,ALS20…].



Due to its commit-and-open nature, QROM security is known.

Other than [Lyu09,Lyu12] not much about QROM security is known.

inefficient

efficient

2. More on Lattice-based QROM NIZKs

Recap: Sigma-Protocol (or Public-Coin Interactive Proof (PCIP))

Prover (x, w)

a →

$c \in C$ ←

z →

Verifier x

Recap: Sigma-Protocol (or Public-Coin Interactive Proof (PCIP))

Prover (x, w) — Verifier x

Prover → Verifier: a

Verifier → Prover: $c \in C$

Prover → Verifier: z

Standard Security Notions

✓ **Honest-Verifier ZK:**

$\exists$PPT Sim such that $\{(a, c, z) \leftarrow \mathrm{Sim}(x, c)\} \approx \{(a, c, z) \leftarrow \langle P(x, w), V_c(x) \rangle\}$.

*Simulator knows no witness!

Recap: Sigma-Protocol (or Public-Coin Interactive Proof (PCIP))

Prover (x, w) Verifier x

a →

$c \in C$ ←

z →

Standard Security Notions

✓ **Honest-Verifier ZK:**

$\exists$PPT Sim such that $\{(a, c, z) \leftarrow \text{Sim}(x, c)\} \approx \{(a, c, z) \leftarrow \langle P(x, w), V_c(x) \rangle\}$.

*Simulator knows no witness!

✓ **Special Soundness:**

$\exists$PT Ext such that $\text{Ext}\big(x, a, (c, z), (c', z')\big) \rightarrow w \; s.t. \; (x, w) \in R_L$

*2 valid transcripts with same a!

Sigma Protocol to NIZK Transform

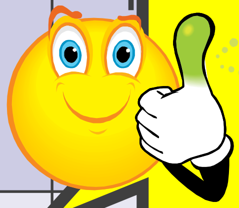
Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir ‘88	any	One hash	rewind			
Fischlin ‘05						
Unruh ‘15						

Sigma Protocol to NIZK Transform

Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir ‘88	any	One hash	rewind			
Fischlin ‘05	-Small $\|C\|$ -Quasi-unique response	None (but parallel rep.)	Straight-line (= tight proof)			
Unruh ‘15						

Sigma Protocol to NIZK Transform

Transform	Classical			Quantum
	Type of Sigma prot.	Proof overhead	PoK	
Fiat-Shamir ‘88	any	One hash	rewind	
Fischlin ‘05	-Small $\|C\|$ -Quasi-unique response	None (but parallel rep.)	Straight-line (= tight proof)	
Unruh ‘15				



In general Fiat-Shamir is the best transform in the classical setting.

Sigma Protocol to NIZK Transform

Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir '88	any	One hash	rewind		????	
Fischlin '05	-Small $\|C\|$ -Quasi-unique response	None (but parallel rep.)	Straight-line (= tight proof)		????	
Unruh '15						

Sigma Protocol to NIZK Transform

Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir '88	any	One hash	rewind		????	
Fischlin '05	-Small $\|C\|$ -Quasi-unique response	None (but parallel rep.)	Straight-line (= tight proof)		????	
Unruh '15		（右同）		Small $\|C\|$	x$\|C\|$ with parallel rep.	Straight-line (= tight proof)

Sigma Protocol to NIZK Transform

Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir '88 ([LZ19,DFMR19])	any	One hash	rewind	⚠ collapsing	One hash	⚠ rewind (lose at least $O(q^2)$)
Fischlin '05	-Small $\|C\|$ -Quasi-unique response	None (but parallel rep.)	Straight-line (= tight proof)		????	q: #RO query t: #valid transcript to extract witness Use to be $O(q^{2t-1})$ till very recent [CMSZ21]
Unruh '15		(右同)		Small $\|C\|$	⚠ x$\|C\|$ with parallel rep.	Straight-line (= tight proof)

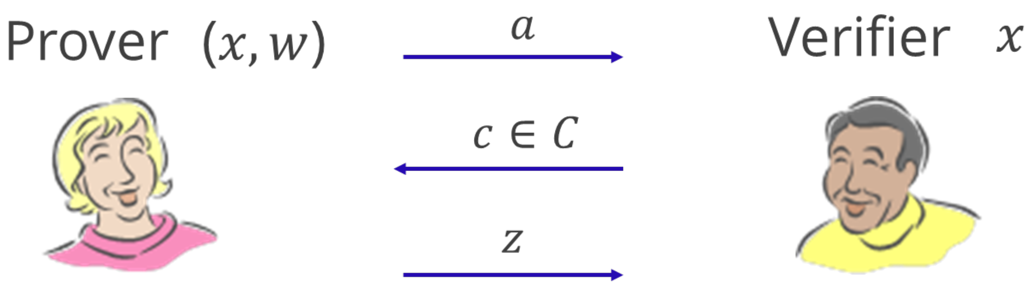
Sigma Protocol to NIZK Transform

Transform	Classical			Quantum: Type of Sigma prot.	Quantum: Proof overhead	Quantum: PoK
Fiat-Shamir '88 ([LZ19,DFMR19])	any	One hash	rewind	⚠ collapsing	One hash	⚠ rewind (lose at least $O(q^2)$)
Fischlin '05	response	parallel	tight		????	q: #RO query t: #valid transcript to extract witness Use to be $O(q^{2t-1})$ till very recent [CMSZ21]
Unruh '15	(右同)			Small $\|C\|$	⚠ x$\|C\|$ with parallel rep.	Straight-line (= tight proof)

Not all existing Sigma prot. are known to be collapsing.

Proof overhead is large…

In a Bit More Detail: Fiat-Shamir



FS

1. a
2. $c \leftarrow H(x, a)$ 
3. z
4. Output(a, c, z)

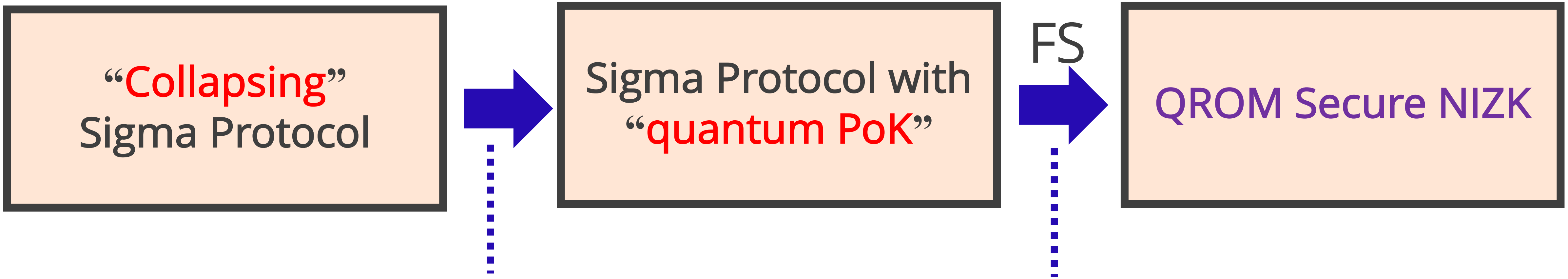
In Classical Setting...

- Rewind the cheating prover so that it answers to two different challenges.
- Adaptively reprogram the RO at $H(x, a)$ to two different challenges.

Both procedures are difficult in QROM...?? 

In a Bit More Detail: Fiat-Shamir

QROM security by [DFMS19,LZ19].

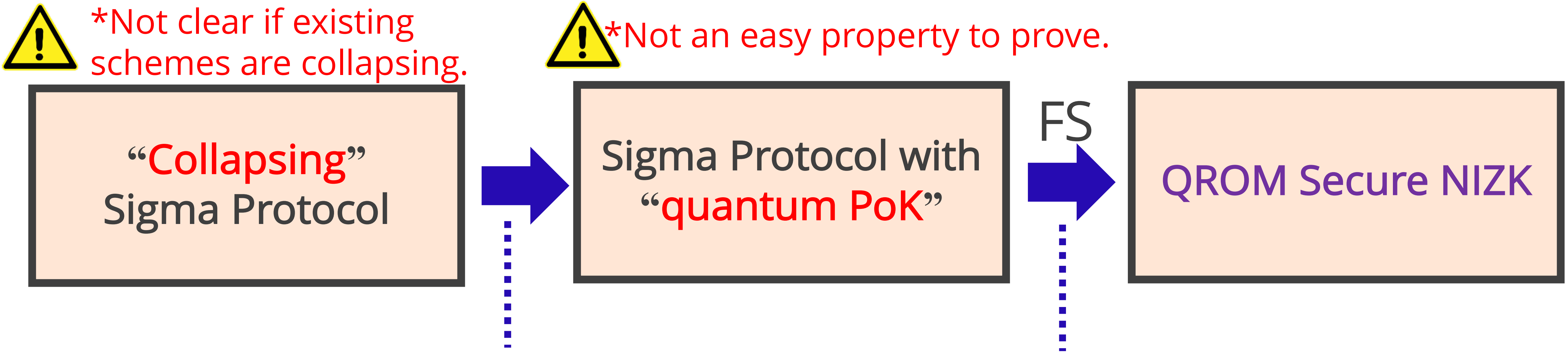


Collapsingness allows to rewind quantum adversaries w/o destroying its state.

General method to reprogram the QRO on specific query input.

In a Bit More Detail: Fiat-Shamir

QROM security by [DFMS19,LZ19].



Collapsingness allows to rewind quantum adversaries w/o destroying its state.

General method to reprogram the QRO on specific query input.

*Seems to incur at least $O(q^{2n})$ reduction loss for n different programmed points.

In a Bit More Detail: Unruh

Getting around rewinding and adaptive reprogramming [U15].

Rough Idea: Let Prover commit to all (challenge, response) pair.

1. a
2. For $i \in C$
   i. Generate response z_i
   ii. $\mathrm{com_i} = \mathrm{Com}(z_i; \mathrm{rand_i})$
3. $c \leftarrow H(x, a, \{\mathrm{com}_i\}_{i \in C})$
4. Output $(a, c, z_c, \{\mathrm{com}_i\}_{i \in C})$

In a Bit More Detail: Unruh

Getting around rewinding and adaptive reprogramming [U15].

Rough Idea: Let Prover commit to all (challenge, response) pair.

1. a
2. For $i \in C$
   i. Generate response z_i
   ii. $\mathrm{com_i} = \mathrm{Com}(z_i; \mathrm{rand_i})$
3. $c \leftarrow H(x, a, \{\mathrm{com}_i\}_{i \in C})$
4. Output $(a, c, z_c, \{\mathrm{com}_i\}_{i \in C})$

Simplified Proof for PoK:

1. The cheating prover must have committed to valid responses $z_i, z_{i'}$ for $i \neq i'$ to have non-negl. advantage.
2. If Com is extractable, then the reduction inverts $\mathrm{com}_i, \mathrm{com}_{i'}$.

In a Bit More Detail: Unruh

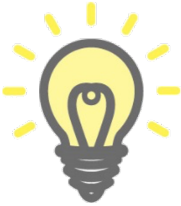
Getting around rewinding and adaptive reprogramming [U15].

Rough Idea: Let Prover commit to all (challenge, response) pair.

1. a
2. For $i \in C$
   i. Generate response z_i
   ii. $\mathrm{com_i} = \mathrm{Com}(z_i; \mathrm{rand_i})$
3. $c \leftarrow H(x, a, \{\mathrm{com}_i\}_{i \in C})$
4. Output $(a, c, z_c, \{\mathrm{com}_i\}_{i \in C})$

Simplified Proof for PoK:

1. The cheating prover must have committed to valid responses $z_i, z_{i'}$ for $i \neq i'$ to have non-negl. advantage.
2. If Com is extractable, then the reduction inverts $\mathrm{com}_i, \mathrm{com}_{i'}$.



- ☐ Challenge space must be poly-large => Parallel repetition.
- ☐ Must include extra $\{\mathrm{com}_i\}_{i \in C}$ in proof.
- ☐ Com can be instantiated by RO, Online Extractable => Tight Proof

Recent CROM Lattice-based PCIP

Non-exhaustive list

- [Lyu09,Lyu12]: Most Basic (Relaxed proof for SIS/LWE relation)
  - [DFMS19,LZ19] showed that it is "collapsing" (w/ a slight increase in the parameters).

QROM secure via Fiat-Shamir

- [BDLOP18]: Opening to commitments
- [ESLL19]: Range proofs, one-out-of-many proofs
- [YAZXYW19]: Exact sound proofs for quadratic relations

QROM secure via Unruh but chall. set is restricted to be small

- [BLS19, ENS20]: Exact sound proofs for SIS/LWE relation (5-round)
- [ALS20]: Product proofs for commitments (5-round)
- [LNS20]: Integer relations (5≥-round)

*5-round protocols *may* be secure via modified Unruh [CHRSS18].

Main Question of This Talk

Can we get the best of the Fiat-Shamir and Unruh transform and more??

Transform	Quantum		
	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir '88	collapsing ⚠	One hash	⚠ rewind (lose at least $O(q)^{2n}$)
Fischlin '05			
Unruh '15	5-round with ⚠ 1st chall. set small and 2nd chall. set $\{0,1\}$.	⚠ x$\lvert C\rvert$ with parallel rep.	Straight-line (= tight proof)

- ✓ FS: No overhead and works for exp. large chall. set size.
- ✓ Unruh: Tight (straight-line extractable) and simple proof.
- ✓ And More: Applies to PCIP that FS or Unruh is not known to apply.

3. Our Result: ExtLinHC

Our Result: A New Transform

A partial answer:

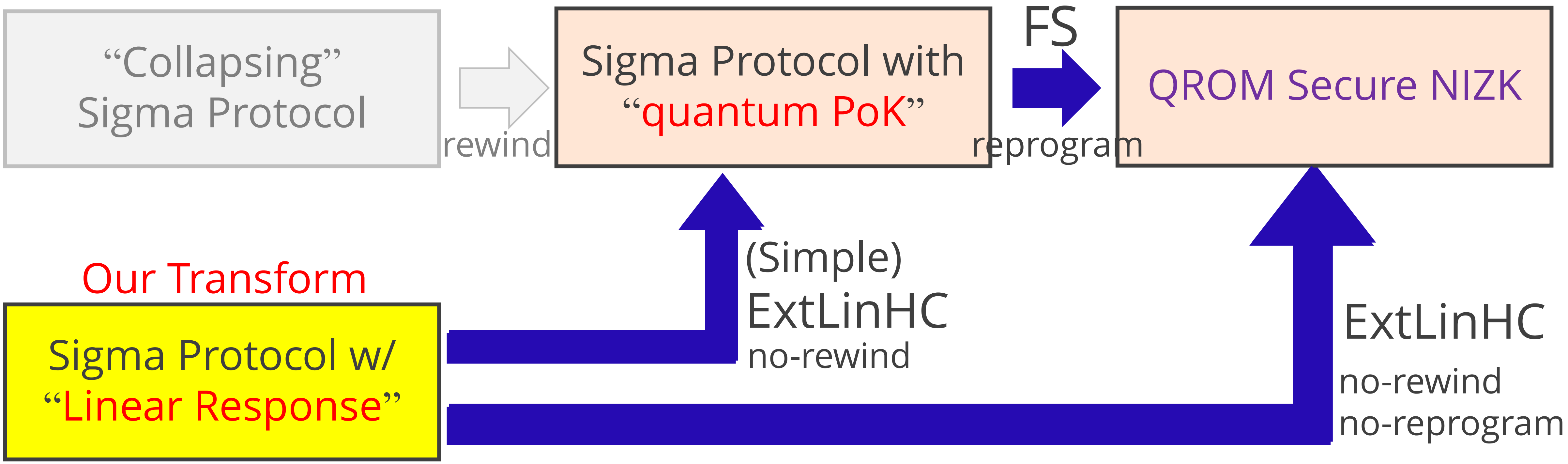
A semi-generic approach that sits somewhere between Fiat-Shamir and Unruh.

Properties

- Works for many lattice-based PCIPs (or in general, **any PCIP with a linear response**)
- Handles **exponential challenge set**
- **|FS overhead| < |Our overhead| < |Unruh overhead|**, for exp. chall. set.
- Reduction loss is smaller than FS (it is **straight-line extractable** like Unurh)
- **Construction and proof is very simple** (almost classical)

New Technical Tool

Extractable Linear Homomorphic Commitment (ExtLinHC)



*Very natural and satisfied by many Sigma protocols [M15]

A Bottom-Up Approach to Our Transform

Base Example: **Sigma protocol for SIS/LWE relation** [Lyu09,12]

Statement: $(\mathrm{A}, \mathrm{u}) \in R_q^{n \times m} \times R_q^n$
Witness: "short" $\mathrm{e} \in R_q^m$

$^*R_q = \mathbb{Z}[X]/(X^d + 1)$

Prover

Verifier

A Bottom-Up Approach to Our Transform

Base Example: **Sigma protocol for SIS/LWE relation** [Lyu09,12]

Statement: $(\mathrm{A}, \mathrm{u}) \in R_q^{n \times m} \times R_q^n$
Witness: "short" $\mathrm{e} \in R_q^m$

$^*R_q = \mathbb{Z}[X]/(X^d + 1)$

Prover	Verifier
1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar} \in R_q^n$

w →

←

→

A Bottom-Up Approach to Our Transform

Base Example: **Sigma protocol for SIS/LWE relation** [Lyu09,12]

Statement: $(\mathrm{A}, \mathrm{u}) \in R_q^{n \times m} \times R_q^n$
Witness: "short" $\mathrm{e} \in R_q^m$

*$R_q = \mathbb{Z}[X]/(X^d + 1)$

Prover		Verifier
1. $\mathrm{r} \leftarrow D^m$		
2. $\mathrm{w} = \mathrm{Ar} \in R_q^n$	w →	
	← c	$c \leftarrow \{0,1\}^d \subset R_q$
3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r} \in R_q^m$	→	
4. RejSamp(z)		

A Bottom-Up Approach to Our Transform

Base Example: **Sigma protocol for SIS/LWE relation** [Lyu09,12]

Statement: $(\mathrm{A}, \mathrm{u}) \in R_q^{n \times m} \times R_q^n$
Witness: "short" $\mathrm{e} \in R_q^m$

*$R_q = \mathbb{Z}[X]/(X^d + 1)$

Prover

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar} \in R_q^n$

w →

← c

Verifier

$c \leftarrow \{0,1\}^d \subset R_q$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r} \in R_q^m$
4. $\mathrm{RejSamp(z)}$

z →

Check
- z is short
- $\mathrm{Az} = \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

Special Sound + HVZK

P: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ V: (A, u)

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$

$\xrightarrow{\mathrm{w}}$

$\xleftarrow{\mathrm{c}}$ $c \leftarrow \{0,1\}^d$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
4. $\mathrm{RejSamp}(\mathrm{z})$

$\xrightarrow{\mathrm{z}}$

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

- Special (Relaxed) Soundness

- HVZK

Special Sound + HVZK

P: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ V: (A, u)

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$

$\xrightarrow{\mathrm{w}}$

$\xleftarrow{\mathrm{c}}$ $c \leftarrow \{0,1\}^d$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
4. $\mathrm{RejSamp}(\mathrm{z})$

$\xrightarrow{\mathrm{z}}$

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

Special (Relaxed) Soundness

Given $(\mathrm{w}, \mathrm{c}, \mathrm{z})$ and $(\mathrm{w}, \mathrm{c}', \mathrm{z}')$

$$\begin{aligned} \mathrm{Az} &= \mathrm{c} \cdot \mathrm{u} + \mathrm{w} \\ \mathrm{Az}' &= \mathrm{c}' \cdot \mathrm{u} + \mathrm{w} \end{aligned} \quad \Rightarrow \quad \mathrm{A}\underline{(\mathrm{z} - \mathrm{z}')} = \underline{(\mathrm{c} - \mathrm{c}')} \cdot \mathrm{u}$$

*The extracted witness lies in a "gap/relaxed" relation.
But this suffices in many applications.

HVZK

Special Sound + HVZK

P: $((\mathrm{A},\mathrm{u}),\mathrm{e})$ V: (A,u)

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$ → w

← c $c \leftarrow \{0,1\}^d$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
4. $\mathrm{RejSamp(z)}$ → z

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

Special (Relaxed) Soundness

Given $(\mathrm{w},\mathrm{c},\mathrm{z})$ and $(\mathrm{w},\mathrm{c}',\mathrm{z}')$

$$\begin{aligned}\mathrm{Az} &= \mathrm{c} \cdot \mathrm{u} + \mathrm{w} \\ \mathrm{Az}' &= \mathrm{c}' \cdot \mathrm{u} + \mathrm{w}\end{aligned} \Rightarrow \mathrm{A}\underline{(\mathrm{z} - \mathrm{z}')} = \underline{(\mathrm{c} - \mathrm{c}')} \cdot \mathrm{u}$$

*The extracted witness lies in a "gap/relaxed" relation. But this suffices in many applications.

HVZK

- Due to RejSamp, z is uniform over some witness-independent dist. G'
- ZKSim just samples z and sets $\mathrm{w} = \mathrm{Az} - \mathrm{c} \cdot \mathrm{u}$.

Special Sound + HVZK

P: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ V: (A, u)

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$ $\xrightarrow{\mathrm{w}}$

$\xleftarrow{\mathrm{c}}$ $c \leftarrow \{0,1\}^d$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
4. $\mathrm{RejSamp}(\mathrm{z})$ $\xrightarrow{\mathrm{z}}$

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

Special (Relaxed) Soundness

Given $(\mathrm{w}, \mathrm{c}, \mathrm{z})$ and $(\mathrm{w}, \mathrm{c}', \mathrm{z}')$

$$\begin{aligned} \mathrm{Az} &= \mathrm{c} \cdot \mathrm{u} + \mathrm{w} \\ \mathrm{Az}' &= \mathrm{c}' \cdot \mathrm{u} + \mathrm{w} \end{aligned} \Rightarrow \mathrm{A}(\underline{\mathrm{z} - \mathrm{z}'}) = (\underline{\mathrm{c} - \mathrm{c}'}) \cdot \mathrm{u}$$

*The extracted witness lies in a "gap/relaxed" relation.
But this suffices in many applications.

HVZK

- Due to RejSar
- ZKSim just sa

Main Question

How do we obtain *two* valid transcripts *w/o* rewinding the quantum adversary??

1st Step: Add Linear Homomorphic Com.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$

CRS: $pk \leftarrow \{0,1\}^L$

Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D}^{\mathrm{m}}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_\mathrm{e}]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_\mathrm{r}]$

$\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}$ →

*Commit to witness e and randomness r

1^{st} Step: Add Linear Homomorphic Com.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$

CRS: $pk \leftarrow \{0,1\}^L$

Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D}^{\mathrm{m}}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_\mathrm{e}]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_\mathrm{r}]$

$\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}$ →

$c \leftarrow \{0,1\}^d$

← c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_\mathrm{z} = \mathrm{c} \cdot \delta_\mathrm{e} + \delta_\mathrm{r}$
7. $\mathrm{RejSamp(z)}$

1^{st} Step: Add Linear Homomorphic Com.

CRS: $pk \leftarrow \{0,1\}^L$

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$

Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D}^{\mathrm{m}}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_\mathrm{e}]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_\mathrm{r}]$

Prover → Verifier: $\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_\mathrm{z} = \mathrm{c} \cdot \delta_\mathrm{e} + \delta_\mathrm{r}$
7. $\mathrm{RejSamp(z)}$

Prover → Verifier: z, δ_z

Verifier:

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$
- $\mathrm{Com_{pk}(z)}[\delta_\mathrm{z}] \stackrel{?}{=} \mathrm{c} \cdot \mathrm{com_e} + \mathrm{com_r}$

*Create $\mathrm{com_z}$ and check if δ_z is a valid opening.

1st Step: Add Linear Homomorphic Com.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$

CRS: $pk \leftarrow \{0,1\}^L$

Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D}^{\mathrm{m}}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_\mathrm{e}]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_\mathrm{r}]$

$\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}$ →

$c \leftarrow \{0,1\}^d$

← c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_\mathrm{z} = \mathrm{c} \cdot \delta_\mathrm{e} + \delta_\mathrm{r}$
7. $\mathrm{RejSamp(z)}$

$\mathrm{z}, \delta_\mathrm{z}$ →

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$
- $\mathrm{Com_{pk}(z)}[\delta_\mathrm{z}] \stackrel{?}{=} \mathrm{c} \cdot \mathrm{com_e} + \mathrm{com_r}$

Is it still a standard Sigma protocol?

✓ Special soundness => Yes, just ignore LinHC

✓ HVZK => Yes, if LinCH is hiding.

2^{nd} Step: Add Extractability

$$\mathrm{pk} \leftarrow \{0,1\}^L$$
$$\approx_c$$
$$(\mathrm{pk}^*, \tau) \leftarrow \mathrm{SimCRS}$$

For any honestly generated $\mathrm{com_x} = \mathrm{Com_{pk^*}(x)[\delta_x]}$, we have $\mathrm{Extract_{Com}(\tau, com_x) \to x}$.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ | CRS: $pk \leftarrow \{0,1\}^L$ | Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D^m}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)[\delta_e]}$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)[\delta_r]}$

Prover → Verifier: $\mathrm{w, com_e, com_r}$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_z = \mathrm{c} \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp(z)}$

Prover → Verifier: z, δ_z

Verifier:
- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$
- $\mathrm{Com_{pk}(z)[\delta_z]} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{com_e} + \mathrm{com_r}$

2^{nd} Step: Add Extractability

$$\mathrm{pk} \leftarrow \{0,1\}^L$$
$$\approx_c$$
$$(\mathrm{pk}^*, \tau) \leftarrow \mathrm{SimCRS}$$

For any honestly generated $\mathrm{com_x} = \mathrm{Com_{pk^*}(x)[\delta_x]}$, we have $\mathrm{Extract_{Com}(\tau, com_x) \rightarrow x}$.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ | CRS: $pk \leftarrow \{0,1\}^L$ | Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D^m}$
2. $\mathrm{w = Ar}$
3. $\mathrm{com_e = Com_{pk}(e)[\delta_e]}$
4. $\mathrm{com_r = Com_{pk}(r)[\delta_r]}$

Prover → Verifier: $\mathrm{w, com_e, com_r}$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $\mathrm{z = c \cdot e + r}$
6. $\mathrm{\delta_z = c \cdot \delta_e + \delta_r}$
7. $\mathrm{RejSamp(z)}$

Prover → Verifier: $\mathrm{z}, \delta_\mathrm{z}$

Verifier:
- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c \cdot u + w}$
- $\mathrm{Com_{pk}(z)[\delta_z]} \stackrel{?}{=} \mathrm{c \cdot com_e + com_r}$

What we want to show

Only given $((\mathrm{w, com_e, com_r}), \mathrm{c}, (\mathrm{z}, \delta_\mathrm{z}))$, extract witness e in the "gap" relation.

2nd Step: Add Extractability

$$\mathrm{pk} \leftarrow \{0,1\}^L$$
$$\approx_c$$
$$(\mathrm{pk}^*, \tau) \leftarrow \mathrm{SimCRS}$$

For any honestly generated $\mathrm{com_x} = \mathrm{Com_{pk^*}(x)[\delta_x]}$, we have $\mathrm{Extract_{Com}}(\tau, \mathrm{com_x}) \to \mathrm{x}$.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ | CRS: $pk \leftarrow \{0,1\}^L$ | Verifier: (A, u)

1. $\mathrm{r} \leftarrow \mathrm{D^m}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)[\delta_e]}$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)[\delta_r]}$

Prover → Verifier: $\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_\mathrm{z} = \mathrm{c} \cdot \delta_\mathrm{e} + \delta_\mathrm{r}$
7. $\mathrm{RejSamp(z)}$

Prover → Verifier: z, δ_z

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$
- $\mathrm{Com_{pk}(z)[\delta_z]} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{com_e} + \mathrm{com_r}$

What we want to show

Only given $((\mathrm{w}, \mathrm{com_e}, \mathrm{com_r}), \mathrm{c}, (\mathrm{z}, \delta_\mathrm{z}))$, extract witness e in the "gap" relation.

Incorrect Naïve Argument

Just run $\mathrm{Extract_{Com}}(\tau, \mathrm{com_e}) \to \mathrm{e}$!

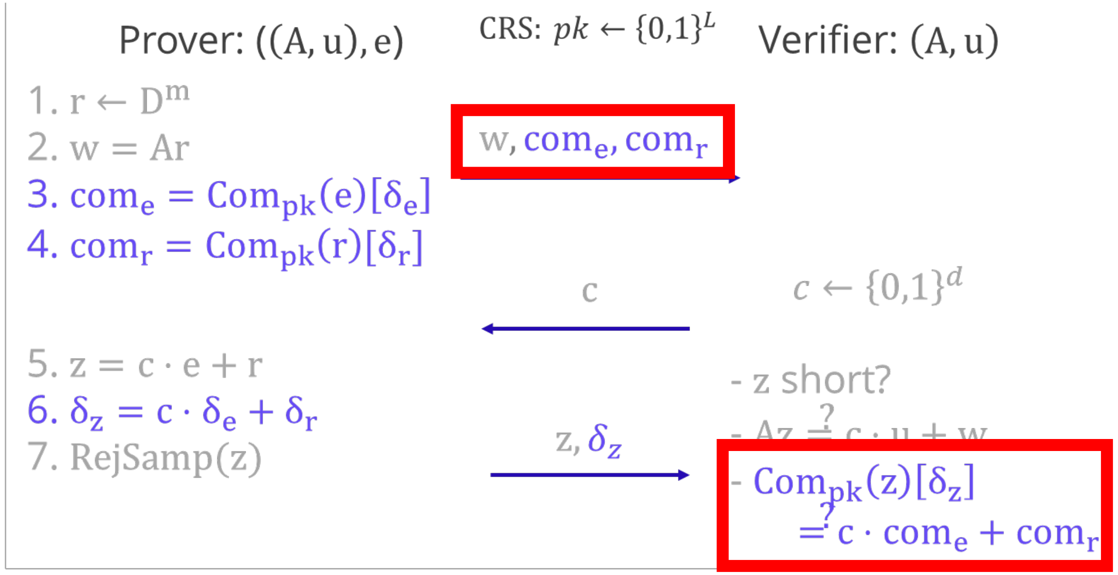
Why Wrong?

- No guarantee that $\mathrm{com_e}$ is valid ☹
- Only $\mathrm{Com_{pk}(z)[\delta_z]}$ is known to be valid.

3^{rd} Step: How to Argue Extraction Correctly

Simple Observation

$(\mathrm{com_e}, \mathrm{com_r})$ is prepared <u>before</u> challenge c.



3rd Step: How to Argue Extraction Correctly

Simple Observation

$(\mathrm{com_e}, \mathrm{com_r})$ is prepared <u>before</u> challenge c.

- Assume $|2^d| = \mathrm{poly}(\lambda)$.
- Assume another $(c', z', \delta_{z'})$ s.t. V accepts.

$\mathrm{Extract_{Sigma}}(\tau, \mathrm{trans})$:

For $i \in \{0,1\}^d$
1. Set $\mathrm{com}_{z_i} := i \cdot \mathrm{com_e} + \mathrm{com_r}$
2. Try $\mathrm{Extract_{com}}(\tau, \mathrm{com}_{z_i}) \to z_i / \perp$

Prover: $((A, u), e)$ — CRS: $pk \leftarrow \{0,1\}^L$ — Verifier: (A, u)

1. $r \leftarrow D^m$
2. $w = Ar$
3. $\mathrm{com_e} = \mathrm{Com_{pk}}(e)[\delta_e]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}}(r)[\delta_r]$

Prover → Verifier: $w, \mathrm{com_e}, \mathrm{com_r}$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $z = c \cdot e + r$
6. $\delta_z = c \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp}(z)$

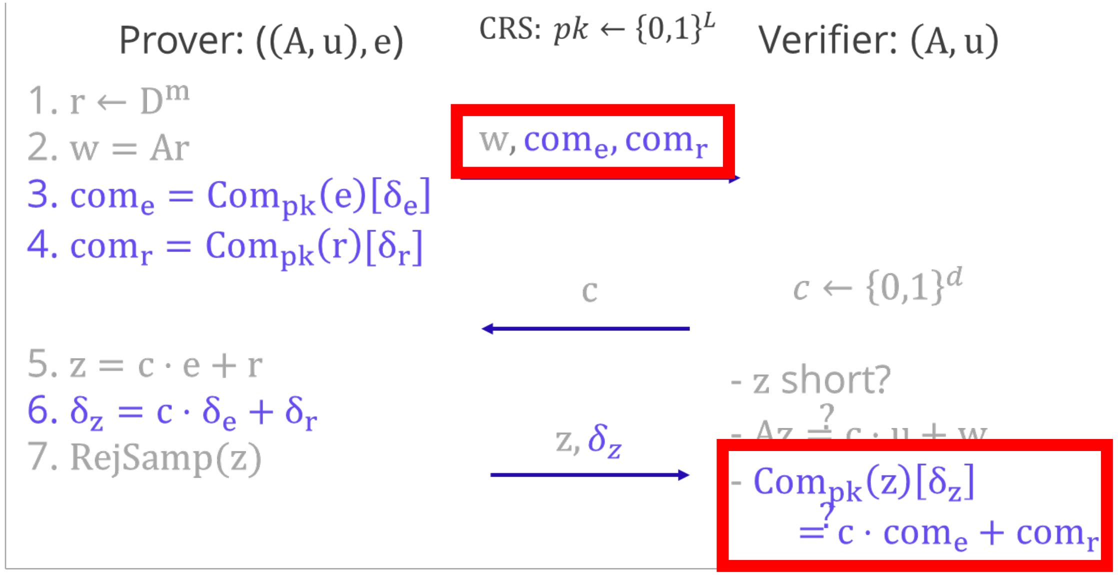
Prover → Verifier: z, δ_z

Verifier checks:
- z short?
- $Az \stackrel{?}{=} c \cdot u + w$
- $\mathrm{Com_{pk}}(z)[\delta_z] \stackrel{?}{=} c \cdot \mathrm{com_e} + \mathrm{com_r}$

3^{rd} Step: How to Argue Extraction Correctly

Simple Observation

$(\mathrm{com_e}, \mathrm{com_r})$ is prepared <u>before</u> challenge c.



- Assume $|2^d| = \mathrm{poly}(\lambda)$.
- Assume another $(c', z', \delta_{z'})$ s.t. V accepts.

$\mathrm{Extract_{Sigma}}(\tau, \mathrm{trans})$:

For $\mathrm{i} \in \{0,1\}^d$
1. Set $\mathrm{com}_{z_i} := \mathrm{i} \cdot \mathrm{com_e} + \mathrm{com_r}$
2. Try $\mathrm{Extract_{com}}(\tau, \mathrm{com}_{z_i}) \rightarrow z_i/\perp$

➡ By assumption, if $\mathrm{i} = \mathrm{c}'$, then $\mathrm{Extract_{com}}$ succeeds since $\mathrm{com}_{z_{c'}} = \mathrm{Com_{pk^*}}(z')[\delta_{z'}]$ is guaranteed to be a valid commitment.

3rd Step: How to Argue Extraction Correctly

Simple Observation

$(\mathrm{com}_e, \mathrm{com}_r)$ is prepared <u>before</u> challenge c.

- Assume $|2^d| = \mathrm{poly}(\lambda)$.
- Assume another $(c', z', \delta_{z'})$ s.t. V accepts.

Prover: $((A, u), e)$ CRS: $pk \leftarrow \{0,1\}^L$ Verifier: (A, u)

1. $r \leftarrow D^m$
2. $w = Ar$
3. $\mathrm{com}_e = \mathrm{Com}_{pk}(e)[\delta_e]$
4. $\mathrm{com}_r = \mathrm{Com}_{pk}(r)[\delta_r]$

→ $w, \mathrm{com}_e, \mathrm{com}_r$

← c $\qquad c \leftarrow \{0,1\}^d$

5. $z = c \cdot e + r$
6. $\delta_z = c \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp}(z)$

→ z, δ_z

- z short?
- $Az \stackrel{?}{=} c \cdot u + w$
- $\mathrm{Com}_{pk}(z)[\delta_z] \stackrel{?}{=} c \cdot \mathrm{com}_e + \mathrm{com}_r$

$\mathrm{Extract}_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

For $i \in \{0,1\}^d$
1. Set $\mathrm{com}_{z_i} := i \cdot \mathrm{com}_e + \mathrm{com}_r$
2. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{z_i}) \rightarrow z_i/\bot$

After extracting z', simply use (w, c, c', z, z') to extract witness e ☺

➡ By assumption, if $i = c'$, then $\mathrm{Extract}_{\mathrm{com}}$ succeeds since $\mathrm{com}_{z_{c'}} = \mathrm{Com}_{pk^*}(z')[\delta_{z'}]$ is guaranteed to be a valid commitment.

4^{th} Step: Making Chall. Set Exponentially Large

$\mathrm{Extract}_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

For $i \in \{0,1\}^d$
1. Set $\mathrm{com}_{z_i} := i \cdot \mathrm{com}_e + \mathrm{com}_r$
2. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{z_i}) \rightarrow z_i / \perp$

Only terminates if 2^d is polynomial…

Prover: $((A, u), e)$ — CRS: $pk \leftarrow \{0,1\}^L$ — Verifier: (A, u)

1. $r \leftarrow D^m$
2. $w = Ar$
3. $\mathrm{com}_e = \mathrm{Com}_{pk}(e)[\delta_e]$
4. $\mathrm{com}_r = \mathrm{Com}_{pk}(r)[\delta_r]$

Prover → Verifier: $w, \mathrm{com}_e, \mathrm{com}_r$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $z = c \cdot e + r$
6. $\delta_z = c \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp}(z)$

Prover → Verifier: z, δ_z

Verifier:
- z short?
- $Az \stackrel{?}{=} c \cdot u + w$
- $\mathrm{Com}_{pk}(z)[\delta_z] \stackrel{?}{=} c \cdot \mathrm{com}_e + \mathrm{com}_r$

4^{th} Step: Making Chall. Set Exponentially Large

$\mathrm{Extract}_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

For $i \in \{0,1\}^d$
1. Set $\mathrm{com}_{z_i} := i \cdot \mathrm{com}_e + \mathrm{com}_r$
2. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{z_i}) \to z_i / \perp$

Only terminates if 2^d is polynomial…

Prover: $((A, u), e)$ — CRS: $pk \leftarrow \{0,1\}^L$ — Verifier: (A, u)

1. $r \leftarrow D^m$
2. $w = Ar$
3. $\mathrm{com}_e = \mathrm{Com}_{pk}(e)[\delta_e]$
4. $\mathrm{com}_r = \mathrm{Com}_{pk}(r)[\delta_r]$

Prover → Verifier: $w, \mathrm{com}_e, \mathrm{com}_r$

Verifier: $c \leftarrow \{0,1\}^d$

Verifier → Prover: c

5. $z = c \cdot e + r$
6. $\delta_z = c \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp}(z)$

Prover → Verifier: z, δ_z

Verifier:
- z short?
- $Az \stackrel{?}{=} c \cdot u + w$
- $\mathrm{Com}_{pk}(z)[\delta_z] \stackrel{?}{=} c \cdot \mathrm{com}_e + \mathrm{com}_r$

$\mathrm{New} - \mathrm{Extract}_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

While $t < N$:
1. $i \leftarrow \{0,1\}^d$
2. Set $\mathrm{com}_{z_i} := i \cdot \mathrm{com}_e + \mathrm{com}_r$
3. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{z_i}) \to z_i / \perp$
4. $t \leftarrow t + 1$

Run for at most N times until $\mathrm{Extract}_{\mathrm{com}}$ succeeds.

Why should this work? How do we set N?

4^{th} Step: Making Chall. Set Exponentially Large

Why it works

Assume adversary A has non-negl adv. ϵ in completing the Sigma protocol.

$\mathrm{New\text{-}Extract}_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

While $\mathrm{t} < \mathrm{N}$:

1. $\mathrm{i} \leftarrow \{0,1\}^{\mathrm{d}}$
2. Set $\mathrm{com}_{\mathrm{z}_{\mathrm{i}}} \coloneqq \mathrm{i} \cdot \mathrm{com}_{\mathrm{e}} + \mathrm{com}_{\mathrm{r}}$
3. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{\mathrm{z}_{\mathrm{i}}}) \rightarrow \mathrm{z}_{\mathrm{i}}/\perp$
4. $\mathrm{t} \leftarrow \mathrm{t} + 1$

4^{th} Step: Making Chall. Set Exponentially Large

New-Extract$_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

While $\mathrm{t} < \mathrm{N}$:

1. $\mathrm{i} \leftarrow \{0,1\}^{\mathrm{d}}$
2. Set $\mathrm{com}_{\mathrm{z_i}} \coloneqq \mathrm{i} \cdot \mathrm{com}_{\mathrm{e}} + \mathrm{com}_{\mathrm{r}}$
3. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{\mathrm{z_i}}) \rightarrow \mathrm{z_i}/\bot$
4. $\mathrm{t} \leftarrow \mathrm{t} + 1$

Why it works

Assume adversary A has non-negl adv. ϵ in completing the Sigma protocol.

Then, there exists at least $2^d \cdot \epsilon$ challenges for which A could have correctly respond w/ prob. 1/2.

*Standard *statistical* argument.

4^{th} Step: Making Chall. Set Exponentially Large

New-Extract$_{\mathrm{Sigma}}(\tau, \mathrm{trans})$:

While $\mathrm{t} < \mathrm{N}$:
1. $\mathrm{i} \leftarrow \{0,1\}^{\mathrm{d}}$
2. Set $\mathrm{com}_{\mathrm{z}_{\mathrm{i}}} \coloneqq \mathrm{i} \cdot \mathrm{com}_{\mathrm{e}} + \mathrm{com}_{\mathrm{r}}$
3. Try $\mathrm{Extract}_{\mathrm{com}}(\tau, \mathrm{com}_{\mathrm{z}_{\mathrm{i}}}) \rightarrow \mathrm{z}_{\mathrm{i}}/\perp$
4. $\mathrm{t} \leftarrow \mathrm{t} + 1$

Why it works

Assume adversary A has non-negl adv. ϵ in completing the Sigma protocol.

Then, there exists at least $2^{d} \cdot \epsilon$ challenges for which A could have correctly respond w/ prob. 1/2.

*Standard *statistical* argument.

If $\mathrm{New} - \mathrm{Extract}_{\mathrm{Sigma}}$ samples $\mathrm{N} = O(\frac{\lambda}{\epsilon})$-random challenge i, then it will hit a valid commitment com_{z_i} with o.w.p.

4^{th} Step: Making Chall. Set Exponentially Large

Why it works

Assume adversary A has non-negl adv. ϵ in completing the Sigma protocol.

Then, there exists at least $2^d \cdot \epsilon$ challenges for which A could have correctly respond w/ prob. 1/2.

*Standard *statistical* argument.

If $\mathrm{New-Extract_{Sigma}}$ samples $\mathrm{N} = O(\frac{\lambda}{\epsilon})$-random challenge i, then it will hit a valid commitment com_{z_i} with o.w.p.

$\mathrm{New\text{-}Extract_{Sigma}}(\tau, \mathrm{trans})$:

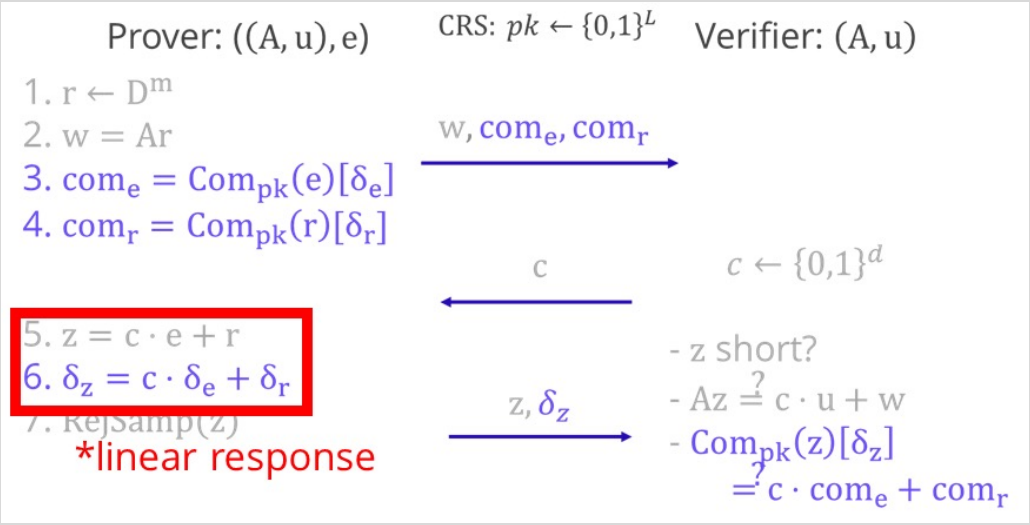
While $\mathrm{t} < \mathrm{N}$:

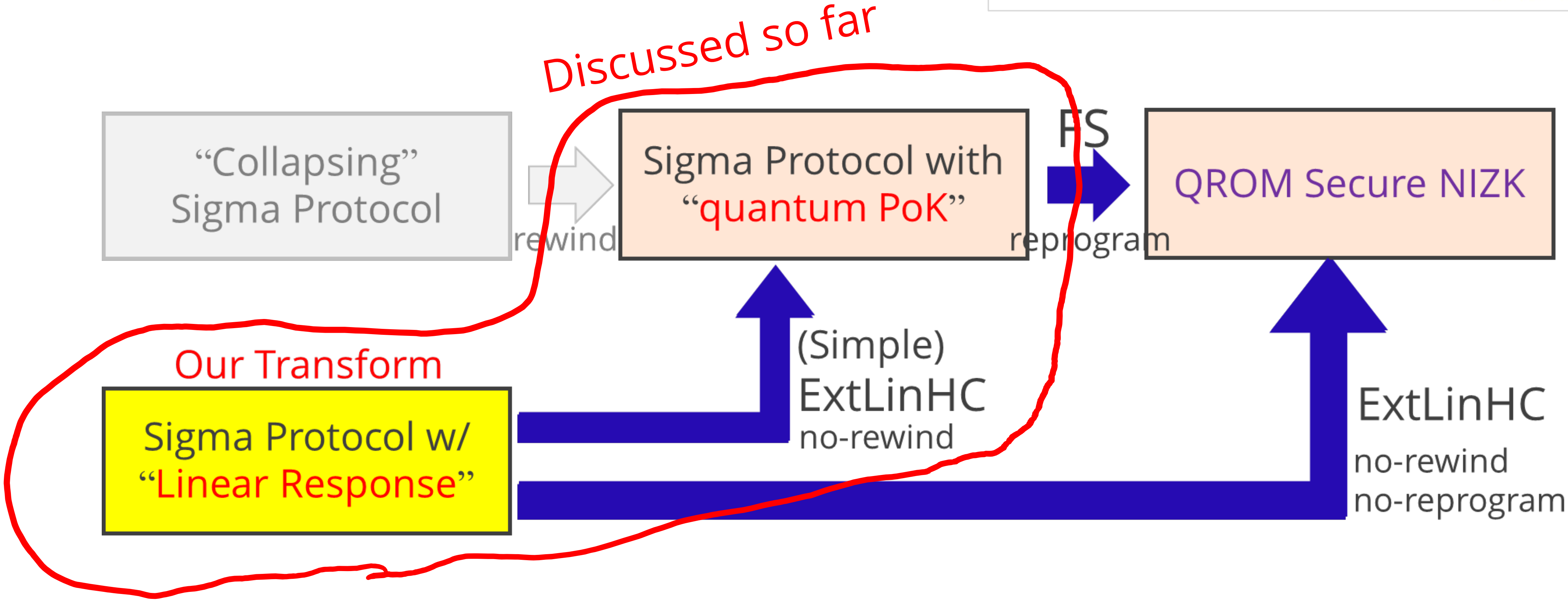
1. $\mathrm{i} \leftarrow \{0,1\}^{\mathrm{d}}$
2. Set $\mathrm{com}_{\mathrm{z_i}} := \mathrm{i} \cdot \mathrm{com_e} + \mathrm{com_r}$
3. Try $\mathrm{Extract_{com}}(\tau, \mathrm{com}_{\mathrm{z_i}}) \rightarrow \mathrm{z_i} / \perp$
4. $\mathrm{t} \leftarrow \mathrm{t} + 1$

Analysis is the same even if A is quantum ☺

Summary So Far

- PoK of Sigma protocol can be shown w/o rewinding the adversary ☺

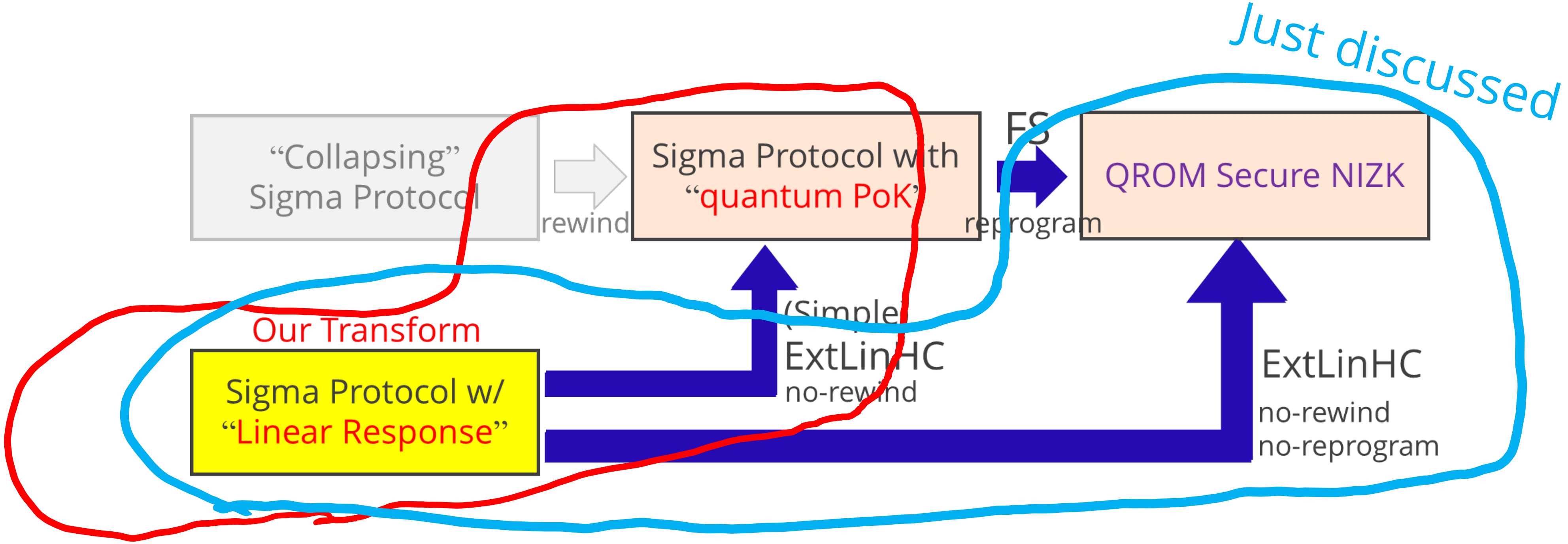




Extending to QROM-Secure NIZK

We start with **a Sigma protocol w/ quantum "straight-line"** PoK.

We can make it non-interactive via **Fiat-Shamir with a simpler proof** (i.e., **no-reprogramming**) akin to [U15,KLS18] ☺



4. Constructing ExtLinHC

Lattice-based ExtLinHC

Com. Key: $\mathrm{pk} = (\mathrm{A}, \mathrm{B}) \leftarrow R_q^{m\times n} \times R_q^{m\times n}$

$p < q$: some large enough integer

$$\mathrm{com_e} := \left(p\cdot\left(\mathrm{As}_{e,1} + \mathrm{s}_{e,2}\right), p\cdot\left(\mathrm{Bs}_{e,1} + \mathrm{s}_{e,3}\right) + \mathbf{e}\right)$$

*witness

$$\mathrm{com_r} := \left(p\cdot\left(\mathrm{As}_{r,1} + \mathrm{s}_{r,2}\right), p\cdot\left(\mathrm{Bs}_{r,1} + \mathrm{s}_{z,3}\right) + \mathbf{r}\right)$$

*randomness

where $\delta_e = \left(\mathrm{s}_{e,i}\right)_{i\in[3]}, \delta_r = \left(\mathrm{s}_{r,i}\right)_{i\in[3]}$.

Prover: $((\mathrm{A},\mathrm{u}),\mathrm{e})$

CRS: pk

1. $\mathrm{r} \leftarrow \mathrm{D^m}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_e]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_r]$

$\mathrm{w}, \mathrm{com_e}$

c

5. $\mathrm{z} = \mathrm{c}\cdot\mathrm{e} + \mathrm{r}$
6. $\delta_z = \mathrm{c}\cdot\delta_e + \delta_r$
7. RejSamp(z)

z, δ

Lattice-based ExtLinHC

Com. Key: $\mathrm{pk} = (\mathrm{A}, \mathrm{B}) \leftarrow R_q^{m\times n} \times R_q^{m\times n}$

$p < q$: some large enough integer

$$\mathrm{com_e} \coloneqq (p \cdot (\mathrm{A}s_{e,1} + s_{e,2}), p \cdot (\mathrm{B}s_{e,1} + s_{e,3}) + \mathbf{e})$$

*witness

$$\mathrm{com_r} \coloneqq (p \cdot (\mathrm{A}s_{r,1} + s_{r,2}), p \cdot (\mathrm{B}s_{r,1} + s_{z,3}) + \mathbf{r})$$

*randomness

where $\delta_e = (s_{e,i})_{i\in[3]}, \delta_r = (s_{r,i})_{i\in[3]}$.

Linear homomorphism

$$\mathrm{com_z} \coloneqq (p \cdot (\mathrm{A}s_{z,1} + s_{z,2}), p \cdot (\mathrm{B}s_{z,1} + s_{z,3}) + \mathrm{c} \cdot \mathbf{e} + \mathbf{r})$$

where $\delta_z = (s_{z,i} = \mathrm{c} \cdot s_{e,i} + s_{r,i})_{i\in[3]}$.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$ CRS: pk

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_e]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_r]$

$\mathrm{w}, \mathrm{com_e}$

c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_z = \mathrm{c} \cdot \delta_e + \delta_r$
7. RejSamp(z)

z, δ

Extraction Mode: Dual Regev PKE

Sim Com. Key: $\mathrm{pk} = (\mathrm{A}, \mathrm{B}) = (\mathrm{A}, \mathrm{D_1 A} + \mathrm{D_2})$

$\tau = \text{"small"}\ \mathrm{D_1}, \mathrm{D_2}$

$$\mathrm{com_x} \coloneqq (\mathrm{t_1}, \mathrm{t_2}) \coloneqq \left(p \cdot \left(\mathrm{A s_{x,1}} + \mathrm{s_{x,2}}\right), p \cdot \left(\mathrm{B s_{x,1}} + \mathrm{s_{x,3}}\right) + \mathrm{x}\right)$$

$\mathrm{Extract_{com}}(\tau, \mathrm{com_x})$:

Output $(t_2 - D_1 t_1) \bmod q \bmod p$

$$= (p \cdot \text{"noise"} + x) \bmod q \bmod p$$
$$= (p \cdot \text{"noise"} + x) \bmod p$$
$$= x$$

Com. keys are indistinguishable due to LWE.

Prover: $((\mathrm{A}, \mathrm{u}), \mathrm{e})$

CRS: pk

1. $\mathrm{r} \leftarrow \mathrm{D^m}$
2. $\mathrm{w} = \mathrm{Ar}$
3. $\mathrm{com_e} = \mathrm{Com_{pk}(e)}[\delta_\mathrm{e}]$
4. $\mathrm{com_r} = \mathrm{Com_{pk}(r)}[\delta_\mathrm{r}]$

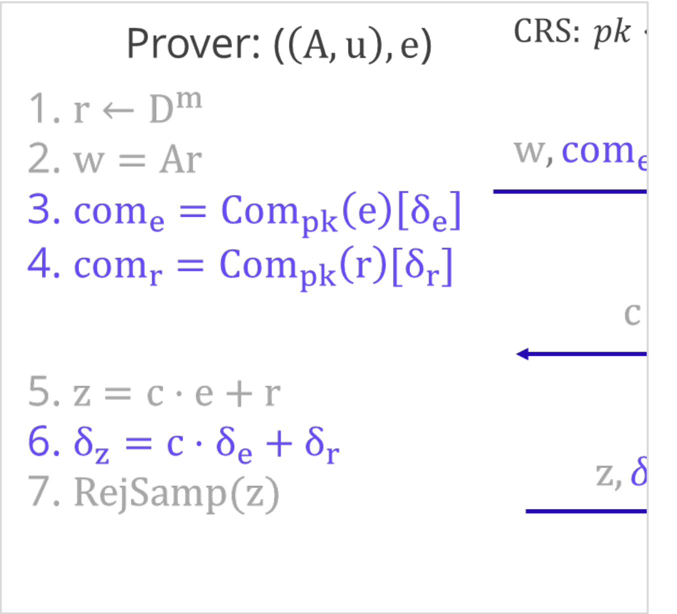
$\mathrm{w}, \mathrm{com_e}$

c

5. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
6. $\delta_\mathrm{z} = \mathrm{c} \cdot \delta_\mathrm{e} + \delta_\mathrm{r}$
7. $\mathrm{RejSamp(z)}$

z, δ

Extraction Mode: Dual Regev PKE



Sim Com. Key: $pk = (A, B) = (A, D_1 A + D_2)$

$\tau = \text{"small" } D_1, D_2$

$$com_x := (t_1, t_2)$$
$$:= (p \cdot (As_{x,1} + s_{x,2}), p \cdot (Bs_{x,1} + s_{x,3}) + x)$$

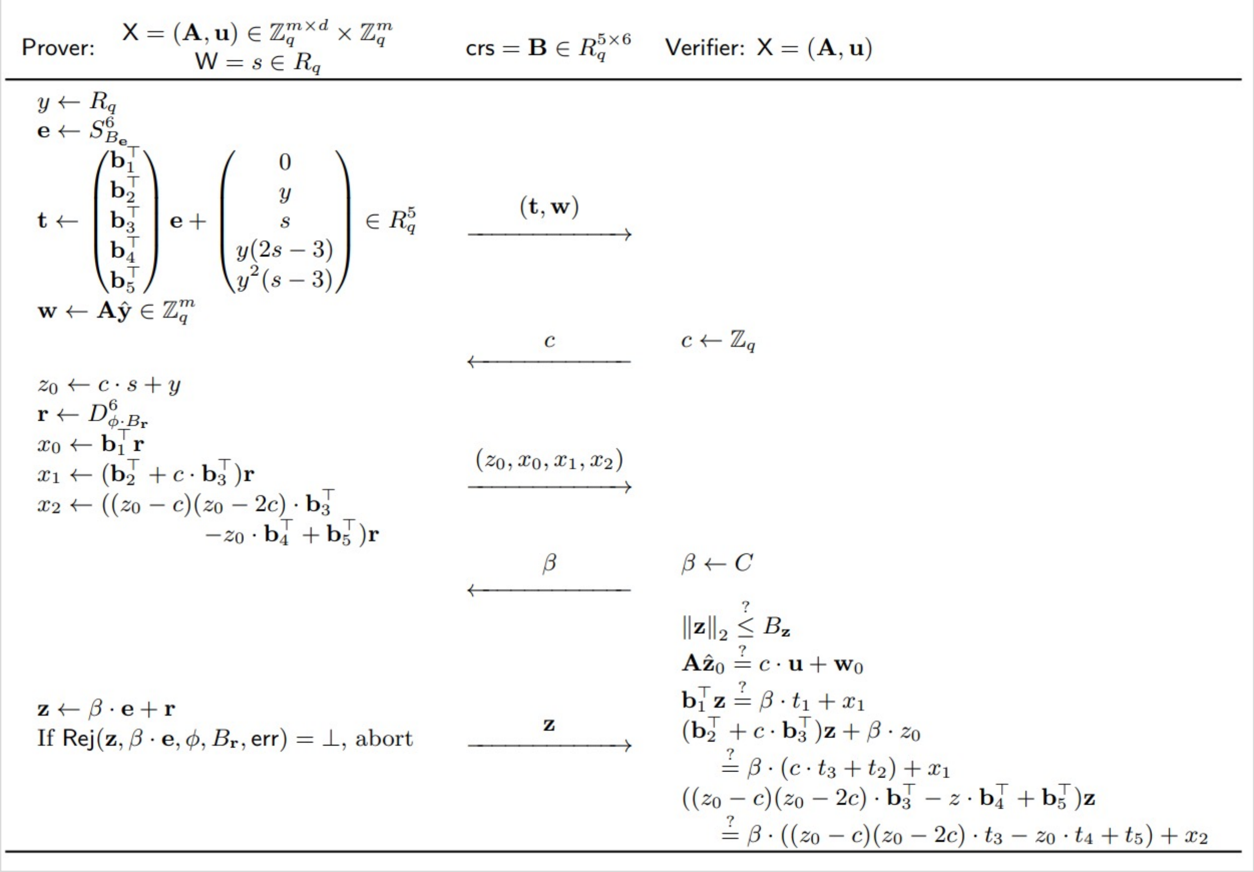
$Extract_{com}(\tau, com_x)$:

Output $(t_2 - D_1 t$

For concrete efficiency, we can optimize the scheme by using NTRU-like PKE.

Com. keys are indistinguishable due to LWE.

Concrete Application



[BLS19] Exact Sound 5-Round PCIP

- Not obvious if Fiat-Shamir applies.
- Modified Unruh [CHRSS18] may apply.

- CROM NIZK = 812 KB
- QROM NIZK via Unruh = 44.9 MB (CROM x134.7)
- QROM NIZK via ExtLinHC = 2071 KB (CROM x2.6)

Summary & Open Problems

A simple method to construct QROM secure NIZKs via ExtLinHC

- ➢ Works for many lattice-based PCIPs before early 2020-ish but what about the more recent ones, e.g., [BLNS20,BLNS20,LNS21]?
- ➢ Can we make ExtLinHC more efficient (possibly w/o trapdoor)??
- ➢ General method to show collapsingness of existing lattice-based Sigma protocols?? => No need using ExtLinHC ☺

Special Sound + HVZK

Prover $((\mathrm{A},\mathrm{u}),\mathrm{e})$

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar} \in R_q^n$

Verifier (A,u)

w →

c ←

$c \leftarrow \{0,1\}^d \subset R_q$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r} \in R_q^m$
4. RejSamp(z)

z →

Check
- z is short
- $\mathrm{Az} = \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

P: $((\mathrm{A},\mathrm{u}),\mathrm{e})$

1. $\mathrm{r} \leftarrow D^m$
2. $\mathrm{w} = \mathrm{Ar}$

V: (A,u)

w →

c ←

$c \leftarrow \{0,1\}^d$

3. $\mathrm{z} = \mathrm{c} \cdot \mathrm{e} + \mathrm{r}$
4. RejSamp(z)

z →

- z short?
- $\mathrm{Az} \stackrel{?}{=} \mathrm{c} \cdot \mathrm{u} + \mathrm{w}$

1^{st} Step: Add Linear Homomorphic Com.

Prover: $((A, u), e)$ CRS: $pk \leftarrow \{0,1\}^L$ Verifier: (A, u)

1. $r \leftarrow D^m$
2. $w = Ar$
3. $com_e = Com_{pk}(e)[\delta_e]$
4. $com_r = Com_{pk}(r)[\delta_r]$

w, com_e, com_r →

← c $c \leftarrow \{0,1\}^d$

5. $z = c \cdot e + r$
6. $\delta_z = c \cdot \delta_e + \delta_r$
7. $\mathrm{RejSamp}(z)$

z, δ_z →

- z short?
- $Az \stackrel{?}{=} c \cdot u + w$
- $Com_{pk}(z)[\delta_z] \stackrel{?}{=} c \cdot com_e + com_r$

Is it still a standard Sigma protocol?

✓ Special soundness => Yes, just ignore LinHC
✓ HVZK => Yes, if LinCH is hiding.

*General $(2n + 1)$-Round PCIP

Transform	Classical			Quantum		
	Type of Sigma prot.	Proof overhead	PoK	Type of Sigma prot.	Proof overhead	PoK
Fiat-Shamir ‘88	any	One hash	rewind	collapsing ⚠	One hash	⚠ rewind (lose at least $O(q^{2n})$)
Fischlin ‘05						
Unruh ‘15				5-round with ⚠ 1st chall. set small and 2nd chall. set $\{0,1\}$.	⚠ x$\lvert C \rvert$ with parallel rep.	Straight-line (= tight proof)

[CHRSS18]

Limited to specific Sigma protocols.

Some Details Worth Mentioning

- In our Fiat-Shamir transform, we require a slightly stronger flavor of ExtLinHC since the Sigma protocol is only *comp.* HVZK.

- Analysis extends to multi-round.

- Since commitment key $\mathrm{pk} \leftarrow \{0,1\}^{\mathrm{d}}$, we can use RO rather than relying on a CRS.

- It is a dual-mode NIZK (i.e., depending on pk, it will be stat. ZK or stat. sound).