Formulations and Constructions of Remote State Preparation with Verifiability, with Applications

Jiayu Zhang*[1]

[1]Zhongguancun Laboratory

July 11, 2025

Abstract

Remote state preparation with verifiability (RSPV) is an important quantum cryptographic primitive [29, 57]. In this primitive, a client would like to prepare a quantum state (sampled or chosen from a state family) on the server side, such that ideally the client knows its full description, while the server holds and only holds the state itself. In this work we make several contributions on its formulations, constructions and applications. In more detail:

- We first work on the definitions and abstract properties of the RSPV problem. We select and compare different variants of definitions [13, 29, 57, 6], and study their basic properties (like composability and amplification).
- We also study a closely related question of how to certify the server's operations (instead of solely the states). We introduce a new notion named *remote operator application with verifiability* (ROAV). We compare this notion with related existing definitions [53, 41, 34, 42, 45], study its abstract properties and leave its concrete constructions for further works.
- Building on the abstract properties and existing results [17], we construct a series of new RSPV protocols. Our constructions not only simplify existing results [29] but also cover new state families, for example, states in the form of $\frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle)$. All these constructions rely only on the existence of weak NTCF [18, 5], without additional requirements like the adaptive hardcore bit property [16, 5].
- As a further application, we show that the classical verification of quantum computations (CVQC) problem [2, 39] could be constructed from assumptions on group actions [4]. This is achieved by combining our results on RSPV with group-action-based instantiation of weak NTCF [5], and then with the quantum-gadget-assisted quantum verification protocol [27].

*zhangjy@zgclab.edu.cn

Contents

1 Introduction

1.1 Background

Development of quantum computers [8, 60, 14] leads to demands of various quantum cryptographic protocols. In a typical setting, there are a client and a remote quantum server. The client would like to achieve some quantum tasks, but it does not trust the server; thus the client would like to make use of cryptography to achieve its goal. Famous examples include quantum computation verification [39, 57], multiparty quantum computations [11], etc. In this work, we are interested in a basic and very important primitive called *remote state preparation* (RSP) [13], which we introduce below.

1.1.1 Remote state preparation: an overview

In the RSP problem, ideally, the client would like to prepare a quantum state (sampled or chosen from a state family) on the server side; thus in the end the client knows the description of the state, while the server simply holds the state. The trivial solution is to simply send the quantum state through a quantum channel. RSP asks: how could we achieve this task using cheaper resources (for example, only classical communication), possibly under computational assumptions?

Studies of RSP have a long history [47, 13]. One setting of RSP is the fully honest setting [13]. In this work, we are interested in the cryptographic setting where the server could be malicious. Then a formulation of RSP should at least have a correctness requirement and a security requirement.

The natural correctness requirement for RSP says that when the server is honest, the server gets the state while the client gets the state description. For security, there are different security notions, including blindness (or privacy, secrecy) and verifiability (or soundness) [25, 29, 59]. In this paper we focus on RSP with verifiability (RSPV). In RSPV, intuitively, the client is able to verify that in the case of passing (or called acceptance, non-aborting) the server (approximately) really gets the state, as if it is sent through a quantum channel. A malicious server who attempts to get other information by deviating from the protocol would be caught cheating by the client.

As a natural quantum task, the RSPV problem is interesting on its own. What's more, it has become an important building block in many other quantum cryptographic protocols. For example, [29] first constructs a classical channel cryptography-based RSPV and uses it to achieve classical verification of quantum computations; [6] explores more applications of RSPV; [57] takes the RSPV approach to achieve classical verification of quantum computations with linear total time complexity. Many quantum cryptographic protocols rely on the quantum channel and quantum communication, and an RSPV protocol could serve as a compiler: it allows us to replace these quantum communication steps by other cheaper resources, like the classical communication.

On the one hand, there have been many important and impressive results in this direction; on the other hand, there are also various limitations or subtleties in existing works, including formulations and constructions. Below we discuss existing works in more detail and motivate our results.

1.2 Existing Works and Motivating Questions

1.2.1 Formulations and abstract properties of RSPV

We first note that there are many variants of definitions for RSPV. For example, there are two subtlely different types of security notions, the *rigidity-based* (or isometry-based) soundness [25, 6] and *simulation-based* soundness [13, 29, 57]. Existing works do not seem to care about the differences; we note that these differences could have impact on the abstract properties of the definitions and could affect their well-behaveness. For example, we would like RSPV to have sequential composability between independent instances: if the client and the server execute an RSPV protocol for a state family $\mathcal{F}_1$, and then execute an RSPV protocol for a state family $\mathcal{F}_2$, we would like the overall protocol to be automatically an RSPV for $\mathcal{F}_1 \otimes \mathcal{F}_2$. If such a sequential composability property holds, protocols for tensor products of states could be reduced to protocols for each simple state family.

In this background, we argue that:

It's helpful to compare variants of definitions and formalize basic abstract properties.

We would like to have a more well-behaved framework for RSPV, which will lay a solid foundation for concrete constructions.

1.2.2 Definitions of the primitive for certifying the server's operations

RSPV talks about the certification of server-side *states*. A closely related question is: how could the client certify the server-side *operations*?

To address this problem, existing works raise the notion of self-testing [53, 48, 41]. One famous scenario of self-testing is in non-local games [34, 50]. In this scenario, the verifier sends questions to two spatially-separated but entangled quantum provers (or called servers). The verifier's questions and passing conditions are specially designed so that the provers have to perform specific operations to pass. This provides a way to constrain the provers' operations through only classical interactions and spatial separation, which has become a fundamental technique in the study of non-local games.

Recently a series of works [42, 32, 17, 45] study the single-server cryptographic analog of the non-local game self-testing. [42] studies the cryptographic analog of the CHSH game, where the server needs to prepare the Bell states and perform measurements on two of its registers. [43, 32] further extend it to three-qubit and N-qubit; [35, 45] make use of QFHE [38] to formulate and address the problem, where the FHE-encrypted part takes the role of one prover and the unencrypted part takes the role of the other prover.

What's common in these existing works is that they are defining the single-server cryptographic analog of the non-local game self-testing to be a protocol where the single server is playing the roles of *both* of the two provers. In this work we are interested in another viewpoint, which is:

How could we define a single-server cryptographic analog of the non-local game self-testing, where the single server is playing the role of one *of the two provers?*

1.2.3 Existing RSPV constructions

Maybe the most natural question in the direction of RSPV is:

For what state families could we achieve RSPV?

Let's quickly review what state families have been achieved in existing works. [29] achieves RSPV for $\{|+_\theta\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{\mathrm{i}\theta\pi/4}|1\rangle), \theta \in \{0, 1, 2 \cdots 7\}\}$; independently [25] gives a candidate RSPV construction for this state family and conjectures its security. [29] achieves RSPV for tensor products of BB84 states: $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}^{\otimes n}$. [57] achieves RSPV for tensor products of $|+_\theta\rangle$ states. [17] achieves RSPV for BB84 states.

For a broader viewpoint let's also review some famous self-testing protocols. [42] achieves cryptographic self-testing for Bell states and the corresponding X/Z measurements. [43] achieves self-testing for a 3-qubit magic state and the corresponding measurements. [32] achieves self-testing for multiple Bell pairs and measurements. [45] achieves self-testing for "all-X" and "all-Z" operations.

As a summary, one significant limitations of existing works is that they could only handle simple tensor product states and operators. We consider this situation very undesirable since ideally we want a protocol that is sufficiently powerful to cover all the computationally-efficient state families. The lack of concrete protocols also restricts the applications of RSPV as a protocol compiler: suppose that we have a quantum cryptographic protocol that starts with sending states in (for example) $\{\frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle), x_0, x_1 \in \{0,1\}^n\}$, it's not clear how to use existing RSPV protocols to compile this quantum communication step to the classical communication. (Note that although there are mature techniques for creating this type of states with some other security notions [38, 16, 39], it's not known how to construct RSPV for it.)

Besides the existency problem, it's also desirable if we could weaken the assumptions needed or simplify existing results.

1.2.4 Classical verification of quantum computations and its assumption

We will apply our results on RSPV to the classical verification of quantum computations (CVQC) problem. In this problem a classical client would like to delegate the evaluation of a BQP circuit C to a quantum server, but it does not trust the server; thus it wants to make use of a protocol to verify the results. The first and perhaps the most famous construction is given by Mahadev [39]. Building on Mahadev's work and related results [38, 16, 39], a series of new CVQC protocols are constructed [3, 29, 24, 57, 45, 5].

However, one undesirable situation is that all the existing constructions are either based on LWE, or based on the random oracle heuristic, or based on some new conjecture on the reduction between security notions. In more detail:

- Most existing works [39, 24, 29] make use of NTCF with the "adaptive hardcore bit property" (or some stronger variants). The only known instantiation of this primitive from standard assumption is based on LWE.
- [5] gives an instantiation of NTCF with a weaker variant of the adaptive hardcore bit property using cryptographic group actions. However, to achieve the adaptive hardcore bit property it relies on an unproven conjecture on the reduction between these two notions. Alternatively one could make use of the random oracle to construct CVQC without the adaptive hardcore bit property [18, 57] but the instantiation of the random oracle is heuristic.
- If we only make use of NTCF without the adaptive hardcore bit property, we could do RSPV for BB84 states [17], but BB84 states are not known to be sufficient for constructing CVQC.
- [45] makes use of a quantum FHE-based approach to construct CVQC. However instantiations of quantum FHE rely on (variants of) NTCF and the classical FHE [38, 31], which still rely

on the LWE assumption.

As mentioned above, a type of cryptographic assumption that is different from LWE is the cryptographic group actions, for example, supersingular isogeny [4, 23]. In this background, we ask:

Could we construct CVQC from cryptographic group actions?

1.3 Our Contributions

In this work we address or make progress to the questions above.

1.3.1 Definitions and abstract properties of RSPV

We first work on the definitions and abstract properties of RSPV. This part is in Section 2 and 3.

1. We first formulate and review the hierarchy of notions for formalizing RSPV. This includes the notions of registers, cq-states, protocols, paradigms of security definitions, etc.

2. We then formalize the notion of RSPV. We formulate the soundness as a simulation-based definition and compare this definition with several other variants (like the rigidity-based soundness, which is also popular).

3. We then study the abstract properties of RSPV including the sequential composability and amplification. This allows us to reduce the constructions of RSPV to primitives that are relatively easier to construct.

In summary, we clarify subtleties and build a well-behaved framework for RSPV, which enables us to build larger protocols from smaller or easier-to-construct components. In later part of this work we will build concrete RSPV constructions under this framework.

1.3.2 Our cryptographic analog of self-testing: remote operator application with verifiability (ROAV)

We then introduce a new notion called remote operator application with verifiability (ROAV), as our answer to the question in Section 1.2.2. This part is in Section 5.

Let's first review how the self-testing-based protocols typically work in the non-local game setting. Suppose the verifier would like to make use of the prover 1 and prover 2 to achieve some tasks —- for example, testing the ground state energy of a local Hamiltonian. One typical technique is to design two subprotocols π_{test} and π_{comp}. Subprotocol π_{test} is a non-local game with a self-testing property, while subprotocol π_{comp} is to test the Hamiltonian. Furthermore, the games are designed specially so that the prover 2, without communicating with the prover 1, could not decide which subprotocol the verifier is currently performing. Then the setting, the overall protocol and the security proof roughly go as follows:

(Setting) The prover 1 and prover 2 initially hold EPR states. They receive questions from the verifier, make the corresponding measurements and send back the results.

(Overall protocol) The verifier randomly chooses to execute either π_{test} or π_{comp} (without telling the provers the choices).

(Security proof)

1. To pass the overall protocol the provers have to pass π_{test} with high probability. By the property of π_{test} the operations of the prover 2 has to be close to the honest behavior.
2. By the design of π_{test} and π_{comp}, and the fact that the prover 2 is close to the honest behavior in π_{test}, we could argue that the prover 2 is also close to the honest behavior in π_{comp}.
3. Since we already know the prover 2 is close to be honest in π_{comp}, it's typically easy to analyze the execution of π_{comp} directly and show that it achieves the task.

Recall that we would like to define a single-server cryptographic analog of the non-local game self-testing where the single server plays the role of one of the two provers. So what does the "non-local game self-testing" mean here? Our idea is to consider both the step 1 and step 2 in the security proof template above as the "non-local game self-testing". Then let's focus on the prover 2 (as "one of the two provers" in our question) and assume the prover 1 is honest. Then we could do the following simplifications in the non-local game self-testing:

- In π_{test} we could assume the prover 1 first measures its states following the verifier's question; as a result, the joint state of the prover 1 and prover 2 becomes a cq-state where the verifer knows its description. Then the verifier also gets the measurement results from the prover 1's answer; so in the end we only need to consider the joint cq-state between the verifier and the prover 2.
- For π_{comp}, since we do not consider the step 3 above as a part of the self-testing notion, the question to the prover 1 in π_{comp} could be left undetermined. Then π_{comp} is not designed for any specific task any more, which in turn makes the self-testing a general notion.

Now the setting and the first two steps in the security proof could be updated as follows:

(Setting) In π_{test} the client and the prover 2 initially hold a cq-state, where the verifier holds the classical part and the prover 2 holds the quantum part. In π_{comp} the prover 1 and prover 2 hold EPR states and the verifier does not have access to the prover 1's information.

(Security proof) To make the first two steps in the security proof template above go through, we should at least require that:

"The prover could pass in π_{test}" should imply that the prover's operation in π_{comp} is close to the desired one.

This gives us the basic intuition for formalizing our single-server cryptographic analog of the non-local game self-testing. Below we introduce the notion, which we call *reomte operator application with verifiability* (ROAV).

An ROAV for a POVM $(\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_D)$ is defined as a tuple $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ where:

- The setting contains the following registers. $\boldsymbol{D}$ is a client-side classical register, $\boldsymbol{Q}$ is a server-side quantum register, $\boldsymbol{P}$ is a quantum register in the environment with the same dimension with $\boldsymbol{Q}$.
- ρ_{test} is a cq-state on registers $\boldsymbol{D}$ and $\boldsymbol{Q}$.

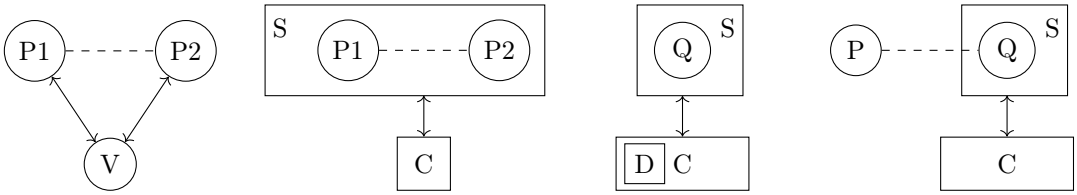

Figure 1: From the left to the right: the non-local game self-testing, previous definitions of its single party analog, and our definition (π_{test} and π_{comp} for the last two diagrams). Here $\leftrightarrow$ stands for interactions, and --- stands for quantum entanglements; C stands for the client and S stands for the server.

- π_{test} is the protocol for the test mode. In the test mode ρ_{test} is used as the input state and $\boldsymbol{P}$ is empty.
- π_{comp} is the protocol for the computation mode (where the operations are applied). Denote the state of maximal entanglement (multiple EPR pairs) between $\boldsymbol{P}$ and $\boldsymbol{Q}$ as Φ. In the comp mode Φ is used as the input state and $\boldsymbol{D}$ is empty. Note that the execution of π_{comp} does not touch $\boldsymbol{P}$.

The soundness of ROAV is defined roughly as follows: for any adversary Adv, at least one of the following is true:

- In the test mode (π_{test} is executed with input state ρ_{test} against adversary Adv), the adversary gets caught cheating with significant probability;
- In the comp mode (π_{comp} is executed with input state Φ against adversary Adv), the final state gives the outcome of the following operations:

  The measurement described by $(\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_D)$ is applied on $\boldsymbol{Q}$, and the client gets the measurement result $i \in [D]$; the corresponding output state (on register $\boldsymbol{P}$ and $\boldsymbol{Q}$) is $(\mathbb{I} \otimes \mathcal{E}_i)(\Phi)$.

We finally note that the definition of ROAV also use the simulation-based security definition paradigm, like RSPV.

ROAV as the operator analog of RSPV In the previous discussion we focus on the analog between our ROAV primitive and the non-local game self-testing. There is also another intuition that is analogous to the intuition of RSPV (see Section 1.1.1): In the ROAV problem the server is provided an undetermined input state, and the client would like to apply an operation (from an operator family) on it; in the end the client knows the description of the operation, while the server simply holds the output of the operation.

We also note that such a state-operator analog also appears in other directions like [51, 37].

Abstract constructions using ROAV After giving the abstract definitions, we show several potential applications of our notions. We first show that ROAV is potentially a useful tool for constructing RSPV protocols for more general state families. Then we construct a Hamiltonian ground energy testing protocol based on specific RSPV and ROAV. Our construction shares similarities to Grilo's Hamiltonian verification protocol in the 2-party setting [30]. We note that

these constructions are abstract constructions and concrete constructions for nontrivial ROAV remain open.

1.3.3 New RSPV constructions

Now we introduce a series of new RSPV constructions. Our results not only give arguably simplified constructions for existing results but also cover new state families. This part is in Section 6.

Overall approach Instead of constructing each protocol directly from cryptographic assumptions, we study how different protocols could be *reduced* to existing ones in a black-box manner. Following this approach, we build a series of RSPV protocols step by step from RSPV for BB84 states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. Furthermore, these steps could be classified into two classes:

- In one class of steps, the reduction either has a simple intuition or is an application of the abstract properties that we have proved in Section 3 (in more detail, sequential composability and amplification).
- In the other class of steps, we could work on an "information-theoretic core" (IT-core) where the analysis is purely quantum information theoretic, and there is no appearance of computational notions like computational indistinguishability.

Our reductions are illustrated in Figure 2. We elaborate technical details in Section 6.1. As an example, let's describe the first step of our reductions. We assume an RSPV for BB84 states; then the client would like to prepare a sequence of such states in which there is only one $|+\rangle$ state and no $|-\rangle$ state. Equivalently, this could be written as $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$ where the hamming weight of $x_0 \oplus x_1$ is exactly 1. What the client needs to do is to repeat the RSPV-for-BB84 protocol for many rounds and tell the server which states it wants to keep. This could be easily understood as a repeat-and-pick process.

We argue that our approach is cleaner and easier to understand compared to the original construction in [29] at least in the following sense: In [29] the computational indistinguishability arguments and quantum information theoretic arguments are mixed together, which could lead to complicated details [1]. By separating two types of steps in the reductions explicitly, each step has a relatively simple intuition and there is much less room for complicated details.

Main results Among these reductions, we consider the following two results particularly interesting.

Theorem 1.1. *Assuming the existence of RSPV for BB84 states, there exists an RSPV for state family* $\{\frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle), x_0, x_1 \in \{0,1\}^n\}$.

This is an RSPV protocol for a new state family. Note that although there are plenty of works [16, 18] that allow the client to prepare these states with some other types of security (like claw-freeness, etc), as far as we know, this is the first time that an RSPV for it is constructed.

We also recover the results of RSPV for 8-basis states [29].

Theorem 1.2. *Assuming the existence of RSPV for BB84 states, there exists an RSPV for state family* $\{|+_\theta\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{\mathrm{i}\pi\theta/4}|1\rangle), \theta \in \{0,1,2\cdots 7\}\}$.

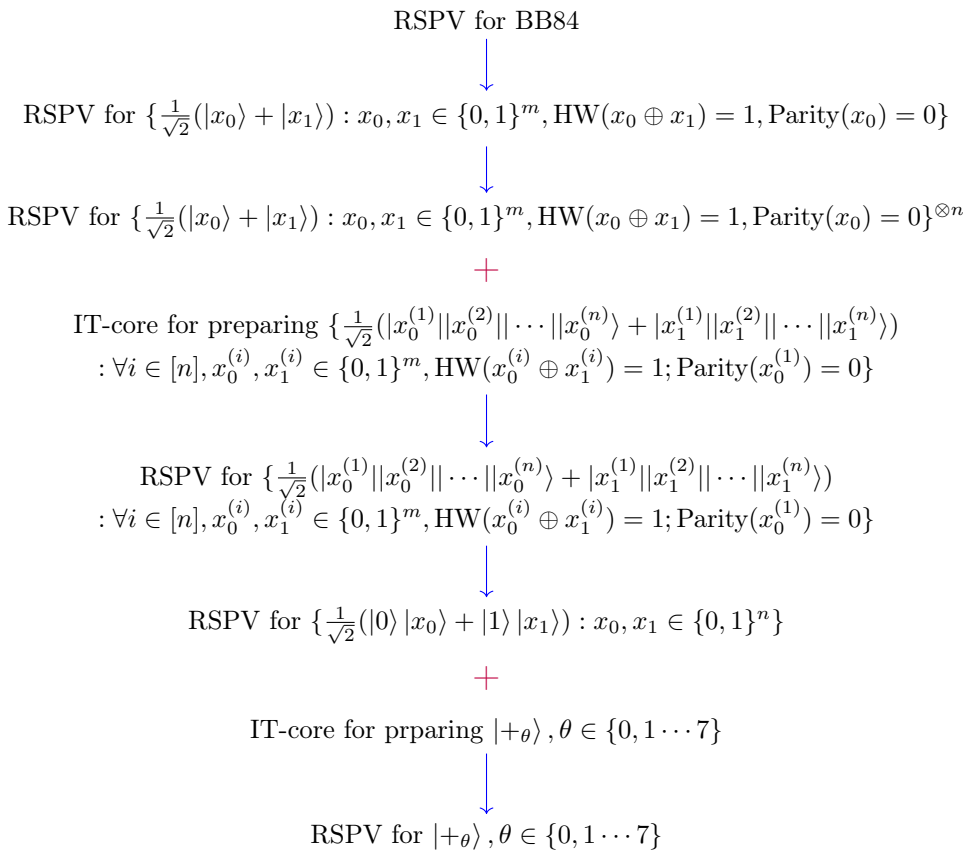

Figure 2: The reduction diagram. Here $\rightarrow$ means this step either has a simple intuition or is from the framework that we formalize in Section 3; $+$ means the analysis of the IT-core below it is purely quantum information theoretic, and the RSPV above it is used to compile the IT-core to a full cryptographic protocol. HW is the hamming weight, Parity is the total parity.

Instantiation of RSPV for BB84 We note that we still need to instantiate the RSPV for BB84 part to get a concrete protocol. Luckily the RSPV for BB84 has been studied relatively thoroughly: there are multiple constructions [6, 17] and we have a better understanding on the assumptions needed [17]. After instantiating the BB84 part by [17], we get RSPV constructions for these state families from weak NTCF, without requiring the adaptive hardcore bit property (elaborated below).

1.3.4 Application: CVQC from cryptographic group actions

Now we apply our results to the classical verification of quantum computations (CVQC) problem. This part is in Section 7.

As the preparation, we give a more detailed review on the variants of NTCF, and their relations to RSPV. We refer to Section 1.2.4 for a CVQC-centric background.

More backgrounds Noisy trapdoor claw-free functions (NTCF) is a popular and powerful primitive in quantum cryptographic tasks like CVQC and RSPV. Informally, this primitive is defined to be a function family that satisfies the following requirements. Below we use f to denote a function sampled from this function family.

- Trapdoor: this means that all the parties could evaluate f, but only the client, who holds the "trapdoor" information, could invert f.
- Noisy 2-to-1: the un-noisy 2-to-1 means that for each y in the range there exist exactly two preimages x_0, x_1 such that $f(x_0) = f(x_1) = y$. "Noisy" means that the evaluation of the function could be randomized, which makes the instantiation of the primitive easier.

  Below when we describe other properties we use the un-noisy version to simplify the description.
- Claw-free: the adversary could not efficiently find x_0, x_1 such that $f(x_0) = f(x_1)$.

In practice, we often use some variants of NTCF instead of the standard requirements above, to either make it more powerful or make the instantiation easier. Popular variants or additional requirements of NTCF include:

- Adaptive hardcore bit: the adversary could not find (d, y) with probability better than $\frac{1}{2}$ (the probability of random guessing) such that $d \cdot x_0 \equiv d \cdot x_1 \mod 2 \wedge d \neq 0$, where x_0, x_1 are two preimages of y.

  The RSPV for BB84 states is known to exist without this property [17], but Mahadev's constructions for CVQC [39] and RSPV for $|+_\theta\rangle$ [29] require this property.
- Extended function family: there is another function family g that is injective (instead of 2-to-1) and indistinguishable to f. Existing constructions for CVQC and RSPV for $|+_\theta\rangle$ [39, 29] also require this property.
- Inverse-polynomial correctness error (or weak correctness): this means that the 2-to-1 property is allowed to hold up to an inverse-polynomial error. This is used in [5] to define a primitive called weak TCF.

In our work we build our protocols on weak NTCF, that is, we allow inverse-polynomial correctness error and do not require additional properties like the adaptive hardcore bit property. In other words, we only use a relatively weak assumption from different variants of NTCF/TCF. This assumption could be instantiated from either LWE [16] or assumptions on group actions [5].

Applications of our results on CVQC We first note that, [17] could be adapted easily to weak NTCF:

Fact 1. *By [17], there exists an RSPV for BB84 assuming weak NTCF.*

As discussed in Section 1.3.3, this gives us a series of RSPV protocols from weak NTCF.

Then by [27], if the client samples and sends a series of $|+_\theta\rangle$ states to the server, it could do quantum computation verification using these states. Combining Fact 1, Theorem 1.2 and [27], we get:

Theorem 1.3. *Assuming the existence of weak NTCF, there exists a classical verification of quantum computation protocol.*

Finally we recall the results in [5]:

Theorem 1.4 ([5])**.** *Under certain assumptions on cryptographic group actions, there exists a family of weak TCF.*

Thus we have:

Theorem 1.5. *Under certain assumptions on cryptographic group actions, there exists a CVQC protocol.*

As an additional note, the work in [5] is largely on how to deal with the adaptive hardcore bit property; the fact that we do not need the adaptive hardcore bit may help us simplify the analysis or even construction in [5].

1.4 More Related Works

Previous versions The major versions of this works could be roughly described as follows.

- In the initial versions (comming out in around 2023), we develop a simple framework for working on RSPV problem, and raised the notion of ROAV.
- The next major versions are prepared for ITCS25 and made public in Nov 2024 (below we call it the ITCS version). Compare to the previous versions, the abstract framework parts (described in Section 1.3.1, 1.3.2) of this work are sigfinicantly updated, and the concrete constructions part (described in Section 1.3.3, 1.3.4) are completely new.
- The current versions have the following changes compared to the ITCS version: in Section 4 we give explicitly the translation from the results in [17] to an RSPV for BB84 states (that is, the proof of Fact 1). The additional technical works come from the fact that [17] describes their protocol as a test of a qubit, which is not the same as an RSPV for BB84 states; so some technical works are needed for translation.[1] We also discuss another approach for achieving CVQC from RSPV for BB84 state in Appendix B, based on the results in [44].[2]

[1] We thank Kaniuar Bacho, James Bartusek, Yasuaki Okinaka and anonymous reviewers for pointing this out.
[2] We thank anonymous reviewer for pointing out this approach.

Concurrent works After the ITCS version is made public, we get aware of two other concurrent works [9, 12] that have overlap with our work. In more detail:

- [9]: this work propose a compiler for transforming non-local games to single-server cryptographic protocols. Interestingly, their approach also leads to a protocol for CVQC from any plain (weak) NTCF, which implies a CVQC protocol from cryptographic group actions. Besides this overlap, the focus of their works and our works are different. (Note that although they also make use of a type of remote state preparation in their work, they do not study RSP with verifiablity; for comparison, our work primarily investigates RSPV.)
- [12]: this work studies a notion that they call *oblivious state preparation*. Interestingly, their approach also leads to a protocol for CVQC from any plain (weak) NTCF, which implies a CVQC protocol from cryptographic group actions. Besides this overlap, the focus of their works and our works are different. (Note that oblivious state preparation could be seen as a type of remote state preparation, but its soundness is quite different from RSPV. Our work focuses on RSPV, which is different from their work.)

RSP with other types of security There are many works about remote state preparation but with other types of security (instead of RSPV). These works may or may not use the name "remote state preparation". As examples, [31] gives an RSP for the gadgets in [26]; [52] gives an RSP for the quantum money states. When we only consider the honest behavior, RSP is a very general notion.

Other state-operation analog There are also several notions in quantum cryptography and complexity theory that have a state-operation analog. As examples, [51] studies the complexity of interactive synthesis of states and unitaries. [37] studies pseudorandom states and unitaries. In a sense, the relation of states and unitaries in these works is a "state-operation analog", as the RSPV and ROAV in our work.

1.5 Open Questions and Summary

Our work gives rise to a series of open questions; we consider the following two particularly interesting.

- One obvious open question coming out of this work is to give a construction for ROAV. Our work focuses on its definitions and applications in an abstract sense; an explicit construction of ROAV would allow us to instantiate these applications.
- For RSPV, although our results give new constructions for new state families, this is still far away from a general solution. Ideally we would like to have an RSPV for each computationally efficient state family. Whether this is possible and how to achieve it remain open.

In summary, two major part of this work is the framework and the constructions of RSPV protocols. By formulating and choosing notions and studying their abstract properties, we build an abstract framework for RSPV that is sufficiently well-behaved, which enables us to build more advanced, complicated protocols from more elementary, easy-to-construct components. Building on this framework, we construct a series of new RSPV protocols that not only (in a sense) simplify existing results but also cover new state families. Then we combine our results with existing works to show that the CVQC problem could be constructed from assumptions on cryptographic group actions. We also raised a new notion for certifying the server's operations. We consider our results as a solid progress in the understanding of RSPV; hopefully this will lay the foundation for further works.

Acknowledgements

This work is supported by different fundings at different time during its preparation:

- Partially supported by the IQIM, an NSF Physics Frontiers Center (NSF Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028).
- This work is partially done when the author was visiting Simons Institute for Theory of Computing.
- This work is partially supported by Zhongguancun Laboratory.

The author would like to thank Kaniuar Bacho, James Bartusek, Yasuaki Okinaka, Thomas Vidick, Zhengfeng Ji, Anne Broadbent, Qipeng Liu and anonymous reviewers for discussions.

2 Preparation

2.1 Basics of Quantum Information and Mathematics

We refer to [46] for basics of quantum computing, and refer to [36] for basics of cryptography. In this section we clarify basic notations and review basic notions.

2.1.1 Basic mathematics

Notation 2.1. We use $[m]$ to denote $\{1, 2, \cdots m\}$ and use $[0, m]$ to denote $\{0, 1, 2, \cdots m\}$. We use 1^n to denote the string $\underbrace{111\cdots1}_{\text{for } n \text{ times}}$. We use $S\backslash T$ to denote the set difference operation. We use $|S|$ or size(S) to denote the size of a set S. We use $\leftarrow_r$, or simply $\leftarrow$, to denote "sample from". $|\cdot|$ refers to the Euclidean norm by default when it's applied on a vector. We use $\mathbb{N}$ to denote the set of positive integers, use $\mathbb{Z}$ to denote the set of integers, and use $\mathbb{Z}_q$ to denote $\mathbb{Z}/q\mathbb{Z}$.

Below we review famous bounds in probability [56].

Fact 2 (Markov inequality)**.** *Suppose X is a non-negative random variable. Then*

$$\Pr[X \geq a] \leq \mathbb{E}[X]/a.$$

Fact 3 (Chernoff bounds)**.** *Suppose for all $i \in [K]$, s_i is a random variable independently sampled from $\{0, 1\}$ with probability $1 - p, p$ correspondingly. Then*

$$\forall \delta > 0, \Pr[\sum_{i\in[K]} s_i \geq (1+\delta)pK] \leq e^{-\delta^2 pK/(2+\delta)}$$

$$\forall \delta \in (0, 1), \Pr[\sum_{i\in[K]} s_i \leq (1-\delta)pK] \leq e^{-\delta^2 pK/2}$$

Theorem 2.1 (Azuma inequality)**.** *Suppose $(Z_t)_{t\in[0,K]}$ is a martingale. If for each $t \in [K]$, $|Z_t - Z_{t-1}| \leq c_t$, then*

$$\forall \delta > 0, \Pr[|Z_K - Z_0| \geq \delta] \leq 2e^{-\frac{\delta^2}{\sum_{t\in[K]} c_t^2}}$$

Notation 2.2. $\mathsf{poly}(n)$ means a function of n asymptotically the same as a polynomial in n. $\mathsf{negl}(n)$ means a function of n asymptotically tends to 0 faster than any polynomial in n.

2.1.2 Basics of quantum information

Notation 2.3. We use $\mathrm{Pos}(\mathcal{H})$ to denote the set of positive semidefinite operators over some Hilbert space $\mathcal{H}$ and we use $\mathrm{D}(\mathcal{H})$ to denote the set of density operators over $\mathcal{H}$.

Recall that density operators are positive semidefinite operators with trace equal to 1. We call a positive semidefinite operator a subnormalized density operator if its trace is less than or equal to 1.

Notation 2.4. For a pure state $|\Phi\rangle$, Φ is an abbreviation of $|\Phi\rangle\langle\Phi|$.

Fact 4. *Any $\rho \in D(\mathcal{H})$ could be purified to a state $|\varphi\rangle \in \mathcal{H} \otimes \mathcal{H}_{\boldsymbol{R}}$ such that the partial state of $|\varphi\rangle$ restricted on $\mathcal{H}$ is ρ.*

Notation 2.5. We use $\mathcal{E}(\rho)$ to denote the operation of an operator (either unitary or superoperator) on (normalized or unnormalized) density operator ρ.

We use CPTP as an abbreviation of "completely-positive trace-preserving". We use POVM as an abbreviation of "positive operator value measurement". A POVM is described by a tuple of superoperators $(\mathcal{E}_1, \mathcal{E}_2, \cdots \mathcal{E}_D)$.

Notation 2.6. $\Pr[\mathcal{E}(\rho) \to o]$ is defined to be $\mathrm{tr}(\Pi_o(\mathcal{E}(\rho)))$, where Π_o is the projection on the first qubit onto value o.

Notation 2.7. For positive semidefinite operators ρ, σ we use $\rho \approx_\epsilon \sigma$ to denote $\mathrm{TD}(\rho, \sigma) \leq \epsilon$, where $\mathrm{TD}(\cdot, \cdot)$ is the trace distance. We also say "ρ is ϵ-close to σ". For two pure states $|\varphi\rangle, |\psi\rangle$, we use $|\varphi\rangle \approx_\epsilon |\psi\rangle$ to denote $||\varphi\rangle - |\psi\rangle| \leq \epsilon$ where $|\cdot|$ is the Euclidean norm.

Note that there is a difference between trace distance and the Euclidean distance even for pure states.

Definition 2.1 (Bell basis)**.** In a two qubit system, define the following four states as the Bell basis:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$

$$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

Define $|\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, then these states could be denoted as $\mathsf{X}^a\mathsf{Z}^b|\Phi\rangle$, where X^a means to apply X if $a = 1$ and apply identity if $a = 0$. Z^b is defined similarly.

Now define the Bell-basis measurement as follows: the projection onto the Bell basis $\mathsf{X}^a\mathsf{Z}^b|\Phi\rangle$ has output value (a, b).

Finally we review the local Hamiltonian problem.

Definition 2.2 ([30])**.** The following problem is called the XZ k-local Hamiltonian problem:

Given input (H, a, b) where H is a Hamiltonian on n-qubit registers, a, b are real value function of n, and they satisfy:

$$H = \sum_{j \in [m]} \gamma_j H_j, \quad \forall j, |\gamma_j| \leq 1 \tag{1}$$

$$\forall j, H_j \in \{\sigma_X, \sigma_Z, I\}^{\otimes n} \text{ with at most } k \text{ appearances of non-identity terms} \tag{2}$$

Decide which is the case:

- Yes-instance: The ground energy of H is $\leq a$
- No-instance: The ground energy of H is $\geq b$.

Theorem 2.2 ([33]). *There exist $a(n), b(n) \in [0,1], b-a \geq 1/\mathsf{poly}(n)$ such that the XZ 5-local Hamiltonian problem is QMA-complete.*

2.2 Registers, States, and Protocols

2.2.1 Registers

In this work we work on *registers*. Intuitively, a register is like a modeling of a specific space in the memory of a computer. It allows us to use its name to refer to the corresponding state space. There are two types of registers: classical registers and quantum registers. For a classical register, the underlying state space is a finite set; for a quantum register, it corresponds to a Hilbert space. A classical register could be a tuple of other classical registers and a quantum register could be a tuple of other quantum registers. We refer to [55] for more information.

Notation 2.8. In this work we use the bold font (for example, $\boldsymbol{S}$) to denote registers. The corresponding Hilbert space of register $\boldsymbol{S}$ is denoted as $\mathcal{H}_{\boldsymbol{S}}$.

Then when we design quantum protocols, we could write protocols that take registers as part of inputs. So the inputs of a protocol could be either values or registers. This is also called "call-by-value" versus "call-by-reference" in programming languages like Java or Python. For classical values, both call-by-value and call-by-reference (registers) work; but for quantum states the call-by-value does not work in general due to the quantum no-cloning principle.

Notation 2.9. We use $\Pi_v^{\boldsymbol{reg}}$ to denote the projection onto the subspace that the value of register $\boldsymbol{reg}$ is v. We use $\Pi_{=\boldsymbol{reg}_2}^{\boldsymbol{reg}_1}$ to denote the projection onto the subspace that the value of register $\boldsymbol{reg}_1$ is equal to the value of register $\boldsymbol{reg}_2$.

2.2.2 States

For a classical register $\boldsymbol{C}$, we could consider a probability distribution over states in its state space. For a quantum register $\boldsymbol{S}$ and the associated Hilbert space $\mathcal{H}_{\boldsymbol{S}}$, we could consider the (normalized) density operators $\mathrm{D}(\mathcal{H}_{\boldsymbol{S}})$ as the quantum analog of probability distributions.

The joint state over a classical register $\boldsymbol{C}$ and a quantum register $\boldsymbol{S}$ is modeled as a *cq-state*. The density operator could be written as:

$$\rho = \sum_{c \in \mathrm{Domain}(\boldsymbol{C})} \underbrace{|c\rangle\langle c|}_{\boldsymbol{C}} \otimes \underbrace{\rho_c}_{\boldsymbol{S}}$$

where $\forall c, \rho_c \in \mathrm{Pos}(\mathcal{H}_{\boldsymbol{S}})$.

2.2.3 Execution model of protocols

In our work, we consider a setting where a client and a server interacting with each other. A more general modeling of cryptographic protocols might consider multiple parties and the scheduling of operations and messages might be concurrent; in this work we only consider the two-party setting between a client and a server and we could without loss of generality assume the operations are non-concurrent. That is, the overall protocol repeats the following cycle step by step:

1. The client does some operations on its own registers.
2. The client sends a message to the server.
3. The server does some operations on its own registers.
4. The server sends a messages to the client.

We call one cycle above as one round.

Each party should have a series of operations as the honest execution. We also consider the setting where some party is corrupted (or called malicious, adversarial). In this work we care about the setting where the server could be malicious. The adversary's (that is, malicious server's) operations could also be described by a tuple of operations for each round: $\mathsf{Adv} = (\mathsf{Adv}_{\text{round 1}}, \mathsf{Adv}_{\text{round 2}} \cdots \mathsf{Adv}_{\text{round n}})$, used in the step 3 in the cycle above. In later proofs we may simply use Adv_1, Adv_2, etc to refer to the adversarial operations in each round.

Notation 2.10. Consider a client-server setting described above. Suppose the protocol takes input registers $\boldsymbol{regs}$ and input values vals as its arguments. The overall operation of the whole protocol run against an adversary Adv is denoted as:

$$\text{ProtocolName}^{\mathsf{Adv}}(\boldsymbol{regs}, \text{vals}) \tag{3}$$

Then the final state of the protocol execution is determined by the operation above and the initial state. Suppose the initial state (on the registers considered) is ρ_{in}, the final state could be denoted as

$$\text{ProtocolName}^{\mathsf{Adv}}(\boldsymbol{regs}, \text{vals})(\rho_{\text{in}}) \tag{4}$$

About the messages and transcripts In this work we model the message sending operation in step 2 above as an operation that writes messages on a specific server-side empty register. Since this register is on the server side the adversary in the later steps might erase its value. One alternative way for modeling classical messages is to introduce a stand-alone transcript registers explicitly, whose values could not be changed by later operations. These two modelings are equivalent in the problems that we care about.

2.3 Basic Notions in Cryptographic Protocols

We would like to design protocols to achieve some cryptographic tasks. In this section we review the basic components for defining and constructing cryptographic protocols.

2.3.1 Completeness, soundness, efficiency

Consider the client-server setting where the client would like to achieve some task with the server but does not trust the server. To define the cryptographic primitive for this problem, typically there are at least two requirements: a completeness (or called correctness) requirement and a soundness (or called security) requirement. These two requirements could roughly go as follows:

- (Completeness) If the server is honest, the task is achieved and the client accepts.
- (Soundness) For any malicious server, if the task is not achieved correctly the client catches the server cheating.

Allowing for the completeness error and soundness error, the definition roughly goes as follows.

- (Completeness) If the server is honest, the task is achieved and the client accepts with probability c.
- (Soundness) For any malicious server, the probability of "the task is not achieved correctly and the client does not catch the server cheating" is at most s.

There should be $0 < s < c < 1$ for the definition to make sense. $1 - c$ is called the completeness error[3]. s is called soundness or soundness error.

The discussion above does not consider the efficiency. For many primitives we want the overall protocol to be efficient, and we could only achieve soundness against efficient adversaries. Thus there are three requirements for the primitive:

- (Completeness) Same as above.
- (Soundness) For any efficient malicious server, the probability of "the task is not achieved correctly and the client does not catch the server cheating" is at most s.
- (Efficiency) The honest opeartion of the whole protocol runs in polynomial time.

For many problems, it's not sufficient to simply use the security definitions informally described above. In Section 2.4 we will discuss different paradigms for defining the security.

2.3.2 Inputs, the security parameter, outputs, initial states

In this section we discuss the register set-ups for a cryptographic protocol and its initial states.

Inputs A protocol might take several parameters, input values, and input registers as its arguments.

A common special parameter for cryptographic protocols is the *security parameter*, which determines the security level of the protocol.[4]

Notation 2.11. In this paper we denote the security parameter as κ. It is typically given as inputs in the form of 1^κ.

Outputs As discussed in the previous section, there is a flag in the output which shows whether the client accepts or rejects (in other words, pass/fail, non-abort/abort). Thus for the cryptographic protocols that we care about in this work, the output registers consist of two parts: (1) the registers that hold the outputs of the tasks (for example, preparing a specific state); (2)the flag register.

Notation 2.12. The the flag register is denoted as $\boldsymbol{flag}$. We denote the projection onto the subspace $\boldsymbol{flag} = \mathsf{pass}$ as Π_{pass}.

In later constructions where there are multiple rounds of calling to some subprotocol, there might be multiple temporary flag registers. We use $\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i)}}$ to denote the projection onto the space that $\boldsymbol{flag}^{(1)}, \boldsymbol{flag}^{(2)} \cdots \boldsymbol{flag}^{(i)}$ are all in value pass.

[3] We note that for tasks for which no protocol could achieve perfect completeness or perfect soundness, the definitions of completeness error and soundness error might change.

[4] In addition to the security, we could also relate the completeness error to the security parameter so that when the security parameter increases both the completeness error and soundness error tend to zero.

Initial states In quantum cryptography, we need to consider initial states that are possibly entangled with the running environment of the protocol. The environment is not touched by the protocol itself (but will be given to the distinguisher, see Section 2.4.2). Denote the client-side registers as $\boldsymbol{C}$, denote the server-side registers as $\boldsymbol{S}$, and denote the register corresponding to the environment as $\boldsymbol{E}$, the initial state could be described as $\rho_{\text{in}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{C}} \otimes \mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$. For the client-server setting (as described in Section 2.2.3), when there is no client-side input states, we do not need to consider the client-side part of the initial state; then the initial state could be described as $\rho_{\text{in}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$.

Paper organization using set-ups In this paper we organize the information of parameters, registers etc in a latex environment called Set-up. This helps us organize protocol descriptions more clearly.

2.3.3 Cryptographic assumptions

For many problems, it's not possible to construct a protocol and prove its security unconditionally; we need to rely on *cryptographic assumptions*. One commonly-used and powerful cryptographic assumption is the Learning-with-Errors (LWE) assumption [49].

Definition 2.3 (LWE)**.** Suppose $n = n(\kappa), m = m(\kappa), q = q(\kappa)$ are integers, $\chi = \chi(\kappa)$ is a probability distribution. The $\mathrm{LWE}_{n,m,q,\chi}$ assumption posits that, when

$$\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m$$

for any BQP distinguisher[5], it's hard to distinguish

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$$

from

$$(\mathbf{A}, \mathbf{u}).$$

The $\mathrm{LWE}_{n,q,\chi}$ assumption posits that the $\mathrm{LWE}_{n,m,q,\chi}$ assumption holds for any $m = \mathsf{poly}(\kappa)$.

A typical choice of χ is a suitable discretization of the Gaussian distribution.

Another important cryptographic assumption is the quantum random oracle model (QROM) [15]. In this model we work in a setting where there is a global random oracle. This assumption is also used in the studies of RSP problems [59, 57]; in this work we do not work in this model.

2.4 Security Definition Paradigms

In this section we discuss different paradigms for defining securities. Especially, we review the simulation-based paradigm, which will be used in this work.

[5]It also makes sense to consider non-uniform distinguishers like BQP/qpoly.

2.4.1 State family and the indistinguishability

Convention 1. We first note that when we work on the security, we need to consider a family of states parameterized by the security parameter. More explicitly, consider the family of states $(\rho_\kappa)_{\kappa\in\mathbb{N}}$. Then we could use ρ to denote the operator in this state family corresponding to $\kappa \in \mathbb{N}$. For notation simplicity, we do not explicitly write down the state family and simply work on ρ.[6]

Below we review the notion of the indistinguishability, which is the foundation of many security definitions.

Notation 2.13. We write $\rho \approx_\epsilon^{ind:\mathcal{F}} \sigma$ if $\forall D \in \mathcal{F}$,

$$|\Pr[D(\rho) \to 1] - \Pr[D(\sigma) \to 1]| \leq \epsilon + \mathsf{negl}(\kappa).$$

We write $\rho \approx_\epsilon^{ind} \sigma$ when $\mathcal{F}$ is taken to be all the BQP algorithms. We omit ϵ when $\epsilon = \mathsf{negl}(\kappa)$.

The D in the definition above is called the distinguisher.

2.4.2 The simulation-based paradigm

In this section we introduce the simulation-based paradigm for security definitions.

In this paradigm, we compare the "real world" (or called the real protocol) with the "ideal world" (or called the ideal functionality) in the security definition.

- The real world is the real execution of the protocol itself.
- The ideal functionality achieves the tasks that we aim at as a trusted third party. So the security in the ideal world itself is simply by definition.

For the simulation-based paradigm, intuitively we want to say, each attempt of attacking the real protocol corresponds to an attempt on the ideal functionality; this "attempt on the ideal functionality" is described by a simulator. Since the ideal functionality is ideally secure by definition, we could be satisfied once we achieve the indistinguishability between the real protocol and the ideal functionality.

Let's make the security parameter implicit. Then the simulation-based security roughly goes as follows.

For any polynomial time quantum adversary Adv, there exists a polynomial time quantum simulator Sim such that:

$$\mathrm{Real}^{\mathsf{Adv}} \approx^{ind} \underbrace{\mathsf{Sim}}_{\text{take the role of } \mathsf{Adv}} \circ \mathrm{Ideal} \tag{5}$$

Note that Sim might contain multiple phases, for preparing inputs and processing outputs of Ideal.

Then note that (5) is about the indistinguishability of operations (instead of solely the indistinguishability of states as defined in Notation 2.13): it means that the initial state is adversarily chosen for distinguishing the output states. Furthermore, we need to consider the case that the initial state is entangled with some environment: although this part is not affected by the real protocol or the ideal functionality, the distinguisher has access to it.[7]

Recall that in this work we work in a client-server setting where the server could be malicious; let's expand (5) in this setting. Below we suppose for the initial states the server-side registers are

[6] Note that this notation convention is similar to how we express Definition 2.3: in that definition we use m to denote the value of function $m(\kappa)$ at κ.

[7] Omitting this part in the modeling could lead to missing of important attacks in some problems [19].

denoted as $\boldsymbol{S}$, and the environment is denoted as $\boldsymbol{E}$. For the output registers, denote the server-side output register as $\boldsymbol{Q}$. Then the security definition goes as follows.

For any polynomial time quantum adversary Adv, there exists a polynomial time quantum simulator Sim such that for any initial state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\mathrm{Real}^{\mathsf{Adv}}(\rho_0) \approx^{ind} (\underbrace{\mathsf{Sim}}_{\text{preparing inputs for Ideal and generating states on } \boldsymbol{S},\boldsymbol{Q}} \circ \mathrm{Ideal})(\rho_0) \tag{6}$$

Variants We discuss some variants of the definition.

- (6) is stated with negligible distinguishing advantage. In later sections we will need to work on non-negligible distinguishing advantage.
- We could consider different uniformity on the preparation of initial states. This is related to a notion called "advice" and we refer to [7] for its background.
- A variant that is less desirable but still useful is the inefficient simulator version, where the simulator is not required to be efficient. Typically a protocol with security under inefficient simulation could be sequentially composed with information-theoretically secure protocols, but in general the inefficient simulator might break other parts of the protocol in the security analysis.
- If stand-alone transcript registers are used explicitly (see Section 2.2.3), the simulator should also simulate the transcripts.

An example of the definition of the ideal functionality We note that in the client-server setting that we consider, a frequently used template of the ideal functionality is as follows.

Example 2.1. The Ideal takes a bit $b \in \{0, 1\}$ on the server side as part of the inputs and writes $\mathsf{pass}/\mathsf{fail}$ on a client-side register $\boldsymbol{flag}$ as part of the outputs.

- If $b = 0$, Ideal sets $\boldsymbol{flag}$ to be pass and prepares the outputs as in the honest behavior.
- If $b = 1$, Ideal sets $\boldsymbol{flag}$ to be fail and leave all the other output registers empty.

Intuitively this means that either both parties get the expected outputs in the honest setting, or the server is caught cheating.

Different modes of protocols In this work and several other works [34, 30], we need to deal with a pair of protocols under the same set-up, which are called two modes; typically it includes a test mode and a comp mode. The client does not tell the server which mode it will execute, so the adversary for these two modes could be assumed to be the same operation. The security is typically as follows: if the adversary passes the test mode protocol with high probability, the comp mode will achieve the desired security tasks (which may be defined using a simulation-based notion).

2.4.3 Other security definition paradigms

There are also other types of security definition paradigms. For example:

- Game-based security: in this paradigm there are a challenger and a distinguisher; the challenger samples a challenge bit and gives the distinguisher different states depending on the challenge bit. The distinguisher tries to predict the challenge bit. The security is roughly defined as follows: the distinguisher could not predict the challenge bit with probability non-negligibly higher than $\frac{1}{2}$.
- Framework for universal composability: a protocol with the simulation-based security does not necessarily remain secure when it's composed arbitrarily (for example, concurrently) with other protocols. The universal composability framework [22] and the AC framework [40] give a stronger security definition which allows universal composability.
- It's also possible to define securities by the unpredictability of certain strings or some information-theoretic properties.

Which paradigms or definitions to use could depend on the problems and our goals; stronger security properties are desirable on their own but might be hard to achieve. Depending on what we need, there could be many choices of security definitions even for the same problem.

3 Remote State Preparation with Verifiability: Definitions, Variants and Properties

In this section we formalize the definitions of RSPV, compare it with other variants and other RSP security notions, and study its basic properties.

- In Section 3.1, we formalize the notion of RSPV. We focus on the random RSPV, which means, the states are sampled (instead of chosen by the client) from a finite set of states.
- In Section 3.2 we discuss variants of security definitions.
- In Section 3.3 we discuss variants of the problem modeling. For example, we discuss a variant of RSPV where the states are chosen by the client (instead of sampled during the protocol).
- In Section 3.4 we study the sequential composability of RSPV.
- In Section 3.5 we study the security amplification of RSPV. We define a notion called preRSPV and show that this primitive could be amplified to an RSPV.

3.1 Definitions of RSPV

Below we consider a random RSPV that samples uniformly from a tuple of states.

Let's first formalize the register set-up of an RSPV protocol.

Set-up 1 (Set-up for an RSPV protocol)**.** Consider a tuple of states $(|\varphi_1\rangle, |\varphi_2\rangle \cdots |\varphi_D\rangle)$.

The protocol takes two parameters: security parameter 1^κ and approximation error parameter $1^{1/\epsilon}$.

The output registers are as follows:

- the server-side quantum register $\boldsymbol{Q}$ whose dimension is suitable for holding the states;
- the client-side classical register $\boldsymbol{D}$ with value in $[D]$;
- the client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$.

For modeling the initial states in the malicious setting, assume the server-side registers are denoted by register $\boldsymbol{S}$, and assume the environment is denoted by register $\boldsymbol{E}$.

As discussed in Section 2.3.1, there are three requirements for an RSPV protocol: completeness, soundness, efficiency. Below we formalize them one by one.

Definition 3.1 (Completeness of RSPV)**.** We say a protocol under Set-up 1 is μ-complete if when the server is honest, the output state of the protocol is μ-close to the following state:

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \otimes \underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}} \tag{7}$$

where

$$\rho_{tar} = \sum_{i\in[D]} \frac{1}{D} \underbrace{|i\rangle\langle i|}_{\boldsymbol{D}} \otimes \underbrace{|\varphi_i\rangle\langle\varphi_i|}_{\boldsymbol{Q}}. \tag{8}$$

And we simply say the protocol is complete if it has completeness error $\mathsf{negl}(\kappa)$.

The efficiency of the protocol is defined in the normal way. Below we formalize the soundness based on the paradigm in Section 2.4.2.

Definition 3.2 (Soundness of RSPV)**.** We say a protocol π under Set-up 1 is ϵ-sound if:

For any efficient quantum adversary Adv, there exists an efficient quantum operation Sim such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}(\rho_0)) \approx^{ind}_{\epsilon} \underbrace{\Pi_{\mathsf{pass}}}_{\text{on } \boldsymbol{flag}} (\underbrace{\mathsf{Sim}}_{\text{on } \boldsymbol{S},\boldsymbol{Q},\boldsymbol{flag}} (\underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}} \otimes \underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}})) \tag{9}$$

Here we only aim at approximate soundness (where the approximation error might be inverse polynomial). This is because in RSPV problems achieving negligible approximation error remains open and seems very difficult; and existing works all aim at inverse-polynomial errors [29, 25, 57].

We note that Definition 3.2 is subtlely different from the discussion of simulation-based paradigm in Section 2.4.2 in terms of how to treat the failing space. We choose Definition 3.2 because we feel that it's easier to work on; below we discuss this difference in more detail.

3.1.1 Handling the failing space within Ideal

A difference of Definition 3.2 from equation (6) is how we treat the failing (or called aborting, rejection) space. In Definition 3.2 the simulator is responsible for writing the passing/failing choices to the $\boldsymbol{flag}$ register, which is slightly different from the usual form of the simulation-based paradigm where the passing/failing flag is handled by the the ideal functionality (see Section 2.4.2). Then in Section 2.4.2 the simulator simulates both the passing space and the failing space, while in Definition 3.2 only the indistinguishability on the passing space is considered. Below we formalize the variant of the RSPV security that follows the usual form of the simulation-based paradigm.

Definition 3.3 (Variant of the soundness of RSPV)**.** Under Set-up 1, define RSPVIdeal as follows:

1. RSPVIdeal takes a classical bit $b \in \{0,1\}$ from the server side.
   - If $b = 0$: it sets $\boldsymbol{flag}$ to be pass and prepares ρ_{tar} on $\boldsymbol{D}, \boldsymbol{Q}$.
   - If $b = 1$: it sets $\boldsymbol{flag}$ to be fail.

Then the soundness of RSPV could be defined as follows. We say a protocol π is ϵ-sound if:

For any efficient quantum adversary Adv, there exist efficient quantum operations $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\pi^{\mathsf{Adv}}(\rho_0) \approx_{\epsilon}^{ind} \underbrace{\mathsf{Sim}_1}_{\text{on } \boldsymbol{S},\boldsymbol{Q}} (\text{RSPVIdeal}(\underbrace{\mathsf{Sim}_0}_{\text{on } \boldsymbol{S}\text{, generate } b \in \{0,1\} \text{ as the input of RSPVIdeal}} (\rho_0))) \tag{10}$$

We compare Definition 3.3 with Definition 3.2 as follows. We first compare the strength of the definitions, and compare the convenience of usage.

Comparison of strength First, we note that Definition 3.3 implies Definition 3.2. Starting from (10), we construct Sim that satisfies (9) as follows: Sim executes Sim_0, gets b, and executes Sim_1; then it sets $\boldsymbol{flag}$ to be pass if $b = 0$ and sets $\boldsymbol{flag}$ to be fail if $b = 1$.

But as far as we know, Definition 3.2 does not seem to imply Definition 3.3 directly (at least in an easy way). Intuitively, (9) only says the passing space is simulated by Sim, and there is no guarantee that the failing space is also simulated by the same simulator. It's possible to construct another simulator that simulates the failing space but it's not clear how to combine them into a single simulator directly. (If we are working in the classical world, since the initial states could always be cloned, it's much easier to combine these two simulators into one simulator that works for both cases; but in the quantum world since ρ_0 could not be cloned the simulation might destroy the initial states and the rewinding is not always possible, it's not clear how to go from equation (9) to equation (10).)

However, we will show that, for RSPV problem under Set-up 1, once we have an RSPV protocol with soundness under Definition 3.2, we could amplify it to a new protocol with soundness under Definition 3.3. The amplification is by a simple sequential repetition and cut-and-choose procedure: the same protocol is repeated for many rounds and one round is randomly chosen as the outputs. However, we note that this reduction will not work when there is additional client-side inputs (for example, some variants of RSPV discussed in Section 3.3.1); the amplification relies on the fact that the random RSPV does not have any client-side or server-side inputs other than the parameters.

The amplification protocol is given below.

Below we assume π is an RSPV under Set-up 1 that is complete and ϵ_0-sound under Definition 3.2.

Protocol 1 (Soundness amplification from Definition 3.2 to Definition 3.3)**.** An RSPV protocol under Set-up 1 is constructed as follows.

Parameters: approximation error parameter $1^{1/\epsilon}$, security parameter 1^{κ}. It is required that $\epsilon > \epsilon_0$.

Output registers: client-side classical registers $\boldsymbol{flag}^{(\mathrm{out})}, \boldsymbol{D}^{(\mathrm{out})}$, server-side quantum register $\boldsymbol{Q}^{(\mathrm{out})}$.

1. Define $L = \frac{216}{(\epsilon-\epsilon_0)^3}$.

   For each $i \in [L]$:

   (a) The client executes π with the server. Store the outputs in registers $\boldsymbol{flag}^{(i)}, \boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$.

2. The client randomly chooses $i \in [L]$.

   The client sets $\boldsymbol{flag}^{(\text{out})}$ to be fail if any one of $\boldsymbol{flag}^{(1)}, \boldsymbol{flag}^{(2)} \cdots \boldsymbol{flag}^{(i)}$ is fail. Otherwise it sets $\boldsymbol{flag}^{(\text{out})}$ to be pass and sends i to the server. Both parties use the states in $\boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$ as the outputs.

The completeness and efficiency are from the protocol; below we state and prove the soundness.

Theorem 3.1. *Protocol 1 is ϵ-sound under Definition 3.3.*

Proof. Consider an adversary Adv. Denote the initial state as ρ_0, and denote the output state by the end of the i-th round of step 1 as ρ_i. Then by the soundness of π (Definition 3.2) we get that, for any $i \in [L]$, there exists an efficient simulator Sim_i such that:

$$\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i)}}(\rho_i) \approx^{ind}_{\epsilon_0+\mathsf{negl}(\kappa)} \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\underbrace{\mathsf{Sim}_i}_{\text{on } \boldsymbol{S},\boldsymbol{Q}^{(i)},\boldsymbol{flag}^{(i)}} (\underbrace{\rho_{tar}}_{\boldsymbol{D}^{(i)},\boldsymbol{Q}^{(i)}} \otimes \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i-1)}}(\rho_{i-1}))) \tag{11}$$

Define $S_{\text{low pass}}$ as the set of i such that $\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i)}}(\rho_i)) \le (1 - \frac{1}{6}(\epsilon - \epsilon_0))\,\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i-1)}}(\rho_{i-1}))$. Then we note that:

- For each $i \notin S_{\text{low pass}}$,

$$\mathrm{tr}(\Pi_{\mathsf{fail}}^{\boldsymbol{flag}^{(i)}}(\pi^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i-1)}}(\rho_{i-1})))) = \mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i-1)}}(\rho_{i-1})) - \mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i)}}(\rho_i)) \le \frac{1}{6}(\epsilon - \epsilon_0). \tag{12}$$

- For each $i \in S_{\text{low pass}}$ such that[8] $|S_{\text{low pass}} \cap [i]| \ge 36/(\epsilon - \epsilon_0)^2$,

$$\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\le i-1)}}(\rho_{i-1})) \le \frac{1}{6}(\epsilon - \epsilon_0). \tag{13}$$

Then the simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ where Sim_0 operates on the server-side of ρ_0, and Sim_1 operates on $\mathrm{RSPVIdeal}(\mathsf{Sim}_0(\rho_0))$, is defined as follows (note that we abuse the notation and this $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ is different from the simulator in (11)).

Sim_0:

1. Sample a random coin $i \leftarrow [L]$. This is for simulating the client's random choice in the second step of Protocol 3.

2. Run $\tilde{\pi}_{<1.i}$ on ρ_0 and get $\tilde{\rho}_{i-1}$. Here $\tilde{\pi}_{<1.i}$ denotes the simulated protocol execution of Protocol 3 until the beginning of the i-th round of the first step. Here "simulated protocol execution" means that, instead of interacting with the client, the simulator simulates a client on its own. So the difference of ρ_{i-1} and $\tilde{\rho}_{i-1}$ is only the locations of these "client-side registers".

[8] Recall that $|\cdot|$ denotes the size of a set.

3. If all the $\boldsymbol{flag}$ registers so far have value pass, set[9] $b = 0$. If any $\boldsymbol{flag}$ registers so far has value fail, set $b = 1$.

Sim_1:

1. If $b = 0$, run Sim_i.
   If $b = 1$, run $\tilde{\pi}_{1.i}$ where $\tilde{\pi}_{1.i}$ is defined as above for simulating the i-th rounds of the first step of Protocol 1.

2. Run $\tilde{\pi}_{>1.i}$. Here $\tilde{\pi}_{>1.i}$ is defined similarly as above for simulating the $i+1 \sim L$ rounds of the first step of Protocol 1.

3. Disgard all the auxiliary registers used to simulate the client-side information.

We prove this simulator achieves what we want. We could compare the simulated output states with the output states from the real execution and see where the distinguishing advantage could come from (where the distinguishing advantage refers to the approximation error in the soundness definition; see equation (10)):

- For $i \notin S_{\text{low pass}}$:
  - Equation (11) itself contributes an error of ϵ_0.
  - $\mathrm{tr}(\Pi_{\mathsf{fail}}^{\boldsymbol{flag}^{(i)}}(\pi^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1}))))$ contributes an error of $\frac{\epsilon-\epsilon_0}{6}$ by (12) on both sides of equation (10).
  - $(\mathbb{I} - \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}})(\rho_{i-1})$ part is simulated perfectly in later simulation.
- For $i \in S_{\text{low pass}}$, although we do not know the size of $S_{\text{low pass}}$, we could divide this set into S_{head} and S_{tail} where S_{head} is the set of the first $\frac{36}{(\epsilon-\epsilon_0)^2}$ elements and S_{tail} is the set of the remaining elements. Below we bound the effect of these two sets on the distinguishing advantage.
  - The size of S_{head} is at most $\frac{36}{(\epsilon-\epsilon_0)^2}$ thus it contributes an error of $\frac{\epsilon-\epsilon_0}{6}$ on both size of equation (10).
  - For $i \in S_{\text{tail}}$, this part contributes at most an error of $\frac{\epsilon-\epsilon_0}{6}$ by (13) on both size of equation (10).
  - $(\mathbb{I} - \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}})(\rho_{i-1})$ part is simulated perfectly in later simulation.

Summing them up we get the total approximation error to be bounded by $\epsilon_0 + 2(\frac{\epsilon-\epsilon_0}{6} + \frac{\epsilon-\epsilon_0}{6} + \frac{\epsilon-\epsilon_0}{6}) \leq \epsilon$. □

Comparison on the convenience of usage Definition 3.3 is more standard: it follows the usual form of the simulation-based security (as in Section 2.4.2). The fact that it seems stronger than Definition 3.2 could also be considered as an advantage. But Definition 3.2 is simpler to work on when we build new RSPV protocols on smaller protocols: we only need to work on a single simulator modeled as a superoperator instead of two.

[9] Recall Definition 3.3 on the bit b of RSPVIdeal inputs.

Convention Starting from Section 3.3, we will mainly use Definition 3.2.

3.2 Variants of the Security Definition

3.2.1 The rigidity-based soundness

A popular variant of the RSPV security that is subtlely different from the simulation-based soundness is the *rigidity-based soundness*. Roughly speaking, the rigidity-based soundness says the output state, after going through an isometry on the server side, is close to the target state.

Definition 3.4 (Rigidity-based soundness of RSPV)**.** We say a protocol π under Set-up 1 is ϵ-sound under rigidity-based soundness if:

For any efficient quantum adversary Adv, there exist a efficient isometry V and[10] an efficient quantum operation Sim such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\Pi_{\mathsf{pass}}(\underbrace{V}_{\text{on } \boldsymbol{S},\boldsymbol{Q}}(\pi^{\mathsf{Adv}}(\rho_0))) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}} \otimes \underbrace{\mathsf{Sim}}_{\text{on } \boldsymbol{S},\boldsymbol{flag}}(\underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}})) \tag{14}$$

Comparison with the simulation-based soundness We compare the intuition of simulation-based soundness and rigidity-based soundness as follows.

- An interpretation of the rigidity-based soundness is "if the protocol passes, the server really gets the state".
- An interpretation of the simulation-based soundness is "if the protocol passes, what the server gets is no more than the state".

And we could prove the simulation-based soundness as defined in Definition 3.2 is no stronger than the rigidity-based soundness defined above:

Theorem 3.2. *Suppose a protocol π under Set-up 1 is ϵ-sound under rigidity-based soundness, then it's also ϵ-sound under simulation-based soundness.*

Proof. By the rigidity-based soundness we get V, Sim that satisfies (14). Then taking

$$\mathsf{Sim}'(\underbrace{\cdot}_{\rho_{tar}} \otimes \underbrace{\cdot}_{\rho_0}) = V^{\dagger}(\underbrace{\cdot}_{\rho_{tar}} \otimes \mathsf{Sim}(\underbrace{\cdot}_{\rho_0}))$$

as the simulator in (9) completes the proof. □

[10] We note that this definition is slightly different from the (rigidity-based) definitions in existing works [29, 6]. In [29, 6] the left hand side of (14) is statistically close to a state in the form of $\sum_i |\varphi_i\rangle\langle\varphi_i| \otimes \sigma_i$, and then a computational indistinguishability requirement is put on σ_i for different i. We argue that our global indistinguishability captures the same intuition and is more general; what's more, a suitable formulation of variants of definitions in [29, 6] should imply this definition.

One obstacle of showing a direct implication from the definitions in [29, 6] is on the efficient simulatable property on these σ_i: the definitions of rigidity-based soundness used in [29, 6] does not seem to imply it could be written in the form of $\mathsf{Sim}(\rho_0)$. A more careful analysis of the relations between this definition and existing rigidity definitions remains to be done and is out of the scope of this work.

But the inverse does not seem to be true. Actually, the rigidity-based soundness of RSPV is not even resilient to an additional empty timestep (that is, no party does anything) at the end of the protocol: the adversary could destroy everything in the end to violate the rigidity requirement. For comparison, the simulation-based notion has such resilience: the state destroying operation could be absorbed into the simulator in (9). Arguably this also means the simulation-based notion has more well-behaved properties.

Although the rigidity-based notion seems stronger, intuitively the simulation-based version is as useful as the rigidity-based version in common applications of RSPV. When we construct cryptographic protocols, what we are doing is usually to enforce that the malicious parties could not do undesirable attacks. In the simulation-based soundness it is certified that what the adversary gets is no more than the target state, which intuitively means that it is at least as secure as really getting the target state.

3.2.2 Other subtleties

We refer to Section 2.4.2 for discussions on simulator efficiency and the modeling of transcripts.

There are other choices of uniformity when we model the inputs and initial states. In Definition 3.2 we consider all initial states $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$, which implicitly taking the quantum advice into consideration. Alternatively, we could require that all the inputs and initial states are uniformly efficiently preparable, so there is no advice string or state; or we could require the initial quantum states to be efficiently preparable but allow the classical advice. In this work we feel that allowing the quantum advice is more convenient and intuitive since we need to work on quantum states that are not necessarily efficiently preparable (for example, ground states of Hamiltonians).

3.2.3 Other security notions

Basis blindness Besides the RSP with verifiability, there is also a security notion for RSP called *basis blindness*. This is defined in [25] as a security notion for RSP for $|+_\theta\rangle$ states. Intuitively it means that, no malicious server could predict the two less-significant bits of θ; alternatively it could be formulated as a game-based security notion, where the distinguisher needs to distinguish the two less-significant bits from uniformly random bits. An RSP with this type of security does not work as a protocol compiler (for replacing the transmissions of quantum states with cheaper resources) in general; but it could indeed be used to compile certain protocols securely, especially the universal blind quantum computation protocol [21, 59, 10].

Universal composability A stronger security requirement is the universal composability, as mentioned in Section 2.4.3. Unfortunately, as shown in [10], it's not possible to construct RSP protocol with this type of security. But this does not rule out the possibility of using weaker security notions, like Definition 3.2 or any other notion mentioned above. We note that although Definition 3.2 does not have universal composability, it still satisfies sequential composability (see Section 3.4).

Unpredictability of keys For certain state families, we could also formalize the security as unpredictability of certain strings. For example, consider the RSP for states in the form of $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$, $x_0, x_1 \in \{0,1\}^n$. A useful security notion is to require that $x_0||x_1$ is unpredictable (where $||$ is the concatenation notation). Then we could use a primitive called NTCF to construct

such an RSP (see [16] for details). As another example, [59] makes heavy use of this type of security notions (possibly in some advanced forms) as tools.

Other RSP notions There are also RSP works with other types of security notions. For example, [52] studies the remote preparation of quantum money states; the final state satisfies unclonability and verifiability for quantum money states. Note that the verifiability for quantum money states is different from the verifiability in RSPV.

3.2.4 Purified joint states

As a security proof technique, in the studies of RSP we often need to work on the purified joint states. In all the previous discussions we work on density operators. In quantum information density operators could be purified into pure states by introducing a reference register $\boldsymbol{R}$ (see Fact 4); then superoperators could be equivalently replaced by the corresponding unitaries.

In more detail, we could first purify the initial state ρ_0 in Definition 3.2 as a pure state on $\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}} \otimes \mathcal{H}_{\boldsymbol{R}}$. Then notice that we could enlarge $\boldsymbol{E}$ to contain $\boldsymbol{R}$, which gives the distinguisher access to more registers; and this does not weaken the overall statement because indistinguishability on more registers implies indistinguishability on parts of them. Thus to prove Definition 3.2 it's sufficient to only consider pure initial states in $\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}}$.

Then we could do purification for each classical register appeared in the protocol. There are two ways to do this purification:

- Assume the classical registers are purified by corresponding reference registers.
- Simply replace the classical registers and classical states by quantum registers and quantum states that look the same with the original states on computational basis; and then put a requirement that operators operated on them could only read the values (that is, CNOT the contents to other registers) but should not revise the values of these registers. When this requirement is satisfied, the purified state is indistinguishable to the original cq-state.

Working on purified joint states enables some proof techniques that are hard to use or even unavailable when we work on density operators. Different security proofs could use different ways of purification.

As examples, [20] purifies the client-side classical information to reduce the original protocol to a entanglement-based protocol; [59] uses heavily "state decomposition lemmas" and the triangle inequality, which works on purified joint states. In this work we will use this type of technique to prove Lemma 6.3 (see Section 6.3.2), which requires decomposing and combining components of the purified joint state.

3.3 Variants and Generalizations of the Problem Modeling

In the previous sections we focus on random RSPV for a fixed size state family; in this section we discuss its variants and generalizations. In Section 3.3.1 we formalize the RSPV notions where the state families are parameterized by problem size parameters (instead of a fixed size) or where the states are chosen by the client (instead of being sampled randomly), and discuss the notion where the states are prepared from some initial resource states (instead of being generated from scratch). In Section 3.3.2 we discuss a generalization that allows us to prepare arbitrary efficiently preparable states (instead of some specific state families).

3.3.1 Problem size parameters, sampling versus choosing and resource states

Problem size parameters In the previous sections we work on a tuple of states of a fixed size (see Set-up 1). In the later sections we will need to work on state families parameterized by a size parameter. As an example, we need to work on states in $\{\frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle) : x_0, x_1 \in \{0,1\}^n\}$; here the problem size parameter is n. Below we formalize the set-up and the security definition, which are adapted from Set-up 1 by adding the problem size parameter.

Set-up 2. Consider a state family parameterized by 1^n: denoted it as $S = (S_n)_{n\in\mathbb{N}}$, where each S_n is a set of states; suppose the set of their descriptions is D_n, and suppose $S_n = \{|\varphi_k\rangle\}_{k\in D_n}$.

The protocol takes the following parameters:

- problem size parameter 1^n;
- security parameter 1^κ;
- approximation parameter $1^{1/\epsilon}$.

Corresponding to problem size 1^n, the output registers are as follows:

- the server-side quantum register $\boldsymbol{Q}$ for holding states in S_n;
- the client-side classical register $\boldsymbol{D}$ for holding descriptions in D_n;
- the client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$.

For modeling the initial states in the malicious setting, assume the server-side registers are denoted by register $\boldsymbol{S}$, and assume the environment is denoted by register $\boldsymbol{E}$.

Definition 3.5. Corresponding to problem size parameter 1^n, define the target state $\rho_{tar} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}} \otimes \mathcal{H}_{\boldsymbol{Q}})$ as

$$\rho_{tar} = \sum_{k\in D_n} \frac{1}{\mathrm{size}(D_n)} \underbrace{|k\rangle\langle k|}_{\boldsymbol{D}} \otimes \underbrace{|\varphi_k\rangle\langle\varphi_k|}_{\boldsymbol{Q}}. \tag{15}$$

And the completeness is defined similarly to Definition 3.1.

Definition 3.6. We say a protocol π under Set-up 2 is ϵ-sound if for any $n = \mathsf{poly}(\kappa)$, for any efficient quantum adversary, there exists an efficnent quantum operation Sim such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}(1^n)(\rho_0)) \approx^{ind}_{\epsilon} \underbrace{\Pi_{\mathsf{pass}}}_{\text{on } \boldsymbol{flag}} \Big(\underbrace{\mathsf{Sim}(1^n)}_{\text{on } \boldsymbol{S},\boldsymbol{Q},\boldsymbol{flag}} \Big(\underbrace{\qquad\qquad \rho_{tar} \qquad\qquad}_{\boldsymbol{D},\boldsymbol{Q} \text{ as given in equation (15)}} \otimes \underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}} \Big) \Big) \tag{16}$$

Note that in (16) we make 1^n explicit and the other two parameters are implicit as before.

In later sections we also need to consider multiple problem size parameters (for example, $1^m, 1^n$), and the definitions could be adapted correspondingly. We could also consider problem sizes that satisfy some conditions (for example, m is bigger than some functions of n), and the security definition could also be adapted correspondingly.

RSPV where the client chooses the states So far we focus on the *random* RSPV, where the states are sampled randomly by the protocol. We could also study a form of RSPV notion where the states are chosen by the client.

Below we formalize this notion for a tuple of states $(|\varphi_1\rangle, |\varphi_2\rangle \cdots |\varphi_D\rangle)$.

Set-up 3 (Set-up for an RSPV where the client chooses the state)**.** As in Set-up 1, consider a tuple of states $(|\varphi_1\rangle, |\varphi_2\rangle \cdots |\varphi_D\rangle)$.

The protocol takes the security parameter 1^κ and approximation error parameter $1^{1/\epsilon}$ as in Set-up 1.

The protocol takes a classical input $i \in [D]$ from the client.

Compared to Set-up 1, the output registers are solely $\boldsymbol{flag}$ and $\boldsymbol{Q}$.

$\boldsymbol{S}, \boldsymbol{E}$ are defined in the same way as Set-up 1.

Definition 3.7. When the input is $i \in [D]$, the target state is simply $|\varphi_i\rangle\langle\varphi_i| \in \mathrm{D}(\mathcal{H}_{\boldsymbol{Q}})$.

Then the completeness is defined similarly to Definition 3.1:

We say a protocol under Set-up 3 is complete if for any input value $i \in [D]$, when the server is honest, the output state of the protocol is negligibly close to the following state:

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \otimes \underbrace{|\varphi_i\rangle\langle\varphi_i|}_{\boldsymbol{Q}}$$

To formalize the soundness, we first define the ideal functionality, and formalize the soundness using the simulation-based paradigm (see Section 2.4.2).[11]

Definition 3.8. Define RSPVIdeal as follows:

Ideal takes $i \in [D]$ from the client and $b \in \{0,1\}$ from the server. Then depending on the value of b:

- If $b = 0$: it sets $\boldsymbol{flag}$ to be pass and prepares $|\varphi_i\rangle\langle\varphi_i|$ on $\boldsymbol{Q}$.
- If $b = 1$: it sets $\boldsymbol{flag}$ to be fail.

We say a protocol under Set-up 3 is ϵ-sound if:

For any efficient quantum adversary Adv, there exist efficient quantum operations $\mathsf{Sim}_0, \mathsf{Sim}_1$ such that for any input value $i \in [D]$, any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

$$\pi^{\mathsf{Adv}}(\underbrace{i}_{\text{client-side input}})(\rho_0) \approx_\epsilon^{ind} \underbrace{\mathsf{Sim}_1}_{\text{on } \boldsymbol{S},\boldsymbol{Q}} (\mathrm{RSPVIdeal}(\underbrace{\mathsf{Sim}_0}_{\text{on } \boldsymbol{S}\text{, generate } b \in \{0,1\} \text{ as the input of RSPVIdeal}}(\underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}}))) \tag{17}$$

Adding resource states In later sections, we will need to consider the following variants: the protocol does not prepare states from scratch; instead the server holds some initial resource states while the client knows their descriptions, and the overall protocol could be seen as a transformation from the resource states to the target states. To formalize such a notion the set-up, completeness and soundness need to be adapted to include these resources states. We will encounter a notion in this form in Section 3.5.3.

Finally we note that we could consider combinations of these variants; the set-ups, completeness and soundness are defined in the natural way.

[11] Note that, as discussed in Section 3.1.1, the simulate-only-passing-space simplification used in Definition 3.2 does not apply to the case where there are client-side inputs, which is the case here.

3.3.2 RSPV for arbitrary efficiently preparable states

In the previous sections we focus on RSPV for some specific state families. We would like to get RSPV protocols for more and larger state families, and ideally, we could like to have a universal RSPV protocol for arbitrary efficiently preparable states. There are two notions for this problem that seem natural. These two notions roughly go as follows:

- We could consider an arbitrary efficient quantum circuit C that prepares a cq-state on $\mathrm{D}(\mathcal{H}_{\boldsymbol{D}} \otimes \mathcal{H}_{\boldsymbol{Q}})$, and use this state as the target state. C is given as a public input string.
- We could consider an arbitrary efficient quantum circuit C that prepares a quantum state on $\mathrm{D}(\mathcal{H}_{\boldsymbol{Q}})$, where C is chosen by the client (in other words, C is a client-side input string).

We leave the formalizations and concrete constructions for further studies.

3.4 Sequential Composition Property of RSPV

3.4.1 Composition between RSPV protocols

First we could prove RSPV (under the simulation-based soundness as defined in Definition 3.2) has a natural composition property when different RSPV protocols are sequentially composed with each other.

Below we work under Set-up 1. Suppose π_1 is an RSPV for target state ρ_{tar} and is ϵ_1-sound, π_2 is an RSPV for target state σ_{tar} and is ϵ_2-sound.

Protocol 2 (Sequential composition between different RSPV protocols)**.** Output registers: client-side classical register $\boldsymbol{flag}$, client-side classical register $\boldsymbol{D} = (\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)})$, server-side quantum register $\boldsymbol{Q} = (\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)})$.

1. Execute π_1. Store the outputs in $(\boldsymbol{flag}^{(1)}, \boldsymbol{D}^{(1)}, \boldsymbol{Q}^{(1)})$. In the honest setting $\boldsymbol{flag}^{(1)}$ has value pass and $(\boldsymbol{D}^{(1)}, \boldsymbol{Q}^{(1)})$ holds state ρ_{tar}.
2. Execute π_2. Store the outputs in $(\boldsymbol{flag}^{(2)}, \boldsymbol{D}^{(2)}, \boldsymbol{Q}^{(2)})$. In the honest setting $\boldsymbol{flag}^{(2)}$ has value pass and $(\boldsymbol{D}^{(2)}, \boldsymbol{Q}^{(2)})$ holds state ρ_{tar}.
3. The client sets $\boldsymbol{flag}$ to be fail if any one of $\boldsymbol{flag}^{(1)}$, $\boldsymbol{flag}^{(2)}$ is fail. Otherwise it sets $\boldsymbol{flag}$ to be pass.

Theorem 3.3. *Protocol 2 is an RSPV protocol for target state $\rho_{tar} \otimes \sigma_{tar}$ and is $(\epsilon_1 + \epsilon_2)$-sound.*

The completeness and efficiency are from the protocol description. Below we prove the soundness.

Proof. For adversary Adv, suppose the initial joint state is $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$, and the output state of the first step of the protocol (after the execution of π_1) is ρ_1. By the soundness of π_1 there exists an efficient simulator Sim_1 such that:

$$\Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\rho_1) \approx^{ind}_{\epsilon_1} \Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\underbrace{\mathsf{Sim}_1}_{\text{on } \boldsymbol{S},\boldsymbol{Q}^{(1)},\boldsymbol{flag}^{(1)}}(\rho_{tar} \otimes \rho_0)) \tag{18}$$

Then assume the output state after the second step of the protocol (after the execution of π_2) is ρ_2. Then by the soundness[12] of π_2 there exists an efficient simulator Sim_2 such that:

$$\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(2)}}\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_2) \approx_{\epsilon_2}^{ind} \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(2)}}(\underbrace{\mathsf{Sim}_2}_{\text{on } \boldsymbol{S},\boldsymbol{Q}^{(1)},\boldsymbol{Q}^{(2)},\boldsymbol{flag}^{(2)}}(\sigma_{tar}\otimes\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_1))) \tag{19}$$

Let's construct the overall simulator Sim. Sim is defined as follows:

1. Execute Sim_1 on registers $\boldsymbol{S}$, $\boldsymbol{Q}^{(1)}$ and $\tilde{\boldsymbol{flag}}^{(1)}$. (Note that this simulation does not have access to $\boldsymbol{flag}^{(1)}$, but we could use a temporary register $\tilde{\boldsymbol{flag}}^{(1)}$ as a replacement.)
2. Execute Sim_2 on registers $\boldsymbol{S}$, $\boldsymbol{Q}^{(1)}$, $\boldsymbol{Q}^{(2)}$ and $\boldsymbol{flag}^{(2)}$. (Note that this simulation does not have access to $\boldsymbol{flag}^{(2)}$, but we could use a temporary register $\tilde{\boldsymbol{flag}}^{(2)}$ as a replacement.)
3. Set $\boldsymbol{flag}$ to be fail if any one of $\tilde{\boldsymbol{flag}}^{(1)}$, $\tilde{\boldsymbol{flag}}^{(2)}$ is fail. Otherwise set $\boldsymbol{flag}$ to be pass.

Combining (18)(19) we get that

$$\Pi_{\mathsf{pass}}^{\boldsymbol{flag}}(\rho_2) \approx_{\epsilon_1+\epsilon_2}^{ind} \Pi_{\mathsf{pass}}^{\boldsymbol{flag}}(\underbrace{\mathsf{Sim}}_{\text{on } \boldsymbol{S},\boldsymbol{Q},\boldsymbol{flag}}(\sigma_{tar}\otimes\rho_{tar}\otimes\rho_0))$$

which completes the proof. □

3.4.2 RSPV as a protocol compiler

One important application of RSPV is to serve as a protocol compiler: given a protocol starting with "the client prepares and sends some quantum states", we could replace this step with an RSPV protocol for these states, and the security property of the overall protocol will be preserved (approximately).

In more detail, assume a protocol π has the following set-up: the client holds a classical register $\boldsymbol{D}$, the server holds a quantum register $\boldsymbol{Q}$, and in the honest setting initially the client and the server should hold the state $\rho_{tar} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}} \otimes \mathcal{H}_{\boldsymbol{Q}})$.

Then suppose π_{RSPV} is an RSPV protocol for target state ρ_{tar} and is ϵ-sound. Then consider the following protocol:

1. Execute π_{RSPV}. Break out if the protocol fails.
2. Execute π.

Then by the RSPV soundness there is

$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}_2}(\pi_{\mathrm{RSPV}}^{\mathsf{Adv}_1}(\rho_0))) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}_2}(\mathsf{Sim}(\rho_{tar}\otimes\rho_0)))$$

which means that we could replace the honest set-up of initial states by a call to the RSPV protocol and the properties of the final states will be approximately preserved.

[12] We note that to get (19) we apply Definition 3.2 to an unnormalized initial state $\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_1)$. This is fine because we could do a case analysis on the trace of $\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_1)$ before applying Definition 3.2: when $\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_1)) < \epsilon_2/2$ (19) is automatically true, while when $\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(1)}}(\rho_1)) \geq \epsilon_2/2$ we could first rescale the initial state before applying Definition 3.2.

3.5 PreRSPV and Security Amplification

In this section we discuss the soundness amplification of RSPV. We will define a notion called preRSPV and shows that it could be amplified to an RSPV. The name and the notion of preRSPV have already been used in [57] and here we re-formalize this notion. In more detail:

- In Section 3.5.1 we first introduce the notion of preRSPV. The preRSPV is defined as a pair of protocols $(\pi_{\text{test}}, \pi_{\text{comp}})$ that share the same set-up. The soundness is defined intuitively as follows: if π_{test} running against an adversary Adv could pass with high probability, then π_{comp} running against the same adversary Adv will satisfy the RSPV soundness (that is, simulated by the target state).

  Then a preRSPV defined above could be amplified to an RSPV by a *cut-and-choose* procedure: for each round, the client randomly chooses to execute either π_{test} or π_{comp} with some probabilities (without telling the server the choices). If the server could keep passing in many rounds, then for most of the π_{comp} rounds the target state is prepared as expected.

- In Section 3.5.2 we formalize a variant of preRSPV where the client counts *scores* for the server besides the pass/fail flag. The set-up is as follows: in the end of the protocol the client will record a win/lose decision in a $\boldsymbol{score}$ register, and the honest server could achieve the optimal winning probability. We use OPT to denote the optimal winning probability. (Note that when OPT $= 1$ the notion downgrades to the notion in Section 3.5.1.) The soundness is roughly defined as follows: if π_{test} running against an adversary Adv could win with probability close to OPT, then π_{comp} running against the same adversary Adv will satisfy the RSPV soundness.

  In the amplification procedure, the client needs to count the number of winning rounds and see whether the total score is close to the expected value of an honest execution.

- In Section 3.5.3 we formalize a temporary variant of preRSPV where there are initial resource states (instead of preparing states from scratch), which is used in later sections.

In the constructions in later sections, we often need to first construct a preRSPV, and then amplify it to an RSPV. (See Section 6.3, 6.5.)

3.5.1 PreRSPV

Let's formalize the preRSPV notion. In this section we work in a set-up of Set-up 1.

Set-up 4 (Set-up of preRSPV)**.** The set-up is the same as Set-up 1 except that we are considering a pair of protocols $(\pi_{\text{test}}, \pi_{\text{comp}})$ in this set-up instead of solely π.

Below we formalize the completeness and soundness.

Definition 3.9 (Completeness of preRSPV)**.** We say $(\pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 1 is complete if:

- In π_{test}, when the server is honest, the passing probability (the trace of the output state projected onto Π_{pass}) is negligibly close to 1.
- In π_{comp}, when the server is honest, the output state of the protocol is negligibly close to

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \otimes \underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}}$$

  as described in equation (7)(8).

Definition 3.10 (Soundness of preRSPV)**.** We say $(\pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 1 is (δ, ϵ)-sound if:

For any efficient quantum adversary Adv, there exists an efficient quantum operation Sim such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

If

$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi_{\text{test}}^{\mathsf{Adv}}(\rho_0))) \geq 1 - \delta$$

then

$$\Pi_{\mathsf{pass}}(\pi_{\text{comp}}^{\mathsf{Adv}}(\rho_0)) \approx_{\epsilon}^{ind} \underbrace{\Pi_{\mathsf{pass}}}_{\text{on } \boldsymbol{flag}} (\underbrace{\mathsf{Sim}}_{\text{on } \boldsymbol{S},\boldsymbol{Q},\boldsymbol{flag}} (\underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}} \otimes \underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}}))$$

Amplification to RSPV As discussed before, a preRSPV defined above could be amplified to an RSPV.

Below we assume $(\pi_{\text{test}}, \pi_{\text{comp}})$ is a preRSPV under Set-up 1 that is complete and (δ, ϵ_0)-sound.

Protocol 3 (Amplification from preRSPV to RSPV)**.** An RSPV protocol under Set-up 1 is constructed as follows.

Parameters: approximation error parameter $1^{1/\epsilon}$, security parameter 1^{κ}. It is required that $\epsilon > \epsilon_0$.

Output registers: client-side classical registers $\boldsymbol{flag}^{(\text{out})}, \boldsymbol{D}^{(\text{out})}$, server-side quantum register $\boldsymbol{Q}^{(\text{out})}$.

1. Define $L = \frac{512}{\delta(\epsilon-\epsilon_0)^3}$, $p = \frac{\epsilon-\epsilon_0}{8}$.

   For each $i \in [L]$:

   (a) The client randomly chooses $\text{mode}^{(i)} = \text{test}$ with probability p and $\text{mode}^{(i)} = \text{comp}$ with probability $1 - p$.

   (b) The client executes $\pi_{\text{mode}^{(i)}}$ with the server. Store the outputs in registers $\boldsymbol{flag}^{(i)}, \boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$.

2. The client sets $\boldsymbol{flag}^{(\text{out})}$ to be fail if any one of $\boldsymbol{flag}^{(i)}$ is fail. Otherwise it sets $\boldsymbol{flag}^{(\text{out})}$ to be pass.

   The client randomly chooses $i \in [L]$ such that $\text{mode}^{(i)} = \text{comp}$ and sends i to the server. Both parties use the states in $\boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$ as the outputs.

Then we show the protocol above is an RSPV. The completeness and efficiency are from the protocol; below we prove the soundness.

Theorem 3.4. *Protocol 3 is ϵ-sound.*

The proof has some similarities with the proof of Theorem 3.1.

Proof. Consider an adversary Adv. Denote the initial state as ρ_0, and denote the output state by the end of the i-th round of step 1 as ρ_i. Then by the soundness of preRSPV we get, for any $i \in [L]$, there exists an efficient simulator Sim_i such that if

$$\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\pi_{\text{test}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})))) > (1-\delta)\,\mathrm{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})) \tag{20}$$

then

$$\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\pi_{\mathrm{comp}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1}))) \approx_{\epsilon_0}^{ind} \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\mathsf{Sim}_i(\underbrace{\rho_{tar}}_{\boldsymbol{D}^{(i)},\boldsymbol{Q}^{(i)}} \otimes \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1}))) \tag{21}$$

Define $S_{\text{low pass}}$ as the set of i such that (20) does not hold (thus (21) holds for the complement of $S_{\text{low pass}}$). Then we note that when the size of $S_{\text{low pass}}$ is $\geq \frac{8}{p\delta(\epsilon-\epsilon_0)}$ the overall passing probability is $\leq \frac{\epsilon-\epsilon_0}{8}$.

The simulator Sim applied on $(\rho_{tar} \otimes \rho_0)$ is defined as follows:

1. Sample a random coin $i \leftarrow [L]$. This is for simulating the client's random choice in the second step of Protocol 3.
2. Run $\tilde{\pi}_{<1.i}$ on ρ_0 and get $\tilde{\rho}_{i-1}$. Here $\tilde{\pi}_{<1.i}$ denotes the simulated protocol execution of Protocol 3 until the beginning of the i-th round of the first step. Here "simulated protocol execution" means that, instead of interacting with the client, the simulator simulates a client on its own. So the difference of ρ_{i-1} and $\tilde{\rho}_{i-1}$ is only the locations of these "client-side registers".
3. Run Sim_i on $\rho_{tar} \otimes \tilde{\rho}_{i-1}$ where $\rho_{tar} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\mathrm{out})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\mathrm{out})}})$.
4. Run $\tilde{\pi}_{>1.i}$ on $\mathsf{Sim}_i(\rho_{tar} \otimes \tilde{\rho}_{i-1})$. Here $\tilde{\pi}_{>1.i}$ is defined as above.
5. Set $\boldsymbol{flag}^{(\mathrm{out})}$ to be fail if any of the flag or simulated flag is fail; otherwise set $\boldsymbol{flag}^{(\mathrm{out})}$ to be pass. Disgard all the auxiliary registers for simulating the client side.

We prove this simulator achieves what we want. We could compare the simulated output states with the output states from the real execution and see where the distinguishing advantage could come from (see also the proof of Theorem 3.1):

- In the original protocol i is not sampled randomly but on all the *comp* rounds, while in the simulation i is sampled randomly from $[L]$. But in the original protocol with high probability the number of *test* rounds is at most $2pL$ so this part contributes an error of at most $\frac{\epsilon-\epsilon_0}{8}$ on both size of equation (9).
- When $i \notin S_{\text{low pass}}$, (21) holds, which gives the indistinguishabilty for the simulated execution and real execution. Equation (21) itself contributes an error of ϵ_0.
- The i-th round is not necessarily a *comp* round, but when we simulate the i-th round we simply assume it's a *comp* round for simulating it using (21). This contributes an error of p on both sides of equation (9).
- When $i \in S_{\text{low pass}}$ (21) does not hold. Although we do not know the size of $S_{\text{low pass}}$, we could divide this set into S_{head} and S_{tail} where S_{head} is the set of the first $\frac{8}{p\delta(\epsilon-\epsilon_0)}$ elements and S_{tail} is the set of the remaining elements. Below we bound the effect of these two sets on the distinguishing advantage.
  - The size of S_{head} is at most $\frac{8}{p\delta(\epsilon-\epsilon_0)}$ thus it contributes an error of $\frac{8}{Lp\delta(\epsilon-\epsilon_0)}$ on both size of equation (9).
  - For $i \in S_{\text{tail}}$ the passing probability is no more than $\frac{\epsilon-\epsilon_0}{8}$ thus this part contributes at most an error of $\frac{\epsilon-\epsilon_0}{8}$ on both size of equation (9).

Summing them up we get the total approximation error to be bounded by $\epsilon_0+2(\frac{\epsilon-\epsilon_0}{8}+p+\frac{8}{Lp\delta(\epsilon-\epsilon_0)}+\frac{\epsilon-\epsilon_0}{8})\leq\epsilon$. □

3.5.2 PreRSPV with the score

As discussed before, we need to consider a variant of preRSPV where the client counts scores to certify the server's operation. There will be a $\boldsymbol{score}$ register and a optimal winning probability OPT in the set-up of this notion. Below we formalize the set-up, completeness and soundness.

Set-up 5 (Set-up for a preRSPV with the score)**.** Compare to Set-up 4, we consider a real number $\mathrm{OPT}\in[0,1]$.

Then we consider an additional client-side classical register $\boldsymbol{score}$ with value in $\{\mathsf{win},\mathsf{lose},\perp\}$. π_{test} outputs a value in $\{\mathsf{win},\mathsf{lose}\}$ into the $\boldsymbol{score}$ register; π_{comp} does not touch the $\boldsymbol{score}$ register and leave it in the default value $\perp$.

The other parts of the set-up are the same as Set-up 4.

Definition 3.11. We say $(\pi_{\mathrm{test}},\pi_{\mathrm{comp}})$ under Set-up 5 is complete if:

- In π_{test}, when the server is honest, the passing probability (the trace of the output state projected onto Π_{pass}) is negligibly close to 1 and the winning probability (the trace of the output state projected onto $\Pi_{\mathsf{pass}}\Pi_{\mathsf{win}}^{\boldsymbol{score}}$) is negligibly close to OPT.
- In π_{comp}, when the server is honest, the output state of the protocol is negligibly close to

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}}\otimes\underbrace{|\perp\rangle\langle\perp|}_{\boldsymbol{score}}\otimes\underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}}$$

  as described in equation (7)(8).

Definition 3.12. For $(\pi_{\mathrm{test}},\pi_{\mathrm{comp}})$ under Set-up 5, we say winning probability OPT is (δ,λ)-optimal if:

For any efficient quantum adversary Adv, for any state $\rho_0\in\mathrm{D}(\mathcal{H}_{\boldsymbol{S}}\otimes\mathcal{H}_{\boldsymbol{E}})$, at least one of the following two is true:

- (Low passing)
$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi_{\mathrm{test}}^{\mathsf{Adv}}(\rho_0)))\leq 1-\delta+\mathsf{negl}(\kappa) \tag{22}$$
- (Optimal winning)
$$\mathrm{tr}(\Pi_{\mathsf{pass}}\Pi_{\mathsf{win}}^{\boldsymbol{score}}(\pi_{\mathrm{test}}^{\mathsf{Adv}}(\rho_0)))\leq \mathrm{OPT}+\lambda+\mathsf{negl}(\kappa) \tag{23}$$

Definition 3.13. We say $(\pi_{\mathrm{test}},\pi_{\mathrm{comp}})$ under Set-up 5 is (δ,ϵ)-sound if:

For any efficient quantum adversary Adv, there exists an efficient quantum operation Sim such that for any state $\rho_0\in\mathrm{D}(\mathcal{H}_{\boldsymbol{S}}\otimes\mathcal{H}_{\boldsymbol{E}})$, at least one of the following three is true:

- (Low passing)
$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi_{\mathrm{test}}^{\mathsf{Adv}}(\rho_0)))\leq 1-\delta+\mathsf{negl}(\kappa)$$
- (Low winning)
$$\mathrm{tr}(\Pi_{\mathsf{pass}}\Pi_{\mathsf{win}}^{\boldsymbol{score}}(\pi_{\mathrm{test}}^{\mathsf{Adv}}(\rho_0)))\leq \mathrm{OPT}-\delta$$
- (Simulation)
$$\Pi_{\mathsf{pass}}(\pi_{\mathrm{comp}}^{\mathsf{Adv}}(\rho_0))\approx_\epsilon^{ind}\underbrace{\Pi_{\mathsf{pass}}}_{\text{on }\boldsymbol{flag}}\Big(\underbrace{\mathsf{Sim}}_{\text{on }\boldsymbol{S},\boldsymbol{Q},\boldsymbol{flag}}\big(\underbrace{\rho_{tar}}_{\boldsymbol{D},\boldsymbol{Q}}\otimes\underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}}\big)\Big)$$

Variants of Definition 3.12, 3.13 We note that there are also other ways of expressing the optimality of OPT and the soundness; for example, we could only remove the (22) part (and only keep the (23) part) and remove the parameter δ from the definition. The choices of these definitions are based on the consideration of keeping a balance between the difficulty of the preRSPV-with-the-score-to-RSPV amplification process and the difficulty of the construction of the preRSPV-with-the-score protocol. (Note that when the soundness/optimality gets weaker the construction gets easier but the amplification gets harder.)

Amplification to RSPV Similar to Section 3.5.1, the preRSPV under Set-up 5 could also be amplified to an RSPV. Below we do this amplification in two steps: we first amplify a preRSPV under Set-up 5 to a preRSPV under Set-up 4 (that is, removing the need for the score in the modeling). Then we use the results in Section 3.5.1 to get an RSPV protocol.

Below we assume $(\pi_{\text{test}}, \pi_{\text{comp}})$ is a preRSPV under Set-up 5 that is complete, has (δ_0, λ)-optimal winning probability OPT and is (δ_0, ϵ_0)-sound.

Protocol 4 (Amplification from preRSPV under Set-up 5 to preRSPV under Set-up 4)**.** A preRSPV protocol $(\pi'_{\text{test}}, \pi'_{\text{comp}})$ under Set-up 4 is constructed as follows.

Parameters: approximation error parameter $1^{1/\epsilon}$, security parameter 1^{κ}. It is required that $\epsilon > \epsilon_0$ and $\lambda < \frac{1}{6}\delta_0(\epsilon - \epsilon_0)$.

Output registers: client-side classical registers $\boldsymbol{flag}^{(\text{out})}, \boldsymbol{D}^{(\text{out})}$, server-side quantum register $\boldsymbol{Q}^{(\text{out})}$.

Define $L = 4\kappa/(((\frac{1}{6}\delta_0(\epsilon - \epsilon_0) - \lambda))^2(\epsilon - \epsilon_0))$, threshold $= (\text{OPT} - \frac{1}{2}(\frac{1}{6}\delta_0(\epsilon - \epsilon_0) - \lambda))L$.

π'_{test} is defined as follows:

1. For each $i \in [L]$:
   (a) The client executes π_{test} with the server. Store the outputs in registers $\boldsymbol{flag}^{(i)}, \boldsymbol{score}^{(i)}, \boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$.
2. The client sets $\boldsymbol{flag}^{(\text{out})}$ to be fail if any one of $\boldsymbol{flag}^{(i)}$ is fail. Then the client counts the number of win in all the $\boldsymbol{score}$ registers and sets $\boldsymbol{flag}^{(\text{out})}$ to be fail if the total number of win is less than threshold. Otherwise it sets $\boldsymbol{flag}^{(\text{out})}$ to be pass.

π'_{comp} is defined as follows:

1. The client randomly chooses $i_{stop} \in [L]$.
   For each $i \in [i_{stop} - 1]$:
   (a) The client executes π_{test} with the server. Store the outputs in registers $\boldsymbol{flag}^{(i)}, \boldsymbol{score}^{(i)}, \boldsymbol{D}^{(i)}, \boldsymbol{Q}^{(i)}$.
2. The client sets $\boldsymbol{flag}^{(\text{out})}$ to be fail if any $\boldsymbol{flag}^{(i)}$ is fail. Otherwise the client executes π_{comp} with the server; store the outputs in registers $\boldsymbol{flag}^{(\text{out})}, \boldsymbol{D}^{(\text{out})}, \boldsymbol{Q}^{(\text{out})}$.

Theorem 3.5. *The protocol is complete.*

Proof. Note that the expected value of the total score in π'_{test} is $\text{OPT} \cdot L$. By the Chernoff bounds the probability of "the total score is less than $(\text{OPT} - \frac{1}{2}(\frac{1}{6}\delta_0(\epsilon - \epsilon_0) - \lambda))L$" is upper bounded by a negligible function of κ.

The other sources of not passing are all negligible. □

The efficiency is from the protocol; below we prove the soundness.

Theorem 3.6. *Protocol 4 is (δ, ϵ)-sound where $\delta = \frac{1}{2}(\frac{1}{6}\delta_0(\epsilon - \epsilon_0) - \lambda)$.*

Intuition for the proof The overall structure of the proof has some similarity with the proof of Theorem 3.4. One main technical obstacle is how to do the probability calculation for the random process of generating the scores: we would like to show that if the server could win in sufficiently big number of tests, in a large portion of the rounds, the probability that it wins should be big — which implies that the corresponding comp mode should prepare the target state. Formalizing these probability arguments requires some techniques from the probability theory; here we use the Markov inequality[13] and make careful analysis.

Proof. Consider an adversary Adv. Denote the initial state as ρ_0, and denote the output state by the end of the i-th round of step 1 as ρ_i. Then by the soundness of $(\pi_{\text{test}}, \pi_{\text{comp}})$ we get, for any $i \in [L]$, there exists an efficient simulator Sim_i such that at least one of the following three is true:

- (Low passing)

$$\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\pi_{\text{test}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})))) \leq (1-\delta_0)\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})) \tag{24}$$

- (Low winning)

$$\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}\Pi_{\mathsf{win}}^{\boldsymbol{score}^{(i)}}(\pi_{\text{test}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})))) \leq (\text{OPT}-\delta_0)\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})) \tag{25}$$

- (Simulation)

$$\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\pi_{\text{comp}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1}))) \approx_{\epsilon_0}^{ind} \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}(\mathsf{Sim}_i(\underbrace{\rho_{tar}}_{\boldsymbol{D}^{(i)},\boldsymbol{Q}^{(i)}} \otimes \Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1}))) \tag{26}$$

Define $S_{\text{low pass}}$ as the set of i such that (24) holds. By the condition that the overall protocol passes with probability $\geq 1-\delta$, we have that $|S_{\text{low pass}}| \leq \frac{1}{36}(\frac{1}{6}\delta_0(\epsilon-\epsilon_0)-\lambda)L$.

Define $S_{\text{low win}}$ as the set of i such that (25) holds (thus (26) holds for the complement of $S_{\text{low pass}} \cup S_{\text{low win}}$). Below we need to bound the total score on both the $S_{\text{low win}}$ part and its complement. For upper bounding the scores on the complement of $S_{\text{low win}}$ we use the optimality of OPT, which translates to:

For each $i \in [L] - S_{\text{low pass}}$, there is

$$\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(i)}}\Pi_{\mathsf{win}}^{\boldsymbol{score}^{(i)}}(\pi_{\text{test}}^{\mathsf{Adv}_i}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})))) \leq (\text{OPT}+\lambda)\cdot\operatorname{tr}(\Pi_{\mathsf{pass}}^{\boldsymbol{flag}^{(\leq i-1)}}(\rho_{i-1})) \tag{27}$$

Below we show the passing probability could not be high under the condition that $|S_{\text{low win}}| \geq \frac{1}{6}(\epsilon-\epsilon_0)L$. The total expectation for the scores in each step is at most

$$\underbrace{\frac{1}{6}(\epsilon-\epsilon_0)L\cdot(\text{OPT}-\delta_0)}_{\text{rounds in } S_{\text{low win}}} + \underbrace{(L-\frac{1}{6}(\epsilon-\epsilon_0)L)\cdot(\text{OPT}+\lambda)}_{\text{rounds outside } S_{\text{low win}}} \leq (\text{OPT}-(\frac{1}{6}\delta_0(\epsilon-\epsilon_0)-\lambda))L$$

[13] In general the Markov inequality is loose but here we do not care about the tightness of the calculation. For a tighter analysis we could consider, for example, the Azuma's inequality (which also has more restrictions).

which is smaller than the threshold by a significant gap. By the Markov inequality we have the passing probability is no more than $1 - \frac{1}{2}(\frac{1}{6}\delta_0(\epsilon - \epsilon_0) - \lambda)/\mathrm{OPT} < 1 - \delta$, which violates condition of the overall passing probability. In summary, we have proved:

$$|S_{\text{low win}}| \leq \frac{1}{6}(\epsilon - \epsilon_0)L$$

Let's construct the simulator for the π'_{comp} mode and show that it achieves what we need. The simulator Sim applied on $(\rho_{tar} \otimes \rho_0)$ is defined as follows:

1. Sample a random coin $i_{\text{stop}} \leftarrow [L]$.
2. Simulate the first $(i_{\text{stop}} - 1)$ rounds of π'_{comp} to get $\tilde{\rho}_{i_{\text{stop}}-1}$. Here the simulation is similar to what we did in the proof of Theorem 3.4; so the difference of $\rho_{i_{\text{stop}}-1}$ and $\tilde{\rho}_{i_{\text{stop}}-1}$ is only the locations of the "client-side registers".
3. Run Sim_i on $\rho_{tar} \otimes \tilde{\rho}_{i_{\text{stop}}-1}$ where $\rho_{tar} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{out})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{out})}})$.
4. Set $\boldsymbol{flag}^{(\text{out})}$ to be fail if any of the flag or simulated flag is fail; otherwise set $\boldsymbol{flag}^{(\text{out})}$ to be pass. Disgard all the auxiliary registers for simulating the client side.

We prove this simulator simulates the output of π'_{comp}. We could compare the simulated output states with the output states from the real execution and see where the distinguishing advantage (or called approximation error) could come from:

- Equation (26) itself contributes an error of ϵ_0.
- Equation (26) only holds for indices in the complement of $S_{\text{low pass}} \cup S_{\text{low win}}$, but i_{stop} may still fall within $S_{\text{low pass}}$. This contributes an error of no more than $\frac{1}{6}(\epsilon - \epsilon_0) \times 2$.
- Equation (26) only holds for indices in the complement of $S_{\text{low pass}} \cup S_{\text{low win}}$, but i_{stop} may still fall within $S_{\text{low win}}$. This contributes an error of $\frac{1}{6}(\epsilon - \epsilon_0) \times 2$.

Summing them up completes the proof. □

After getting a preRSPV under Set-up 4, we could further amplify it using Protocol 3 to get an RSPV.

3.5.3 A temporary variant of PreRSPV with initial resource states

In the later sections, as an intermediate notion, we need to consider a variant of preRSPV that starts with some resource states (recall the discussion on resource states in Section 3.3.1). These states are assumed to be in the honest form in both the honest setting and malicious setting. Below we formalize this variant, as a preparation for later constructions.

Set-up 6 (Variant of Set-up 1 with initial resource states)**.** The registers are as follows: the client holds a classical register $\boldsymbol{D}^{(\text{in})}$ and the server holds a quantum register $\boldsymbol{Q}^{(\text{in})}$. The client also holds a classical register $\boldsymbol{D}^{(\text{out})}$ and the server also holds a quantum register $\boldsymbol{Q}^{(\text{out})}$, which are both renamed from $\boldsymbol{D}, \boldsymbol{Q}$ in Set-up 1. We also need to consider a state $\rho_{rs} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{in})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{in})}})$ which is used as part of the initial state in in the honest setting. The other parts of the set-up are the same as Set-up 1.

The completeness is defined in the same way as Definition 3.9 with one difference: when we consider the execution of π_{test} and π_{comp} the initial state should be ρ_{rs}. The soundness is defined in the same way as Definition 3.10 with the following differences differences: when we consider the execution of π_{test} and π_{comp} the initial state should be $\rho_{rs} \otimes \rho_0$; and the registers that the simulator could work on now consist of $\boldsymbol{S}, \boldsymbol{Q}^{(\text{in})}, \boldsymbol{Q}^{(\text{out})}, \boldsymbol{flag}$.

Set-up 7 (Variant of Set-up 5 with initial resource states)**.** Similar to what we did in Set-up 6, we consider registers $\boldsymbol{D}^{(\text{in})}$, $\boldsymbol{Q}^{(\text{in})}$ in the set-up and rename $\boldsymbol{D}, \boldsymbol{Q}$ as $\boldsymbol{D}^{(\text{out})}$, $\boldsymbol{Q}^{(\text{out})}$.

The completeness and soundness are adapted similarly.

Compilation to the normal form of preRSPV We note that once we get a protocol under Set-up 6, 7, we could compile the protocol following the approach in Section 3.4.2: we could use an RSPV with ρ_{rs} being the target state to prepare the resource states, then the overall protocol will be a preRSPV under the definitions in Section 3.5.1, 3.5.2. Then we could apply the results in Section 3.5.1, 3.5.2 to amplify them to an RSPV.

4 RSPV and Test of a Qubit

One basic primitive in RSPV is the RSPV for BB84 states. Basically this primitive has been studied in several existing works [29, 25, 17] but these existing works could use different frameworks and notions from our work so some translation is needed. Existing ways to express the relations between operators or states include:

- Characterize the underlying states: if the test passes with high probability, then the states to be tested are as desired. This could further include rigidity-based soundness and simulation-based soundness.
- Characterize the measurement operators: if the test passes with high probability, then the operators used by the server in the test are as desired. This further includes:
  - the operators satisfy, for example, anticommutation relations.
  - the operators are, for example, the Pauli X and Z operators up to an isometry. This is similar to the rigidity-based soundness discussed before.

In this section we study the relation between RSPV and a notion called "test of a qubit", and study the protocols for RSPV for BB84 states.

By [17] we know that we could construct a test for a qubit assuming NTCF (which could be adapted to weak NTCF). Although test for a qubit is closely related to RSPV, they are still not the same[14], so some technical works are needed for translation of results. A test for a qubit is different from an RSPV in the following ways:

- The soundness of a test for a qubit is described in terms of anticommutation of operators [17]; the soundness of RSPV is described by simulation, defined on states.
- In a test for a qubit the qubit is finally tested and destroyed; in RSPV we want to preserve the qubit on the server side.

[14]We thank Kaniuar Bacho, James Bartusek, Yasuaki Okinaka and anonymous reviewers for discussions

We would like to translate the results in the language of test-of-a-qubit to our RSPV framework. We will not start from scratch, but try to make use of existing results as much as possible. Below we describe our approach in more detail.

- The test of a qubit protocol could be regarded as a preRSPV-with-score (see Section 3.5.2), which could be amplified to an RSPV. This solves the second issue above.
- We translate the soundness property as follows. Starting from the anticommutation relation, we know the server's operators are actually Pauli X and Pauli Z operators up to an efficiently computable isometry. Then this together with the fact that the server could win the test with close-to-optimal probability implies that the underlying states are as desired. Finally we use the (computational) basis-blindness to prove the indistinguishability of remaining parts of the states which implies the simulation-based soundness.

In Section 4.1 we first review the results in [17]; in Section 4.2 we give our construction for RSPV-for-BB84 and prove its soundness.

4.1 Test of a Qubit

The protocol in [17], instantiated with weak NTCF, could be described as follows.

Protocol 5. Below we describe the test of a qubit protocol as given in [17].

Parameters: completeness error tolerance parameter $1^{1/\mu}$, soundness error $1^{1/\delta}$, security parameter 1^{κ}.

1. (Phase A) The client and the server perform some protocol.

   In the end the joint state between the client and the server in the honest setting is μ-close to the following state: the client-side $\boldsymbol{flag}$ register holds value pass, the client holds $\theta_1, \theta_2 \leftarrow_r \{0,1\}^2$, the server holds $|+_{4\theta_1+2\theta_2+1}\rangle$.

   Break out from the protocol if the $\boldsymbol{flag}$ is fail.

2. (Phase B) The client randomly samples $c \leftarrow_r \{0,2\}$ and sends it to the server.

   The server measures the state on basis $|+_c\rangle, |+_{c+4}\rangle$ and send back the measurement result to the client. The client checks the server's response is $u_c(4\theta_1 + 2\theta_2 + 1)$ (see Definition 4.1) and record $\boldsymbol{score}$ to be win if the check passes.

Note that we re-describe the protocol to be consistent with the notations in later sections.

By [17], the following holds for Protocol 5:

Theorem 4.1 (Computational basis-blindness)**.** *Suppose the server-side state in the end of phase A corresponding to client-side value $\theta = 4\theta_1 + 2\theta_2 + 1$ is ρ_θ. Then $\rho_1 + \rho_5 \approx^{ind} \rho_3 + \rho_7$.*

Theorem 4.2. *Suppose the server passes the protocol with probability $\geq 1 - \delta$. Then the winning probability is no more than $\cos^2(\pi/8) + \mathsf{poly}(\delta)$.*

Theorem 4.3. *Suppose the server passes the protocol with probability $\geq 1 - \delta$ and wins with probability $\geq \cos^2(\pi/8) - \delta$. Suppose $\rho = \sum_{\theta\in\{1,3,5,7\}} \rho_\theta$ where ρ_θ is defined in Theorem 4.1. Suppose the server's operations corresponding to $c = 0$ and $c = 2$ are described by observables X_0, X_2. Then*

$$\mathrm{tr}(\{X_0, X_2\}^2 \rho) \leq \mathsf{poly}(\delta) \tag{28}$$

4.2 RSPV for BB84 States

Below in Section 4.2.1 and 4.2.2 we first review the lemmas needed for translating test of a qubit to RSPV for BB84 states; then in Section 4.2.3 we construct and prove the soundness of RSPV for BB84 states.

4.2.1 Extracting the qubit from the anticommutation relation

From [29, 54] we know that the anticommutation implies that the operators could be seen as the Pauli X and Pauli Z operators up to an efficiently computable isometry. Below we review the statement (with a basis rotation for consistency to later sections; note that $|+_0\rangle, |+_2\rangle, |+_4\rangle, |+_6\rangle$ are isometric to BB84 states.)

Theorem 4.4 (By [54])**.** *Suppose X_0, X_2, ρ satisfies*

$$\mathrm{tr}(\{X_0, X_2\}^2\rho) \approx_{O(\delta)} 0$$

Then there exists a quantum isometry U^{X_0,X_2} efficiently computable from X_0, X_2 such that

$$X_0(\rho) \approx_{\mathsf{poly}(\delta)} ((U^{X_0,X_2})^\dagger(|+_0\rangle\langle+_0| - |+_4\rangle\langle+_4|)U^{X_0,X_2})(\rho)$$

$$X_2(\rho) \approx_{\mathsf{poly}(\delta)} ((U^{X_0,X_2})^\dagger(|+_2\rangle\langle+_2| - |+_6\rangle\langle+_6|)U^{X_0,X_2})(\rho)$$

Theorem 4.4 allows us characterize the operators. In the next section we state a lemma that allows us to further characterize the underlying states to be measured (together with the condition that the server wins the test with close-to-optimal probability).

4.2.2 Lemmas for characterizing the states to be tested

Lemma 4.5. *For any density operator ρ there is $\frac{1}{2}(\mathrm{tr}(|+_0\rangle\langle+_0|\rho) + \mathrm{tr}(|+_2\rangle\langle+_2|\rho)) \le \cos^2(\pi/8)$. And if $\frac{1}{2}(\mathrm{tr}(|+_0\rangle\langle+_0|\rho) + \mathrm{tr}(|+_2\rangle\langle+_2|\rho)) \approx_\delta \cos^2(\pi/8)$, there is $\rho \approx_{\mathsf{poly}(\delta)} |+_1\rangle\langle+_1| \otimes \psi$ for some density operator ψ.*

The proof is a simple linear algebra calculation (see also the appendix of [57]).

The following lemma is for proving the indistinguishability on the remaining part of the states (other than the qubits to be measured). We first state the statistical indistinguishability version of it.

Lemma 4.6. *Suppose $\rho_{\theta'} + \rho_{\theta'+4} \approx_\delta \rho_{\theta''} + \rho_{\theta''+4}$ where $\theta' \neq \theta''$. Further suppose $\rho_\theta = |+_\theta\rangle\langle+_\theta| \otimes \psi_\theta$ for each $\theta \in \{\theta', \theta'', \theta'+4, \theta''+4\}$. Then $(\psi_\theta)_{\theta\in\{\theta',\theta'',\theta'+4,\theta''+4\}}$ are $\mathsf{poly}(\delta)$-close to each other.*

Then we state the computational analog of Lemma 4.6.

Lemma 4.7. *Suppose $\rho_{\theta'} + \rho_{\theta'+4} \approx^{ind}_\delta \rho_{\theta''} + \rho_{\theta''+4}$ where $\theta' \neq \theta''$. Further suppose $\rho_\theta = |+_\theta\rangle\langle+_\theta| \otimes \psi_\theta$ for each $\theta \in \{\theta', \theta'', \theta'+4, \theta''+4\}$. Then $(\psi_\theta)_{\theta\in\{\theta',\theta'',\theta'+4,\theta''+4\}}$ are $\mathsf{poly}(\delta)$-indistinguishable to each other.*

Note that, similar to Lemma 4.6 we do not have the condition that these ρ have approximately the same trace value (but this could be proved from the condition).

4.2.3 Constructions and proofs for RSPV for BB84

Below we first prove Protocol 5 could be seen as a preRSPV-with-score.

Lemma 4.8. *Consider (π_{test}, π_{comp}) where π_{test} is the whole protocol of Protocol 5, and the computation-mode is the phase A part. Then this protocol is a preRSPV-with-score with optimal winning probability $\cos^2(\pi/8)$ and is $(\delta, \mathsf{poly}(\delta))$-sound.*

The optimality of winning probability $\cos^2(\pi/8)$ is proved in [17] and does not need translation.

Proof of Lemma 4.8. First apply Theorem 4.3 we know if the server passes the protocol with probability $\geq 1-\delta$ and wins with probability $\geq \cos^2(\pi/8)-\delta$, (28) holds. Apply Theorem 4.4 we know X_0, X_2 are efficiently isometric to $|+_0\rangle\langle+_0| - |+_4\rangle\langle+_4|$ and $|+_2\rangle\langle+_2| - |+_6\rangle\langle+_6|$. Use the condition that the server wins with probability $\geq \cos^2(\pi/8)-\delta$ again, by Lemma 4.5 we know $\rho_\theta \approx_{\mathsf{poly}(\delta)} |+_\theta\rangle\langle+_\theta| \otimes \psi_\theta$ for each $\theta \in \{1,3,5,7\}$. Finally by Lemma 4.7 ψ_θ are approximately indistinguishable to each other, which proves the rigidity-based soundness and implies the simulation-based soundness. □

Then this preRSPV-with-score could be amplified to an RSPV for BB84:

Theorem 4.9. *Under the same assumptions as Protocol 5, there exists an RSPV for BB84 states.*

Proof. Apply the amplification described in Section 3.5.2 to the preRSPV-with-score described above completes the proof. □

4.3 Detailed Description of the Information-theoretic Core for Test of a Qubit

In this section we give the missing details when we describe the test of a qubit in Protocol 5. Here we describe the information-theoretic core, which will be used in later sections. Note that although [17] proves stronger results than this information-theoretic core (in more detail, they only need computational basis-blindness instead of statistical basis-blindness), an explicit discussion of the information-theoretic core makes our work and the analysis later more self-contained and accessible.

Definition 4.1 (Repeat of Definition 3.1 in [29])**.** Consider four positive semi-definite operators $(\phi_\theta)_{\theta\in\{1,3,5,7\}}$ and two single-qubit observables X_0 and X_2. Recall that for single-qubit observables the eigenvalues are ±1; for each $i \in \{0,2\}$, use X_i^0 to denote the projection onto the eigenvector of X_i with eigenvalue $+1$, and use X_i^1 to denote the projection onto the space of eigenvector of X_i with eigenvalue -1. Thus $X_i = X_i^0 - X_i^1$.

For $\theta \in \{1,3,5,7\}$ let $u_0(\theta), u_2(\theta) \in \{0,1\}$ be functions as follows:

- $u_0(\theta) = 0$ if and only if $\theta \in \{1,7\}$;
- $u_2(\theta) = 0$ if and only if $\theta \in \{1,3\}$.

The winning probability is

$$\frac{1}{4}\sum_{\theta\in\{1,3,5,7\}} \frac{1}{2}\sum_{i\in\{0,2\}} \mathrm{tr}\left(X_i^{u_i(\theta)}\phi_\theta\right). \tag{29}$$

We give some explanation of the definition. The special choice of indexing is for matching the main protocol in [29]. $\theta \in \{1,3,5,7\}$ encodes two classical bits, which are $u_0(\theta)$ and $u_2(\theta)$. The correspondence is as follows:

- $\theta = 1$: $u_0(\theta) = 0, u_2(\theta) = 0$;
- $\theta = 3$: $u_0(\theta) = 1, u_2(\theta) = 0$;
- $\theta = 5$: $u_0(\theta) = 1, u_2(\theta) = 1$;
- $\theta = 7$: $u_0(\theta) = 0, u_2(\theta) = 1$.

The following states and observables achieve the optimal winning probability $\cos^2(\pi/8)$. From this example we could also get an intuitive interpretation of the test and the indices.

Fact 5. *Recall*

$$|+_\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\mathrm{i}\theta}|1\rangle), \theta \in \{0,1,2\cdots 7\}$$

Then under the following $(\phi_\theta)_{\theta\in\{1,3,5,7\}}$ and X_0, X_2, the winning probability is $\cos^2(\pi/8)$:

$$\phi_\theta = |+_\theta\rangle\langle+_\theta|, \quad \forall\theta \in \{1,3,5,7\}$$

$$\begin{aligned} X_0^0 &:= \textit{ projection onto } |+_0\rangle \\ X_0^1 &:= \textit{ projection onto } |+_4\rangle \\ X_2^0 &:= \textit{ projection onto } |+_2\rangle \\ X_2^1 &:= \textit{ projection onto } |+_6\rangle \end{aligned}$$

Recall for all $i \in \{0,2\}$, $X_i = X_i^0 - X_i^1$.

Comparison to [29, 25] First we change several changes on notations compared to [29]; for example, compared to Definition 3.1 in [29], we make it explicit that u_0, u_2 are functions from $\{1,3,5,7\}$ to $\{0,1\}$. Then we note that in [29] the self-testing property is based on the trace-1 condition on the initial states; but we feel that it's more convenient to formalize the self-testing property based on the basis-blindness property, which is similar to the approaches in [25] (which is called *blind self-testing* in [25]).

Lemma 4.10. *Let $(\phi_\theta)_{\theta\in\{1,3,5,7\}}$ be a tuple of positive semidefinite operators. For simplicity define $N = \mathrm{tr}(\frac{1}{4}\sum_{\theta\in\{1,3,5,7\}}\phi_\theta)$. Suppose $(\phi_\theta/\mathrm{tr}(\frac{1}{4}\sum_{\theta\in\{1,3,5,7\}}\phi_\theta))_{\theta\in\{1,3,5,7\}}$ are information-theoretic basis-blind (that is, $\phi_1 + \phi_5 \approx_{\mathsf{negl}(\kappa)N} \phi_3 + \phi_7$.) Consider Definition 4.1 defined on this tuple of states. Then (29) is no more than $N(\cos^2(\pi/8) + \mathsf{negl}(\kappa))$. What's more, if (29) is δN-close to $N(\cos^2(\pi/8))$, then*

$$\sum_{\theta\in\{1,3,5,7\}} \mathrm{tr}\left(\{X_0, X_2\}^2\phi_\theta\right) = (O(\delta) + \mathsf{negl}(\kappa)) \cdot N\ .$$

On the proof of Lemma 4.10 One way to prove Lemma 4.10 is to reduce it to the self-testing property of the CHSH game in the non-local game setting; below we call the two prover Alice and Bob and the verifier is simply named as verifier. The self-testing property of the CHSH game could be found in many existing works [45].

To do the reduction, imagine Bob is holding $\phi_\theta \in \{\phi_1, \phi_5\}$ and Alice is holding $u_2(\theta)$. (Then the Bob-side's state, after tracing out the Alice's system, is $\phi_1 + \phi_5$). By the statistical basis-blindness, we know there exists a local operation on Alice's side such that after this operation Bob is holding $\phi_\theta \in \{\phi_3, \phi_7\}$ and Alice is holding $u_2(\theta)$. Then the verifier's question to Alice is whether or not to apply this local operation before the measurement, which will collapse Bob's system to either $\{\phi_1, \phi_5\}$ or $\{\phi_3, \phi_7\}$; the question to Bob is whether to measure the state using observable X_0 or X_2. Then the winning probability of the CHSH game is exactly described by (29) and the self-testing property of the CHSH translates to Lemma 4.10.

5 Remote Operator Application with Verifiability

5.1 Overview

In this section we introduce a new notion named *remote operator application with verifiability* (ROAV), for certifying server's operations.

Definitions of ROAV In Section 5.2 we formalize the notion of ROAV for a POVM $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$. Recall that in Section 1.3.2 we have informally introduced the notion of ROAV, which is defined to be a tuple $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ that certifies server-side operations.

We will also formalize a variant of ROAV where the input register of $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$ is bigger than the server-side of ρ_{test}; so the remaining part will be left to the server to decide. In Section 5.3.2 this part will be used to hold the witness state in a Hamiltonian ground state testing protocol.

Potential Applications of ROAV In Section 5.3 we discuss two potential applications of ROAV. In more detail:

- In Section 5.3.1 we show an approach for constructing RSPV from ROAV.

  Recall that we currently do not have a "universal RSPV", which is an RSPV for general state families. Our approach provides a potential way for constructing more complicated RSPV from simpler RSPV and ROAV that are possibly easier to construct, which potentially makes progress to the problem of constructing universal RSPV.

  In more detail, suppose we would like to construct an RSPV for the target state $\mathcal{E}(\rho)$; but $\mathcal{E}(\rho)$ might have a complicated form and it's not easy to construct RSPV for it directly. In this section we show that this problem could be decomposed into two pieces: the constructions of an ROAV for $\mathcal{E}$ and a specific RSPV that is not related to $\mathcal{E}$.

- In Section 5.3.2 we show an approach for constructing Hamiltonian ground state testing protocol using ROAV. As a review, existing Hamiltonian ground energy testing protocols like [28, 30] has a structure as follows:

  Input: a local XZ-Hamiltonian $H = \sum_i \gamma_i H_i$. We would like to design a protocol do decide whether its ground state energy is $\leq a$ or $\geq b$.

  The honest server gets a witness state w that achieves the ground state energy.

1. Repeat (sequentially or in parallel) the following for many rounds: the client samples a random H_i and uses some protocols to get the measurement results of operator H_i on the server-side state.
2. The client calculates the weighted average of the measurement results and compares it to a, b to decide.

Typically the difficulty in the template above is how to make the server measure the H_i honestly. In this subsection we give a new approach for this problem. We show a protocol that reduces this problem to the following two protocols:

1. An ROAV for tensor products of Bell basis measurements.
2. A specific RSPV for state families that depend on the input Hamiltonian and the test state of the ROAV.

The protocol takes ideas from the Grilo's Hamiltonian testing protocol [30] in the non-local game setting. Roughly speaking, in Grilo's protocol the verifier randomly executes one of two modes of protocols, the operatortest mode and energytest mode, where:

- In operatortest mode the vrifier certifies that the provers do the measurements required. Especially, Prover 2 should do Bell basis measurements to teleport some states to Prover 1.
- In energytest mode the witness state is teleported to Prover 1 and Prover 1 should do the energy testing on the state.

By assuming the existence of ROAV for Bell basis measurements and the RSPV for the states needed (the ρ_{test} in ROAV and a state ρ_{comp} for energy testing), we could take this protocol to the single-server setting. Note that we need a variant of ROAV where the server could prepare part of the input states of the Bell measurements for holding the witness state.

In more detail, in the operatortest mode the client executes π_{test} to certify that the server has perform Bell basis measurements on registers $\boldsymbol{Q}^{(\text{in})}$ and $\boldsymbol{w}$, where $\boldsymbol{Q}^{(\text{in})}$ is used to hold the server side of both ρ_{test} and $|\Phi\rangle$; the state of $\boldsymbol{w}$ is decided by the server. If the server passes in the operatortest mode, by the soundness of ROAV, in the energytest mode a Bell basis measurement is applied on the server side of $|\Phi\rangle$ and $\boldsymbol{w}$, which teleports the state in $\boldsymbol{w}$ to the register $\boldsymbol{P}$.

However, we want a protocol where the client is completely classical so it could not do quantum testing directly on the state in register $\boldsymbol{P}$. However, we could imagine that $\boldsymbol{P}$ has already been measured beforehand and consider the post-measurement state on $\boldsymbol{Q}^{(\text{in})}$; we denote this state as ρ_{comp}. Assuming an RSPV for ρ_{comp}, the client could test the energy in the energytest mode.

Finally we note that we still need to construct ROAV concretely to make these reductions work; this is left to future works.

5.2 Definitions of ROAV

Let's first formalize the set-up of an ROAV protocol.

Set-up 8 (Set-up for an ROAV)**.** An ROAV for a POVM $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$ is in the form of $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ under the following set-up:

The protocols $\pi_{\text{test}}, \pi_{\text{comp}}$ take the following parameters: approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ.

Input registers: the client-side classical register $\boldsymbol{D}^{(\text{in})}$, the server-side quantum register $\boldsymbol{Q}^{(\text{in})}$.

Output registers: the client-side classical register $\boldsymbol{D}^{(\text{out})}$ with value in $[D]$, the server-side quantum register $\boldsymbol{Q}^{(\text{out})}$, and the client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$.

The environment holds a quantum register $\boldsymbol{P}$.

$\rho_{\text{test}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{in})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{in})}})$. $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$ maps states in $\boldsymbol{Q}^{(\text{in})}$ to $\boldsymbol{Q}^{(\text{out})}$. $\dim(\boldsymbol{P}) = \dim(\boldsymbol{Q}^{(\text{in})})$.

Define the state

$$|\Phi\rangle = \frac{1}{\sqrt{D}} \sum_{i \in [D]} \underbrace{|i\rangle}_{\boldsymbol{P}} \otimes \underbrace{|i\rangle}_{\boldsymbol{Q}^{(\text{in})}}. \tag{30}$$

For modeling the initial states in the malicious setting, assume the server-side registers (excluding $\boldsymbol{Q}^{(\text{in})}, \boldsymbol{Q}^{(\text{out})}$) are denoted by register $\boldsymbol{S}$, and assume the environment (excluding $\boldsymbol{P}$) is denoted by register $\boldsymbol{E}$.

The completeness and soundness are defined as follows.

Definition 5.1 (Completeness of ROAV)**.** We say $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 8 is complete if when the server is honest:

- In π_{test} when the initial state is ρ_{test} the passing probability is negligibly close to 1.
- In π_{comp} when the initial state is $|\Phi\rangle$ the output state of the protocol is negligibly close to

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \otimes (\underbrace{\mathbb{I}}_{\text{on } \boldsymbol{P}} \otimes \mathcal{E}_{tar})(\Phi)$$

where

$$\mathcal{E}_{tar}(\underbrace{\cdot}_{\boldsymbol{Q}^{(\text{in})}}) = \sum_{i \in [D]} \underbrace{|i\rangle\langle i|}_{\boldsymbol{D}^{(\text{out})}} \otimes \underbrace{\mathcal{E}_i(\cdot)}_{\boldsymbol{Q}^{(\text{out})}}$$

Definition 5.2 (Soundness of ROAV)**.** We say $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 8 is (δ, ϵ)-sound if:

For any efficient quantum adversary Adv, there exists an efficient quantum operation Sim such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

If

$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi_{\text{test}}^{\mathsf{Adv}}(\rho_{\text{test}} \otimes \rho_0))) \leq 1 - \delta$$

then

$$\Pi_{\mathsf{pass}}(\pi_{\text{comp}}^{\mathsf{Adv}}(\Phi \otimes \rho_0)) \approx_\epsilon^{ind} \underbrace{\Pi_{\mathsf{pass}}}_{\text{on } \boldsymbol{flag}} \Big(\underbrace{\mathsf{Sim}}_{\text{on } \boldsymbol{S}, \boldsymbol{Q}^{(\text{in})}, \boldsymbol{Q}^{(\text{out})}, \boldsymbol{flag}} \Big(\underbrace{\mathcal{E}_{tar}(\Phi)}_{\boldsymbol{D}^{(\text{out})}, \boldsymbol{Q}^{(\text{out})}, \boldsymbol{P}} \otimes \underbrace{\rho_0}_{\boldsymbol{S}, \boldsymbol{E}} \Big) \Big) \tag{31}$$

5.2.1 Variant: ROAV with extra server-side states (besides the EPR parts)

Note that in the definitions above we assume the input register of $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$ is the same as the server-side of ρ_{test} (which are both $\boldsymbol{Q}^{(\text{in})}$). Below we formalize a variant where the input register of $(\mathcal{E}_1, \mathcal{E}_2 \cdots \mathcal{E}_D)$ is bigger than the server-side of ρ_{test}.

Set-up 9 (Set-up for an ROAV with extra server-side states)**.** Compare to Set-up 8, the server holds an additional quantum register $\boldsymbol{w}$. The other parts of the set-up are the same as Set-up 8.

Definition 5.3. We say $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 9 is complete if for any $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{w}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

- In π_{test} when the initial state is $\rho_{\text{test}} \otimes \rho_0$ the passing probability is negligibly close to 1.
- In π_{comp} when the initial state is $\Phi \otimes \rho_0$ the output state of the protocol is negligibly close to

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \otimes (\underbrace{\mathbb{I}}_{\boldsymbol{P}} \otimes \mathcal{E}_{tar} \otimes \underbrace{\mathbb{I}}_{\boldsymbol{E}})(\Phi \otimes \rho_0)$$

where

$$\mathcal{E}_{tar}(\underbrace{\cdot}_{\boldsymbol{Q}^{(\text{in})},\boldsymbol{w}}) = \sum_{i\in[D]} \underbrace{|i\rangle\langle i|}_{\boldsymbol{D}^{(\text{out})}} \otimes \underbrace{\mathcal{E}_i(\cdot)}_{\boldsymbol{Q}^{(\text{out})}} \tag{32}$$

The soundness definition contains an additional simulator for simulating the states on $\boldsymbol{w}$ compared to Definition 5.2.

Definition 5.4. We say $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ under Set-up 9 is (δ, ϵ)-sound if:

For any efficient quantum adversary Adv, there exist efficient quantum operations $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for any state $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$:

If

$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi_{\text{test}}^{\mathsf{Adv}}(\rho_{\text{test}} \otimes \rho_0))) \leq 1 - \delta$$

then

$$\Pi_{\mathsf{pass}}(\pi_{\text{comp}}^{\mathsf{Adv}}(\Phi\otimes\rho_0)) \approx_{\epsilon}^{ind} \underbrace{\Pi_{\mathsf{pass}}}_{\text{on } \boldsymbol{flag}} (\underbrace{\mathsf{Sim}_1}_{\text{on } \boldsymbol{S},\boldsymbol{Q}^{(\text{in})},\boldsymbol{Q}^{(\text{out})},\boldsymbol{flag}} ((\underbrace{\mathbb{I}}_{\boldsymbol{P}} \otimes \underbrace{\mathcal{E}_{tar}}_{\text{as (32)}} \otimes \underbrace{\mathbb{I}}_{\boldsymbol{E}})(\Phi\otimes \underbrace{\mathsf{Sim}_0}_{\text{on } \boldsymbol{w},\boldsymbol{S}} (\underbrace{\rho_0}_{\boldsymbol{S},\boldsymbol{E}})))) \tag{33}$$

5.3 Potential Applications of ROAV

5.3.1 Building RSPV from ROAV

In this subsection we give a protocol for building RSPV protocols from ROAV and RSPV that are possibly more basic. The intuition is discussed in the beginning of Section 5 where ρ there corresponds to ρ_{comp} below.

Suppose $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ is an ROAV under Set-up 8 for target operator $\mathcal{E}$. Suppose the client also holds a classical register $\boldsymbol{D}^{(\text{comp})}$ and consider a state $\rho_{\text{comp}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{comp})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{in})}})$. Suppose $\pi_0(\text{mode}), \text{mode} \in \{\text{test}, \text{comp}\}$ is an RSPV (where the client could choose the states, as described in Set-up 3, Section 3.3.1) for the following honest behavior:

- If mode = test, prepare the state ρ_{test} on registers $\boldsymbol{D}^{(\text{in})}, \boldsymbol{Q}^{(\text{in})}$.
- If mode = comp, prepare the state ρ_{comp} on registers $\boldsymbol{D}^{(\text{comp})}, \boldsymbol{Q}^{(\text{in})}$.

Suppose π_0 is ϵ_0-sound; suppose $(\rho_{\text{test}}, \pi_{\text{test}}, \pi_{\text{comp}})$ is (δ, ϵ_1)-sound.

Protocol 6. $(\pi'_{\text{test}}, \pi'_{\text{comp}})$ below achieves a preRSPV for target state $\mathcal{E}(\rho_{\text{comp}})$.

Output registers: client-side classical registers $\boldsymbol{D}^{(\text{comp})}$, $\boldsymbol{D}^{(\text{out})}$, $\boldsymbol{flag}$; server-side quantum register $\boldsymbol{Q}^{(\text{out})}$.

π'_{test} is defined as follows:

1. The client executes $\pi_0(\text{test})$ with the server. Store the outputs in $\boldsymbol{flag}^{(1)}, \boldsymbol{D}^{(\text{in})}, \boldsymbol{Q}^{(\text{in})}$.
2. The client executes π_{test} with the server. Store the outputs in $\boldsymbol{flag}^{(2)}$.
3. The client sets $\boldsymbol{flag}$ to be fail if any one of $\boldsymbol{flag}^{(1)}, \boldsymbol{flag}^{(2)}$ is fail; otherwise it sets $\boldsymbol{flag}$ to be pass.

π'_{comp} is defined as follows:

1. The client executes $\pi_0(\text{comp})$ with the server. Store the outputs in $\boldsymbol{flag}^{(1)}, \boldsymbol{D}^{(\text{comp})}, \boldsymbol{Q}^{(\text{in})}$.
2. The client executes π_{comp} with the server. Store the outputs in $\boldsymbol{flag}^{(2)}, \boldsymbol{D}^{(\text{out})}, \boldsymbol{Q}^{(\text{out})}$.
3. The client sets $\boldsymbol{flag}$ to be fail if any one of $\boldsymbol{flag}^{(1)}, \boldsymbol{flag}^{(2)}$ is fail; otherwise it sets $\boldsymbol{flag}$ to be pass.

The completeness is from the protocol description: it passes in the test mode and prepares $\mathcal{E}(\rho_{\text{comp}})$ in the comp mode. The efficiency is also trivial. Below we state the soundness.

Theorem 5.1. *Protocol 6 is a preRSPV for target state $\mathcal{E}(\rho_{comp})$ that is $(\epsilon_0 + \delta, \epsilon_0 + \epsilon_1)$-sound.*

Compare the ROAV soundness with what we want, a missing step is how to relate the Φ in the ROAV soundness with ρ_{comp}. This step is by the following fact.

Fact 6. *For Φ defined in Set-up 8, there exists a POVM such that measuring $\boldsymbol{P}$ and storing results in $\boldsymbol{D}^{(comp)}$ transforms Φ to ρ_{comp}.*

Proof for Theorem 5.1. Suppose the adversary is Adv, by the soundness of π_0 (Definition 3.8) there exists an efficient quantum operation Sim_1 such that for any $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$,

$$\Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\pi_0^{\mathsf{Adv}_1}(\text{test})(\rho_0)) \approx^{ind}_{\epsilon_0} \Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\underbrace{\mathsf{Sim}_1}_{\text{on } \boldsymbol{Q}^{(\text{in})}, \boldsymbol{S}, \boldsymbol{flag}^{(1)}}(\rho_{\text{test}} \otimes \rho_0)) \tag{34}$$

$$\Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\pi_0^{\mathsf{Adv}_1}(\text{comp})(\rho_0)) \approx^{ind}_{\epsilon_0} \Pi^{\boldsymbol{flag}^{(1)}}_{\mathsf{pass}}(\mathsf{Sim}_1(\rho_{\text{comp}} \otimes \rho_0)) \tag{35}$$

To prove the soundness of Protocol 6, let's assume Adv could make π'_{test} passes with probability $\geq 1 - (\epsilon_0 + \delta)$. This combined with (34) implies that if the protocol π_{test} is executed on initial states $\rho_{\text{test}} \otimes \rho_0$ against adversary $\mathsf{Adv}_2 \circ \mathsf{Sim}_1$, the passing probability is $\geq 1 - \delta$. Then by the soundness of ROAV we have that, there exists an efficient quantum operation Sim_2 such that

$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}_2}_{\mathsf{comp}}(\mathsf{Sim}_1(\Phi \otimes \rho_0))) \approx^{ind}_{\epsilon_1} \Pi_{\mathsf{pass}}(\underbrace{\mathsf{Sim}_2}_{\text{on } \boldsymbol{S}, \boldsymbol{Q}^{(\text{in})}, \boldsymbol{Q}^{(\text{out})}, \boldsymbol{flag}^{(2)}}(\underbrace{\mathcal{E}}_{\boldsymbol{Q}^{(\text{in})} \to \boldsymbol{Q}^{(\text{out})}}(\underbrace{\Phi}_{\boldsymbol{P}, \boldsymbol{Q}^{(\text{in})}}) \otimes \rho_0))) \tag{36}$$

Note that the register $\boldsymbol{P}$ is in the environment, which is accessible by the distinguisher but not by any operators explicitly appeared in (36). Thus we could first apply Fact 6 and make the distinguisher

measure the $\boldsymbol{P}$ register of Φ to collapse Φ to ρ_{comp}, and then assume the collapsing happen in the beginning. Thus we get

$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}_2}_{\mathsf{comp}}(\mathsf{Sim}_1(\rho_{\text{comp}} \otimes \rho_0))) \approx^{ind}_{\epsilon_1} \Pi_{\mathsf{pass}}(\mathsf{Sim}_2(\mathcal{E}(\rho_{\text{comp}}) \otimes \rho_0)) \tag{37}$$

Combining (37) and (35) completes the proof. □

5.3.2 Testing ground state energy by ROAV

In this subsection we give a Hamiltonian ground energy testing protocol based on specific ROAV and RSPV. We first give the set-ups for the ROAV in our protocol. The set-up is based on the template set-up for ROAV with the server-side states (Set-up 9).

Set-up 10. Parameters:

- problem size parameter 1^n which describes the size of the witness of the Hamiltonian;
- 1^K which describes the number of repetition;
- approximation error parameter $1^{1/\epsilon}$;
- security parameter 1^κ.

Registers:

- client-side classical registers $\boldsymbol{D}^{(\text{in})}, \boldsymbol{D}^{(\text{comp})}, \boldsymbol{D}^{(\text{out})}$; $\boldsymbol{D}^{(\text{out})}$ holds $2nK$ classical bits. $\boldsymbol{D}^{(\text{in})}, \boldsymbol{D}^{(\text{comp})}$ are defined to be compatible with the states below.
- server-side quantum register $\boldsymbol{Q}^{(\text{in})}$, $\boldsymbol{w}$; each of both holds nK qubits.
- a register $\boldsymbol{P}$ in the environment; $\dim(\boldsymbol{P}) = \dim(\boldsymbol{Q}^{(\text{in})})$.

For $k \in [K]$, denote the k-th block of $\boldsymbol{D}^{(\text{out})}$ as $\boldsymbol{D}^{(\text{out})}_k$ which holds $2n$ bits and $\boldsymbol{D}^{(\text{comp})}_k, \boldsymbol{Q}^{(\text{in})}_k$, $\boldsymbol{P}_k$ are defined similarly.

Define the following state and operations:

$\rho_{\text{test}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{in})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{in})}})$. $|\Phi\rangle$ is the EPR entanglement between $\boldsymbol{P}$ and $\boldsymbol{Q}^{(\text{in})}$.

Define $\mathcal{E}_{Bells}$ to be the operation that measures $\boldsymbol{Q}^{(\text{in})}, \boldsymbol{w}$ in the Bell basis and stores the results in $\boldsymbol{D}^{(\text{out})}$; thus when the state in $\boldsymbol{P}, \boldsymbol{Q}^{(\text{in})}$ is $|\Phi\rangle$, the application of $\mathcal{E}_{Bell}$ teleports the state in $\boldsymbol{w}$ to $\boldsymbol{P}$.

Corresponding to an XZ local Hamimltonian $H = \sum_{j\in[m]} \gamma_j H_j$, 1^K, define $\rho_{\text{comp}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{D}^{(\text{comp})}} \otimes \mathcal{H}_{\boldsymbol{Q}^{(\text{in})}})$ as the outcome of the following operations applied on $|\Phi\rangle$:

1. For each $k \in [K]$:

   The client randomly chooses $j^{(k)} \in [m]$ and $\boldsymbol{P}_k$ is measured as follows:

   - If the observable on the t-th qubit of $H_{j^{(k)}}$ is σ_Z, measure the t-th qubit of $\boldsymbol{P}_k$ on the computational basis. The measurement outcome is represented by value in $\{0, 1\}$.
   - If the observable on the t-th qubit of $H_{j^{(k)}}$ is σ_X, measure the t-th qubit of $\boldsymbol{P}_k$ on the Hadamard basis. The measurement outcome is represented by value in $\{0, 1\}$.

Store $j^{(k)}$ together with the measurement results on $\boldsymbol{D}_k^{(\mathrm{comp})}$. Suppose the index $j^{(k)}$ is stored in register $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{index})}$ and the measurement results are stored in register $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})}$

Thus the application of $\mathcal{E}_{Bells}$ on ρ_{comp} and $w \in \mathrm{D}(\mathcal{H}_{\boldsymbol{w}})$ could be understood as follows: $w \in \mathrm{D}(\mathcal{H}_{\boldsymbol{w}})$ is teleported to $\boldsymbol{P}$, and then the client samples $j^{(k)}$ for each $k \in [K]$ and measures $H_{j^{(k)}}$ and stores the outcome in $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})}$. $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})}$ records the outcome in each round of Hamiltonian ground state testing, under the one-time-pad-encryption under corresponding keys in $\boldsymbol{D}_k^{(\mathrm{out})}$.

Then we introduce notations for calculating the estimated energy in such a procedure:

Notation 5.1. Then define $\mathrm{val}^H(\boldsymbol{D}^{(\mathrm{comp})}, \boldsymbol{D}^{(\mathrm{out})})$ as follows:

$$\mathrm{val}^H(\boldsymbol{D}^{(\mathrm{comp})}, \boldsymbol{D}^{(\mathrm{out})}) = \frac{1}{K} \sum_{k \in [K]} \mathrm{val}^H(\boldsymbol{D}_k^{(\mathrm{comp})}, \boldsymbol{D}_k^{(\mathrm{out})})$$

$$\mathrm{val}^H(\boldsymbol{D}_k^{(\mathrm{comp})}, \boldsymbol{D}_k^{(\mathrm{out})}) = \gamma_{j^{(k)}} \cdot (-1)^{\mathrm{Parity}(\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})} \oplus \mathrm{decodekey}^{H_{j^{(k)}}}(\boldsymbol{D}_k^{(\mathrm{out})}))}$$

where $j^{(k)}$ is the value of $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{index})}$, and $\mathrm{decodekey}^{H_{j^{(k)}}}(\boldsymbol{D}_k^{(\mathrm{out})})$ is defined as follows:

- If the observable on the t-th qubit of $H_{j^{(k)}}$ is σ_Z, the decode key for the corresponding bit in $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})}$ is the $(2t-1)$-th bit of $\boldsymbol{D}_k^{(\mathrm{out})}$.
- If the observable on the t-th qubit of $H_{j^{(k)}}$ is σ_X, the decode key for the corresponding bit in $\boldsymbol{D}_k^{(\mathrm{comp})(\mathrm{mr})}$ is the $2t$-th bit of $\boldsymbol{D}_k^{(\mathrm{out})}$.

Thus the $\mathrm{val}^H(\boldsymbol{D}^{(\mathrm{comp})}, \boldsymbol{D}^{(\mathrm{out})})$ is the estimation of the Hamiltonian ground state energy by taking the weighted average of K sampling.

Under Set-up 10, suppose $(\rho_{\mathsf{test}}, \pi_{\mathsf{test}}, \pi_{\mathsf{comp}})$ is an ROAV for $\mathcal{E}_{Bells}$ that is (δ, ϵ)-sound. Suppose $\pi_0(\mathrm{mode}, H, 1^K)$ as an RSPV (where the client could choose the states, as described in Set-up 3, Section 3.3.1) for the following honest behavior:

- If mode = test, prepare ρ_{test}.
- If mode = comp, prepare ρ_{comp} as defined in Set-up 10.

and π_0 is ϵ_0-sound.

Protocol 7. Input: an XZ 5-local Hamiltonian $H = \sum_{j \in [m]} \gamma_j H_j$, a, b, $b - a \geq 1/\mathsf{poly}(n)$, as Definition 2.2. Witness size parameter 1^n. Security parameter 1^κ.

Take $K = 100\kappa^2 \frac{1}{(b-a)^2}$.

1. The client samples mode $\in \{\mathsf{operatortest}, \mathsf{energytest}\}$ with probability $(\frac{1}{2}, \frac{1}{2})$ randomly. Depending on the value of roundtype:
   - If mode = operatortest:
     (a) The client executes $\pi_0(\mathrm{test}, 1^n, 1^K)$.
     (b) Then the client executes π_{test} with the server.
     (c) Reject if any step fails and accept otherwise.

- If mode = $\mathsf{energytest}$:
  (a) The client executes $\pi_0(\text{comp}, H, 1^K)$.
  (b) Then the client executes π_{comp} with the server.
  (c) Reject if any step fails or $\text{val}^H(\boldsymbol{D}^{(\text{comp})}, \boldsymbol{D}^{(\text{out})}) \geq \frac{a+b}{2}$ and accept otherwise.

The completeness is by an application of the Chernoff bound and the efficiency is trivial. Below we state and prove the soundness.

Theorem 5.2. *When H has ground state energy $\geq b$, Protocol 7 accepts with probability at most $\mathsf{negl}(\kappa) + \max\{1 - \frac{1}{2}(\delta - \epsilon_0), \frac{1}{2} + \frac{1}{2}(\epsilon + \epsilon_0)\}$.*

Proof. Suppose H has ground state energy $\geq b$, the adversary is Adv, [15] and the protocol accepts with probability[16] $\max\{1-\delta+\epsilon_0, \epsilon+\epsilon_0+\frac{1}{2}\}$. This implies that in the $\mathsf{operatortest}$ mode the protocol passes with probability $\geq 1 - \delta + \epsilon_0$. By the soundness of π_0 there exists a simulator Sim_1 that simulates the state after the first step with approximation error ϵ_0, which implies that π_{test} running from state ρ_{test} passes with probability $\geq \delta$. By the soundness of ROAV we have that, there exists an efficent simulator Sim_2, a quantum state $w \in \mathrm{D}(\mathcal{H}_{\boldsymbol{w}})$ such that

$$\Pi_{\mathsf{pass}}(\pi_{\mathsf{comp}}^{\mathsf{Adv}_2 \circ \mathsf{Sim}_1}(\Phi)) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\mathsf{Sim}((\underbrace{\mathbb{I}}_{\boldsymbol{P}} \otimes \mathcal{E}_{Bells})(\Phi \otimes w))) \tag{38}$$

Now we consider a distinguisher that measures the register $\boldsymbol{P}$ as described in Set-up 10 to collapse it to ρ_{comp}. This implies

$$\Pi_{\mathsf{pass}}(\pi_{\mathsf{comp}}^{\mathsf{Adv}_2 \circ \mathsf{Sim}_1}(\rho_{\text{comp}})) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\mathsf{Sim}((\underbrace{\mathbb{I}}_{\boldsymbol{D}^{(\text{comp})}} \otimes \mathcal{E}_{Bells})(\rho_{\text{comp}} \otimes w))) \tag{39}$$

The passing probability of the right hand side could be bounded directly by Chernoff bound, which is negligible when the ground state energy is $\geq b$. Then (39), the approximation error of π_0 together with the fact that the $\mathsf{energytest}$ mode happens with probability $\frac{1}{2}$ imply that the passing probability is at most $\frac{1}{2} + \frac{1}{2}(\epsilon + \epsilon_0)$. □

6 Concrete Constructions of RSPV Protocols

6.1 Overview

Let's recall the reduction diagram (Figure 2). In the following sections we will complete these reductions step by step; let's first give an overview that elaborates the intuition of each step.

6.1.1 From BB84 to OneBlock and OneBlockTensor

In Protocol 8 we build an RSPV for state family

$$\{\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle : x_0, x_1 \in \{0,1\}^m, \text{HW}(x_0 \oplus x_1) = 1, \text{Parity}(x_0) = 0)\}$$

[15] Note that we omit the initial state which is previously described by $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$. This is because since we only need to bound the passing probability of the overall protocol instead of proving a simulation-based soundness, we could simply assume the initial state is prepared by Adv.

[16] Below we omit the $\mathsf{negl}(\kappa)$ part in the soundness error during the proof.

from the RSPV for BB84 states; this protocol is called OneBlock. The intuition is as described in Section 1.3.3: notice that states in this family could be written as sequence of BB84 states where there is only one $|+\rangle$ state and no $|-\rangle$ state; thus the client only needs to do a repeat-and-pick on RSPV-for-BB84 protocol.

In Protocol 9 we build the RSPV for

$$\{\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle) : x_0, x_1 \in \{0,1\}^m, \mathrm{HW}(x_0 \oplus x_1) = 1, \mathrm{Parity}(x_0) = 0\}^{\otimes n} \tag{40}$$

which is the tensor products of the state family in OneBlock; this protocol is called OneBlockTensor. This is by taking the sequential repetition (as discussed in Section 3.4.1) of the protocol OneBlock.

6.1.2 Construction of MultiBlock

In Section 6.3 we go from OneBlockTensor to construct an RSPV for the following state family:

$$\{\frac{1}{\sqrt{2}}(|x_0^{(1)}||x_0^{(2)}||\cdots||x_0^{(n)}\rangle + |x_1^{(1)}||x_1^{(2)}||\cdots||x_1^{(n)}\rangle) :$$

$$\forall i \in [n], x_0^{(i)}, x_1^{(i)} \in \{0,1\}^m, \mathrm{HW}(x_0^{(i)} \oplus x_1^{(i)}) = 1; \mathrm{Parity}(x_0^{(1)}) = 0\} \tag{41}$$

Roughly speaking, the client forces the server to measure the state in (40) to get (41). To see this clearly, we could first consider two blocks as an example. Suppose the server initially holds the state

$$(|x_0^{(1)}\rangle + |x_1^{(1)}\rangle) \otimes (|x_0^{(2)}\rangle + |x_1^{(2)}\rangle), \quad \forall b, i, \mathrm{Parity}(x_b^{(i)}) = b$$

The client asks the server to measure the total parity of the strings it holds. Then if the server performs the measurement honestly, the state will collapse to:

$$\text{outcome} = 0 : |x_0^{(1)}||x_0^{(2)}\rangle + |x_1^{(1)}||x_1^{(2)}\rangle \tag{42}$$

$$\text{outcome} = 1 : |x_0^{(1)}||x_1^{(2)}\rangle + |x_1^{(1)}||x_0^{(2)}\rangle \tag{43}$$

The client could update the keys (that is, the state description) using the reported outcome and the original keys. If the client does the same for each $i = 2, 3 \cdots n$, in the honest setting the state in (40) will collapse to (41).

So what if the server cheats? One possible attack is that the server may not do the measurements and report the total parity honestly. To detect this attack, we will first construct a preRSPV (MultiBlockTest, MultiBlockComp) and use MultiBlockTest to test the server's behavior. In MultiBlockTest after getting the total parities the client will ask the server to measure all the states on the computational basis and report the measurement results; the client could check the results with its keys and see whether the results is consistent with the original keys and the total parities. As a concrete example, in (42) if the client asks the server to measure all the states, the measurement results should be either $x_0^{(1)}||x_0^{(2)}$ or $x_1^{(1)}||x_1^{(2)}$, otherwise the server is caught cheating.

After we get a preRSPV, we could make use of the amplification in Section 3.5.1 to get an RSPV for (41).

Let's briefly discuss how the security proof goes through. One desirable property of the security proof of this step is that, we only need to analyze a "information-theoretic core": if we assume the initial state is (40), the analysis of the preRSPV will be purely information-theoretic, which means,

the soundness holds against unbounded provers and we do not need to work on computational notions in the security analysis. After we prove the soundness of this information-theoretic part, we could prove the soundness of the overall protocol by calling the abstract properties in Section 3.4.2, 3.5.1.

6.1.3 Construction of KP

In Section 6.4 we go from MultiBlock to construct an RSPV for state family

$$\{\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle) : x_0, x_1 \in \{0,1\}^n\} \tag{44}$$

This is by first calling MultiBlock to prepare a sufficiently big state in the form of (41), and then letting the client reveals suitable information to allow the server to transform (41) to (44). Let's explain the intuition.

We first note that by calculating the parity of the first block, (41) could be transformed to

$$\frac{1}{\sqrt{2}}(|0\rangle|x_0^{(1)}||x_0^{(2)}||\cdots\rangle + |1\rangle|x_1^{(1)}||x_1^{(2)}||\cdots\rangle) : \forall i, x_0^{(i)}, x_1^{(i)} \in \{0,1\}^m, \mathrm{HW}(x_0^{(i)} \oplus x_1^{(i)}) = 1 \tag{45}$$

The difference to (44) is that these keys are not sampled uniformly randomly. Note that in (45) we omit some conditions on the keys and focus on the most significant one.

Then the client will reveal lots of information about these keys, which allows the server to transform each two blocks in (45) to a pair of uniform random bits. Let's use the first two blocks as an example. The client will reveal $x_0^{(1)}$ and $x_1^{(2)}$. Then by doing xor between them and the corresponding blocks, (45) becomes:

$$\frac{1}{\sqrt{2}}(|0\rangle|0^m||000\cdots1000\cdots||\cdots\rangle + |1\rangle|000\cdots1000\cdots||0^m||\cdots\rangle) \tag{46}$$

Now the $000\cdots1000\cdots$ could be seen as a unary encoding of a random number in $[m]$. By choosing m to be power of 2, converting unary encoding to binary encoding and trimming out extra zeros, this state becomes

$$\frac{1}{\sqrt{2}}(|0\rangle|\gamma_0^{(1)}||\cdots\rangle + |1\rangle|\gamma_1^{(1)}||\cdots\rangle) \tag{47}$$

where $\gamma_0, \gamma_1 \in \{0,1\}^{[\log m]}$ and are uniformly random. Doing this for each two blocks in (45) gives (44).

6.1.4 Construction of QFac (RSPV for $|+_\theta\rangle$)

Now we have an RSPV for states $\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle)$ with uniformly random x_0, x_1; we are going to construct an RSPV for the state[17]

$$|+_\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\mathrm{i}\pi\theta/4}|1\rangle), \theta \in \{0,1,2\cdots7\}.$$

We first note that existing work [29] also takes a similar approach: the client first instruct the server to prepare the state $|0\rangle|x_0\rangle + |1\rangle|x_1\rangle$, and then transforms it to $|+_\theta\rangle$.

[17]The protocol name QFac is from [25].

Let's explain the constructions. The overall ideas for the construction are basically from [29] (adapted to the languages of our framework). We will first construct a preRSPV with the score, as follows: in both the test mode and comp mode the client will instruct the server to prepare $|+_\theta\rangle$ state, then in the test mode the client will instruct the server to measure $|+_\theta\rangle$, and stores a score based on the reported result; in the comp mode $|+_\theta\rangle$ will be kept. Then once we show this protocol is indeed a preRSPV, we could amplify it to an RSPV as in Section 3.5.2.

The first step is to allow the honest server to transform $\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle)$ to $|+_\theta\rangle$. There are multiple ways to do it, for example:

1. The client will first instruct the server to introduce a phase of $e^{\mathrm{i}\pi\theta_{2,3}/4}, \theta_{2,3} \in \{0,1,2,3\}$ where $\theta_{2,3}$ are hidden in the server's view. This could be done by selecting the xor of the first two bits of x_0, x_1 as θ_2, θ_3. Then on the one hand $\theta_{2,3} = 2\theta_2 + \theta_3$ will be completely hidden; on the other hand using the phase-table-like technique in [58, 57] the honest server could introduce a phase of $e^{\mathrm{i}\pi\theta_{2,3}/4}$ to the x_1 branch up to a global phase.

2. The server does a Hadamard measurement on each bit of the x-part and get a measurement result d; this introduces a phase of $e^{\mathrm{i}\pi(d\cdot(x_0+x_1))}$ to the qubits. Then the server sends back d to the client and the client could calculate $\theta_1 = (d\cdot(x_0+x_1)) \mod 2$. The server holds a single qubit in the state $|+_\theta\rangle, \theta = 4\theta_1 + 2\theta_2 + \theta_3$.

Now in the comp mode we are done. In the test mode the client will continue to ask the server to make measurement on a random basis $|+_\varphi\rangle, |+_{\varphi+4}\rangle, \varphi \leftarrow_r \{0,1,2\cdots 7\}$. First we could see that when $\theta = \varphi$ the measurement will collapse to $|+_\varphi\rangle$ with probability 1, and when $\theta = \varphi + 4$ the measurement will collapse to $|+_{\varphi+4}\rangle$ with probability 1. Thus the client could record a "win" score if he has seen such an outcome. But solely doing this does not give us a full control on the server's behavior and states; an important idea is that, when φ is close to θ (or $\theta + 4$), the measurement should also collapse to $|+_\varphi\rangle$ (or $|+_{\varphi+4}\rangle$, correspondingly). This gives some probability of losing even for an honest server; however by analyzing the game it's possible to say "if the server wins with probability close to the optimal winning probability, the state before the testing measurement has to be close to the target state up to an isometry", which is still sufficient for amplification.

Security proofs, existing works and their limitations, and our approach So how could the quoted claim just now be proved? Existing works like [29, 25] has already done a lot of works on this part. In [25] the authors introduce a notion called *blind self-testing*. In more detail, let's denote the server-side state by the time that the comp mode is done corresponding to the client-side phase θ as ρ_θ. (In other words, the overall state is $\sum_\theta |\theta\rangle\langle\theta| \otimes \rho_\theta$.) Then the blind self-testing requires that the state $\rho_{\theta_{2,3}} + \rho_{\theta_{2,3}+4}$ is the same for any $\theta_{2,3}$, which is called (information-theoretic) *basis-blindness* (here $\theta_{2,3}$ is the "basis" and the notion means that the basis is completely hidden after randomization of θ_1). [25] shows that if the initial state satisfies the basis blindness property, the claim "high winning probability $\Rightarrow$ close to the target state up to an isometry" holds.

However, the proof of this claim given in [25] does not generalize to the computational analog of basis blindness. [25] does not solve the problem and leave the security of the whole protocol as a conjecture. [29] makes use of computational indistinguishability between states in the form of $\rho_{\theta_{2,3}} + \rho_{\theta_{2,3}+4}$ together with some quantum information theoretic arguments to prove the claim; in their proofs computational indistinguishability and quantum information theoretic arguments are mixed together, which could be complicated to work on and lead to sophisticated details [1].

Our new hammer for getting rid of this problem is the KP protocol, which is an RSPV for $|0\rangle|x_0\rangle + |1\rangle|x_1\rangle$. (Note that in previous works the preparation of $|0\rangle|x_0\rangle + |1\rangle|x_1\rangle$ is not known to

satisfy the RSPV soundness in the malicious setting.) By starting from KP, we are able to prove the security in a much nicer way:

1. The "information-theoretic core": in the security proof we could first simply assume the server holds exactly the state $|0\rangle|x_0\rangle + |1\rangle|x_1\rangle$; then we could prove the ρ_θ generated from it has the (information-theoretic) basis blindness property; then the analysis of the testing on $|+_\theta\rangle$ is basically from existing results [29, 25].

2. Once we complete the analysis of this information-theoretic core, we could compile it using results in Section 3.4.2 to get the desired soundness for the overall protocol, and then compile the preRSPV to an RSPV by amplification procedure in Section 3.5.2.

6.1.5 A summary

In summary we have achieve each step of the reductions as shown in Figure 2. The construction of OneBlock is by a repeat-and-pick procedure, OneBlockTensor is by the sequential composition, KP is by revealing some information to allow the server to transform the state to some other forms. For constructions of MultiBlock and QFac, we first analyze an "information-theoretic core" where we only need to work on statistical closeness, and compile the IT-core (IT=information-theoretic) to a full protocol by calling existing soundness properties.

6.2 From BB84 to OneBlock and OneBlockTensor

We define protocol BB84 as follows.

Definition 6.1. $\mathsf{BB84}(1^{1/\epsilon}, 1^\kappa)$ is defined to be an RSPV protocol for $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ that is complete, efficient and ϵ-sound.

The protocols below will build on the BB84 protocol.

Protocol 8 (OneBlock)**.** This is the RSPV for state family $\{\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle : x_0, x_1 \in \{0,1\}^m, \mathrm{HW}(x_0 \oplus x_1) = 1, \mathrm{Parity}(x_0) = 0)\}$.

Parameters: problem size 1^m, approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ.

Output registers: client-side classical registers $\boldsymbol{K}^{(\mathrm{out})} = (\boldsymbol{x}_0^{(\mathrm{out})}, \boldsymbol{x}_1^{(\mathrm{out})})$, where each of both holds m bits; client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$; server-side quantum register $\boldsymbol{Q}^{(\mathrm{out})}$ which holds m qubits.

Take $L = 4(m + \kappa)$.

1. For $i \in [L]$:

   (a) Execute $\mathsf{BB84}(1^{L/\epsilon}, 1^\kappa)$. The client stores the outcome in $\{0, 1, +, -\}$ in register $\boldsymbol{D}^{(\mathrm{temp})(i)}$ and the server stores the outcome in $\boldsymbol{Q}^{(\mathrm{temp})(i)}$.

   The client sets $\boldsymbol{flag}$ to be fail if any round fails.

2. The client randomly samples indices $i^{(1)}, i^{(2)} \cdots i^{(m)} \in [L]^m$ such that there is no repetition, there is exactly one i among them such that $\boldsymbol{D}^{(\mathrm{temp})(i)}$ is "+", and there is no i among them such that $\boldsymbol{D}^{(\mathrm{temp})(i)}$ is "-". The client tells the server its choices and server could

store the states in $\boldsymbol{Q}^{(\mathrm{temp})(i^{(1)})}, \boldsymbol{Q}^{(\mathrm{temp})(i^{(2)})}, \cdots \boldsymbol{Q}^{(\mathrm{temp})(i^{(m)})}$ in $\boldsymbol{Q}^{(\mathrm{out})}$. Then the state in $\boldsymbol{Q}^{(\mathrm{out})}$ could be equivalently written as

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle : x_0, x_1 \in \{0,1\}^m, \mathrm{HW}(x_0 \oplus x_1) = 1, \mathrm{Parity}(x_0) = 0) \tag{48}$$

and the client calculates x_0, x_1 and stores them in $\boldsymbol{x}_0^{(\mathrm{out})}, \boldsymbol{x}_1^{(\mathrm{out})}$.

The completeness and efficiency are from the protocol description.

Proposition 6.1. *Protocol 8 is ϵ-sound.*

Proof. By sequential composability of RSPV (Section 3.4.1) there exists a simulator that simulates the outputs of the first step from the state $\rho_{tar,BB84}^{\otimes L}$ where $\rho_{tar,BB84}$ to denote the target state of BB84.

Then the client samples indices, calculates $\boldsymbol{D}_0^{(\mathrm{out})}, \boldsymbol{D}_1^{(\mathrm{out})}$ and disgards the other registers. We note that, if the client does these operations from $\rho_{tar,BB84}^{\otimes L}$, the remaining state could be perfectly simulated from (48) (up to different locations of client-side registers): the simulator prepares the state being disgarded and reverses the honest execution.

Combining the two simulators above completes the proof. □

Protocol 9 (OneBlockTensor)**.** The state family is

$$\{\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle) : x_0, x_1 \in \{0,1\}^m, \mathrm{HW}(x_0 \oplus x_1) = 1, \mathrm{Parity}(x_0) = 0\}^{\otimes n}$$

.

Parameters: problem size $1^m, 1^n$, approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ.

Output registers: client-side classical registers $\boldsymbol{K} = (\boldsymbol{K}^{(i)})_{i\in[n]}$, $\boldsymbol{K}^{(i)} = (\boldsymbol{x}_0^{(i)}, \boldsymbol{x}_1^{(i)})$, where each of both holds m classical bits; client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$; server-side quantum register $\boldsymbol{Q} = (\boldsymbol{Q}^{(i)})_{i\in[n]}$, where each of them holds m qubits.

1. For each $i \in [n]$:
   (a) Execute OneBlock$(1^m, 1^{n/\epsilon}, 1^\kappa)$. The client stores the outcome in $\boldsymbol{x}_0^{(i)}, \boldsymbol{x}_1^{(i)}$, and the server stores the outcome in $\boldsymbol{Q}^{(i)}$.

   The client sets $\boldsymbol{flag}$ to be fail if any round fails.

The completeness and efficiency are from the protocol description.

Proposition 6.2. *Protocol 9 is ϵ-sound.*

The proof is by the sequential composition of RSPV (Section 3.4.1).

6.3 Construction of MultiBlock

In this section we construct RSPV for states

$$\{\frac{1}{\sqrt{2}}(|x_0^{(1)}||x_0^{(2)}||\cdots||x_0^{(n)}\rangle + |x_1^{(1)}||x_1^{(2)}||\cdots||x_1^{(n)}\rangle) :$$

$$\forall i \in [n], x_0^{(i)}, x_1^{(i)} \in \{0,1\}^m, \mathrm{HW}(x_0^{(i)} \oplus x_1^{(i)}) = 1; \mathrm{Parity}(x_0^{(1)}) = 0\} \tag{49}$$

We will give the preRSPV protocol, analyze its information-theoretic core, and prove the soundness of the preRSPV and amplify it to an RSPV protocol.

6.3.1 PreRSPV protocol for (49)

Protocol 10. The state family is (49). Below we construct a preRSPV for it.

Parameters: problem size $1^m, 1^n$, approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ. It is required that $\epsilon > 11n/\sqrt{m}$.

Output registers: client-side classical registers $\boldsymbol{K}^{(\mathrm{out})} = (\boldsymbol{K}^{(\mathrm{out})(i)})_{i\in[n]}$, $\boldsymbol{K}^{(\mathrm{out})(i)} = (\boldsymbol{x}_0^{(\mathrm{out})(i)}, \boldsymbol{x}_1^{(\mathrm{out})(i)})$, where each of both holds m classical bits; client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$; server-side quantum register $\boldsymbol{Q} = (\boldsymbol{Q}^{(i)})_{i\in[n]}$, where each of them holds m qubits.

Take $\epsilon_0 = \epsilon - 10n/\sqrt{m}$.

MultiBlockTest is defined as:

1. Both parties run $\mathsf{OneBlockTensor}(1^m, 1^n, 1^{1/\epsilon_0}, 1^\kappa)$; store the client-size output in register $\boldsymbol{K}^{(\mathrm{temp})}$ and the server-side output states in registers $\boldsymbol{Q}$. The client sets $\boldsymbol{flag}$ to be fail if it fails.

2. For each $i \in [2, \cdots n]$, the server evaluates $\mathsf{xorparity}(1, i)$ defined as the xor of parities of $\boldsymbol{Q}^{(1)}$ and $\boldsymbol{Q}^{(i)}$. The server sends the value back to the client and the client stores it in $\boldsymbol{xor}^{(i)}$.

3. The client asks the server to measure all the $\boldsymbol{Q}$ register on the computational basis and reveal the results. The client checks:

   - For each $i \in [n]$ the server's measurement results for $\boldsymbol{Q}^{(i)}$ is in $\boldsymbol{K}^{(\mathrm{temp})(i)}$.
   - For each $i \in [2, \cdots n]$ the xor of parities of measurement results for $\boldsymbol{Q}^{(1)}$ and $\boldsymbol{Q}^{(i)}$ is the same as $\boldsymbol{xor}^{(i)}$.

   The client sets $\boldsymbol{flag}$ to be fail of check fails and pass otherwise.

MultiBlockComp is defined as:

1. The same as the first step of MultiBlockTest.

2. First do the same thing as the second step of $\mathsf{MultiBlockTest}$.
   The client calculates $\boldsymbol{K}^{(\mathrm{out})(i)}$ as follows: $\boldsymbol{K}^{(\mathrm{out})(1)} = \boldsymbol{K}^{(\mathrm{temp})(1)}$. For $i \geq 2$ $(\boldsymbol{x}_0^{(\mathrm{out})(i)}, \boldsymbol{x}_1^{(\mathrm{out})(i)}) = (\boldsymbol{x}_{\boldsymbol{xor}^{(i)}}^{(\mathrm{temp})(i)}, \boldsymbol{x}_{1\oplus\boldsymbol{xor}^{(i)}}^{(\mathrm{temp})(i)})$.

The completeness and efficiency are from the protocol description. To prove its soundness, we first analyze its second and third steps, which is its "information-theoretic core".

6.3.2 Analysis of the information-theoretic core

Let's formalize a set-up that describes the information-theoretic core of Protocol 10.

Set-up 11. Parameters: $1^m, 1^n$.

Consider the following registers as used in Protocol 10: client-side classical registers $\boldsymbol{K}^{(\mathrm{temp})}$ for holding the classical description of the states prepared by $\mathsf{OneBlockTensor}$. Server-side quantum register $\boldsymbol{Q}$ for holding the quantum states from $\mathsf{OneBlockTensor}$. Client-side classical register $\boldsymbol{xor} = (\boldsymbol{xor}^{(2)}, \boldsymbol{xor}^{(3)} \cdots \boldsymbol{xor}^{(n)})$ where each of them is a single bit.

Part of the initial state is $\rho_{OBT} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{K}^{(\mathrm{temp})}} \otimes \mathcal{H}_{\boldsymbol{Q}})$ where ρ_{OBT} is the target state of $\mathsf{OneBlockTensor}$. Use $\mathsf{sendxorparity}$ to denote the operation that measures $\mathsf{xorparity}(1, i)$ for each $i \in [2, n]$ and sends the results to register $\boldsymbol{xor}$. Then the target state of Protocol 10 is $\mathsf{sendxorparity}(\rho_{OBT})$ (up to a change of client side descriptions). For modeling the initial states in the malicious setting, consider server-side quantum register $\boldsymbol{S}$ and environment register $\boldsymbol{E}$. The initial states in the malicious setting that we consider could be described as $\rho_{OBT} \otimes \rho_0$ where $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$.

We use $(\mathsf{MultiBlockTest}_{\geq 2}, \mathsf{MultiBlockComp}_{\geq 2})$ to denote the protocols that starts from the second step of Protocol 10. Below we introduce notations for describing each step of protocol executions.

- Use $\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(\mathrm{temp})}}$ to denote the projection onto the space that the value of $\boldsymbol{Q}$ is within $\boldsymbol{K}^{(\mathrm{temp})}$. Use $\Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}$ to denote the projection onto the space that for each $i \in [2, \cdots n]$ the xor of parities of $\boldsymbol{Q}^{(1)}$ and $\boldsymbol{Q}^{(i)}$ is the same as the value in $\boldsymbol{xor}^{(i)}$. Thus the passing condition in the third step of $\mathsf{MultiBlockTest}$ corresponds to[18] $\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(\mathrm{temp})}} \Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}$.
- Suppose $\mathsf{Adv}_2, \mathsf{Adv}_3$ are two superoperators operated on $\boldsymbol{Q}, \boldsymbol{S}$, which describe the adversary's operations in the second and third steps; we could purify them to be unitaries on $\boldsymbol{Q}, \boldsymbol{S}$ and some reference registers. Use $\mathsf{send}_{\mathrm{to}\ \boldsymbol{xor}}$ to denote the message sending operation that sends the outcome of Adv_2 to $\boldsymbol{xor}$.

For purifying the density operators, consider reference registers $\boldsymbol{R}_{\boldsymbol{K}^{(\mathrm{temp})}}, \boldsymbol{R}_{\boldsymbol{xor}}$. Use $|\varphi_{OBT}\rangle$ to denote the purification of ρ_{OBT} where $\boldsymbol{K}^{(\mathrm{temp})}$ is purified by $\boldsymbol{R}_{\boldsymbol{K}^{(\mathrm{temp})}}$.

We will show that if the adversary could pass in the test mode, the output state in the comp mode could be statistically simulated from the target state (we could simply work on $\mathsf{sendxorparity}(\rho_{OBT})$, which is the same as the target state up to a client-side change of representation of descriptions). Below we define the simulator and the corresponding statement is Corollary 6.5. Corollary 6.5 is a corollary of Proposition 6.3, which relates the real execution with simulation using low-level

[18] In other words, if the adversary could pass the checking in the third step of $\mathsf{MultiBlockTest}$ with high probability, then consider the state before the adversary sends back the measurement results, this state is approximately in the space of $\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(\mathrm{temp})}} \Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}$.

descriptions. To prove Proposition 6.3, we first prove Lemma 6.4. These statements are stated and prove below.

Definition 6.2. Under Set-up 11, for any Adv_2, define Sim as follows:

1. Apply Adv_2.
2. Instead of doing $\mathsf{send}_{\text{to } \boldsymbol{xor}}$, the simulator simply copies[19] the response to a temporary register $\boldsymbol{simxor}$. Denote this step as $\mathsf{send}_{\text{to } \boldsymbol{simxor}}$.

   Then $\boldsymbol{simxor}$ is disgarded. Denote this step as $\mathsf{Disgard}_{\boldsymbol{simxor}}$.

Proposition 6.3. *Under Set-up 11, for each $\rho_0 \in D(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$, $\mathsf{Adv}_2, \mathsf{Adv}_3$, there is*

$$\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(temp)}} \Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}} \circ \mathsf{Adv}_3 \circ \mathsf{send}_{to\ \boldsymbol{xor}} \circ \mathsf{Adv}_2(\rho_{OBT} \otimes \rho_0) \tag{50}$$

$$\approx_{4n/\sqrt{m}} \Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(temp)}} \circ \mathsf{Disgard}_{\boldsymbol{simxor}} \circ \Pi_{\boldsymbol{xor}=\boldsymbol{simxor}} \circ \mathsf{Adv}_3 \circ \mathsf{send}_{to\ \boldsymbol{simxor}} \circ \mathsf{Adv}_2(\mathsf{sendxorparity}(\rho_{OBT}) \otimes \rho_0) \tag{51}$$

Here $\Pi_{\boldsymbol{xor}=\boldsymbol{simxor}}$ denotes the projection onto the space that the value of $\boldsymbol{xor}$ is the same as the value of $\boldsymbol{simxor}$.

The form of Proposition 6.3 strikes a balance between what is easy to prove and what we want. What we want is something like "if the test mode passes, the output state of the comp mode is the same as the output of Sim". Such a statement is given in Corollary 6.5 which will be proved as a corollary of Proposition 6.3. So what does Proposition 6.3 mean? We first note in (51) there are indeed the operators used in the construction of Sim. Then based on the operations of Sim, (51) continues to apply Adv_3 and $\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(\text{temp})}}$, and insert a projection $\Pi_{\boldsymbol{xor}=\boldsymbol{simxor}}$, which makes it a simulation of a subspace of the test mode. Proposition 6.3 does not rely on the condition that the test mode passes with high probability; in Corollary 6.5 we will combine this condition with Proposition 6.3 and remove these extra operators to get a simulation of the comp mode.

We will purify and decompose the state to prove Proposition 6.3. To do this let's first prove a lemma.

Lemma 6.4. *For any normalized pure state $|\varphi_0\rangle \in \mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}}$, any operator O that operates on $\boldsymbol{Q}, \boldsymbol{S}$ and could be written as unitaries and projections, there is*

$$\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(temp)}} O(|\varphi_{OBT}\rangle \otimes |\varphi_0\rangle) \approx_{2n/\sqrt{m}} \sum_{\vec{b} \in \{0,1\}^n} \Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(temp)}_{\vec{b}}} O \Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(temp)}_{\vec{b}}} (|\varphi_{OBT}\rangle \otimes |\varphi_0\rangle) \tag{52}$$

where $\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(temp)}_{\vec{b}}}$ denotes the projection onto the space that the value of $\boldsymbol{Q}$ is the same as $\boldsymbol{x}^{(temp)(1)}_{b^{(1)}} || \boldsymbol{x}^{(temp)(2)}_{b^{(2)}} \cdots \boldsymbol{x}^{(temp)(n)}_{b^{(n)}}$, where we use $b^{(1)} b^{(2)} \cdots b^{(n)}$ to denote the coordinates of $\vec{b}$.

A variant of this lemma where the states are computationally indistinguishable is used in [57]. Below we give the proof.

Proof of Lemma 6.4. We consider a sequence of states in the following form, for each $i \in [0, n]$:

$$\Pi^{\boldsymbol{Q}^{(>t)}}_{\in \boldsymbol{K}^{(\text{temp})(>t)}} \sum_{\vec{b} \in \{0,1\}^t} \Pi^{\boldsymbol{Q}^{(\leq t)}}_{\vec{\boldsymbol{x}}^{(\text{temp})(\leq t)}_{\vec{b}}} O \Pi^{\boldsymbol{Q}^{(\leq t)}}_{\vec{\boldsymbol{x}}^{(\text{temp})(\leq t)}_{\vec{b}}} (|\varphi_{OBT}\rangle \otimes |\varphi_0\rangle) \tag{53}$$

[19] We mean bit-wise CNOT.

where $\Pi^{\boldsymbol{Q}^{(\le t)}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})(\le t)}_{\vec{b}}}$ denotes the projection onto the space that the value of $\boldsymbol{Q}^{(\le t)}$ is the same as $\boldsymbol{x}^{(\mathrm{temp})(1)}_{b^{(1)}}||\boldsymbol{x}^{(\mathrm{temp})(2)}_{b^{(2)}}\cdots\boldsymbol{x}^{(\mathrm{temp})(t)}_{b^{(t)}}$, where we use $b^{(1)}b^{(2)}\cdots b^{(t)}$ to denote the coordinates of $\vec{b}$, and $\Pi^{\boldsymbol{Q}^{(>t)}}_{\in\boldsymbol{K}^{(\mathrm{temp})(>t)}}$ denotes the projection onto the space that the value of $\boldsymbol{Q}^{(>t)}$ is within $\boldsymbol{K}^{(\mathrm{temp})(>t)}$. Thus for $t=0$ we get the left hand side of (52) and for $t=n$ we get the right hand side of (52).

We only need to prove the difference of (53) has difference at most $\frac{2}{\sqrt{m}}$ for between $t-1,t$ for each $t\in[n]$. The subtraction of these two states gives:

$$\Pi^{\boldsymbol{Q}^{(>t)}}_{\in\boldsymbol{K}^{(\mathrm{temp})(>t)}}\sum_{\vec{b}\in\{0,1\}^{t-1}}\Pi^{\boldsymbol{Q}^{(t)}}_{=\boldsymbol{x}^{(\mathrm{temp})(t)}_1}\Pi^{\boldsymbol{Q}^{(<t)}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})(<t)}_{\vec{b}}}O\Pi^{\boldsymbol{Q}^{(<t)}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})(<t)}_{\vec{b}}}\Pi^{\boldsymbol{Q}^{(t)}}_{=\boldsymbol{x}^{(\mathrm{temp})(t)}_0}(|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)$$

$$+\Pi^{\boldsymbol{Q}^{(>t)}}_{\in\boldsymbol{K}^{(\mathrm{temp})(>t)}}\sum_{\vec{b}\in\{0,1\}^{t-1}}\Pi^{\boldsymbol{Q}^{(t)}}_{=\boldsymbol{x}^{(\mathrm{temp})(t)}_0}\Pi^{\boldsymbol{Q}^{(<t)}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})(<t)}_{\vec{b}}}O\Pi^{\boldsymbol{Q}^{(<t)}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})(<t)}_{\vec{b}}}\Pi^{\boldsymbol{Q}^{(t)}}_{=\boldsymbol{x}^{(\mathrm{temp})(t)}_1}(|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)\qquad(54)$$

each of them could be understood as guessing one of $\boldsymbol{x}^{(\mathrm{temp})(t)}$ given the other. The relation between $\boldsymbol{x}^{(\mathrm{temp})(t)}_0,\boldsymbol{x}^{(\mathrm{temp})(t)}_1$ is that they differ in one bit, thus there are m choices that are equally possible. Thus the norm of (54) could be upper bounded by $2/\sqrt{m}$, which completes the proof. □

Proof of Proposition 6.3. Consider the purification of Proposition 6.3. Then ρ_0 is replace by a pure state $|\varphi_0\rangle$ in $\mathcal{H}_{\boldsymbol{S}}\otimes\mathcal{H}_{\boldsymbol{E}}$, ρ_{OBT} is replaced by $|\varphi_{OBT}\rangle$, Advs are considered to be unitaries, send operators will also copy the information to the corrsponding reference register (corresponding to $\boldsymbol{xor}$, it's $\boldsymbol{R_{xor}}$). The partial trace $\mathsf{Disgard}_{\boldsymbol{simxor}}$ is handled as follows: the purification simply removes it from (51) and requires that the purification of both sides of (50)(51) could be transformed to each other by a local operaion on $\boldsymbol{R_{xor}}$ and $\boldsymbol{simxor}$ (in other words, we do not require the purified states to be close to each other; what we need is only the closeness when $\boldsymbol{R_{xor}}$ and $\boldsymbol{simxor}$ are all traced out).

Thus (50) becomes

$$\Pi^{\boldsymbol{Q}}_{\in\boldsymbol{K}^{(\mathrm{temp})}}\Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}\mathsf{Adv}_3\mathsf{send}_{\mathrm{to}\ \boldsymbol{xor}}\mathsf{Adv}_2(|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)\qquad(55)$$

and (51) becomes

$$\Pi^{\boldsymbol{Q}}_{\in\boldsymbol{K}^{(\mathrm{temp})}}\Pi_{\boldsymbol{xor}=\boldsymbol{simxor}}\mathsf{Adv}_3\mathsf{send}_{\mathrm{to}\ \boldsymbol{simxor}}\mathsf{Adv}_2(\mathsf{sendxorparity}\,|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)\qquad(56)$$

Applying Lemma 6.4 to (55)(56), we get that (55) is $(2n/\sqrt{m})$-close to

$$\sum_{\vec{b}\in\{0,1\}^n}\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})}_{\vec{b}}}\Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}\mathsf{Adv}_3\mathsf{send}_{\mathrm{to}\ \boldsymbol{xor}}\mathsf{Adv}_2\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})}_{\vec{b}}}(|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)\qquad(57)$$

and (56) is $(2n/\sqrt{m})$-close to

$$\sum_{\vec{b}\in\{0,1\}^n}\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})}_{\vec{b}}}\Pi_{\boldsymbol{xor}=\boldsymbol{simxor}}\mathsf{Adv}_3\mathsf{send}_{\mathrm{to}\ \boldsymbol{simxor}}\mathsf{Adv}_2(\mathsf{sendxorparity}\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})}_{\vec{b}}}\,|\varphi_{OBT}\rangle\otimes|\varphi_0\rangle)\qquad(58)$$

We could compare (57) and (58) directly. Both of them are a summation of 2^n different branches so we could compare each branch of them. On each branch given by $\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}^{(\mathrm{temp})}_{\vec{b}}}$, $\mathsf{xorparity}(\boldsymbol{Q})$ has a fixed

value; for simplicity let's denote it as α. In (57) send sends a value to $\boldsymbol{xor}$ (which is not necessarily α but contains a sub-branch with value α) and finally a projection onto α is performed (notice that we could assume (57) first do $\Pi^{\boldsymbol{Q}}_{\vec{\boldsymbol{x}}_b^{(\mathrm{temp})}}$ and then do $\Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}}$). In (58) $\mathsf{sendxorparity}$ writes α to $\boldsymbol{xor}$, and send sends a value to $\boldsymbol{simxor}$ (which is not necessarily α but contains a sub-branch with value α) and finally a projection $\Pi_{\boldsymbol{xor}=\boldsymbol{simxor}}$ is performed. We could see the only difference is the usage of $\boldsymbol{simxor}$; thus (57)(58) are the same up to a local operation on $\boldsymbol{simxor}, \boldsymbol{R}_{\boldsymbol{xor}}$.

This completes the proof. □

Corollary 6.5. *Under Set-up 11,* $(\mathsf{MultiBlockTest}_{\geq 2}, \mathsf{MultiBlockComp}_{\geq 2})$ *is* $(n/\sqrt{m}, 10n/\sqrt{m})$*-sound.*

Proof of Corollary 6.5. For any adversary Adv and $\rho_0 \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$, $\mathsf{MultiBlockTest}_{\geq 2}$ passes with probability $\geq 1 - n/\sqrt{m}$ translates to:

$$\mathrm{tr}(\Pi^{\boldsymbol{Q}}_{\in \boldsymbol{K}^{(\mathrm{temp})}} \Pi_{\mathsf{xorparity}(\boldsymbol{Q})=\boldsymbol{xor}} \circ \mathsf{Adv}_3 \circ \mathsf{send}_{\mathrm{to}\ \boldsymbol{xor}} \circ \mathsf{Adv}_2(\rho_{OBT} \otimes \rho_0)) \geq 1 - n/\sqrt{m} \tag{59}$$

This is as given in (50), which by Proposition 6.3 is $(4n/\sqrt{m})$-close to (51). Both (50)(51) has a form of projecting a trace-1 operator to a subspace; this fact, (50)(51) and (59) imply that the states before the projection are also close to each other:[20]

$$\mathsf{Adv}_3 \circ \mathsf{send}_{\mathrm{to}\ \boldsymbol{xor}} \circ \mathsf{Adv}_2(\rho_{OBT} \otimes \rho_0) \approx_{10n/\sqrt{m}} \mathsf{Adv}_3 \circ \mathsf{Sim}(\mathsf{sendxorparity}(\rho_{OBT}) \otimes \rho_0) \tag{60}$$

where we encapsulate (51) using the construction of Sim in Definition 6.2.

Inverting Adv_3 completes the proof. □

6.3.3 Compilation and amplification to preRSPV and RSPV

We could first prove the soundness of Protocol 10 using Corollary 6.5 and the soundness of $\mathsf{OneBlockTensor}$. (see 3.4.2).

Theorem 6.6. *Protocol 10 is* $(\epsilon - 11n/\sqrt{m}, \epsilon)$*-sound.*

Proof. By the soundness of $\mathsf{OneBlockTensor}$ the output state of the first step of Protocol 10 could be simulated from ρ_{OBT} with approximation error $\epsilon_0 = \epsilon - 10n/\sqrt{m}$. This together with the condition that $\mathsf{MultiBlockTest}$ passes with probability $\geq 1 - (\epsilon - 11n/\sqrt{m})$ implies that $\mathsf{MultiBlockTest}_{\geq 2}$ passes with probability $\geq 1 - n/\sqrt{m}$ (against the corresponding adversary). By Corollary 6.5 $\mathsf{MultiBlockComp}_{\geq 2}$ is simulated by Sim with approximation error $10n/\sqrt{m}$. Combining Sim with the simulator for the first step of Protocol 10 gives a simulator for $\mathsf{MultiBlockComp}$ (up to a change of client-side representation of descriptions). This completes the proof. □

Then we could amplify Protocol 10 (preRSPV for (49)) to an RSPV protocol:

Protocol 11 ($\mathsf{MultiBlock}$)**.** The set-up is the same as Protocol 10.

The protocol is the amplification procedure (Protocol 3) running on Protocol 10.

[20] The details are as follows. We could first use (59) and (50)(51) to show the trace of (51) is $(5n/\sqrt{m})$-close to 1, which implies that (51) is $(5n/\sqrt{m})$-close to the state before the projection. Combining it with (59) and (50)(51) completes the proof.

The completeness, efficiency and soundness are from the properties of Protocol 10 and the preRSPV-to-RSPV amplification procedure (Section 3.5.1).

6.4 Construction of KP

To formalize the protocol, define function u2b : $\{0,1\}^m \to \{0,1\}^{\log_2(m)}$ as follows: given a string in the form of $000\cdots1000\cdots$, it outputs the binary representation for the locations of number 1.

Protocol 12 (KP)**.** This is the RSPV for state family $\{\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle+|1\rangle|x_1\rangle) : x_0, x_1 \in \{0,1\}^n\}$.

Parameters: problem size 1^n, approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ.

Output registers: client-side classical registers $\boldsymbol{K}^{(\mathrm{out})} = (\boldsymbol{x}_0^{(\mathrm{out})}, \boldsymbol{x}_1^{(\mathrm{out})})$, where each of both holds n bits; client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$; server-side quantum register $\boldsymbol{Q}^{(\mathrm{out})} = (\boldsymbol{Q}^{(\mathrm{out})(\mathrm{subs})}, \boldsymbol{Q}^{(\mathrm{out})(\mathrm{key})})$ which hold 1 qubit and n qubits correspondingly.

Take $n_0 = 2n$, m_0 to be the smallest power of 2 such that $m > (12n_0/\epsilon)^2$.

1. Execute $\mathsf{MultiBlock}(1^{m_0}, 1^{n_0}, 1^{1/\epsilon}, 1^\kappa)$. The client stores the outcome in $\boldsymbol{K}^{(\mathrm{temp})} = (\boldsymbol{K}^{(\mathrm{temp})(i)})_{i\in[n_0]}, \boldsymbol{K}^{(\mathrm{temp})(i)} = (\boldsymbol{x}_0^{(\mathrm{temp})(i)}, \boldsymbol{x}_1^{(\mathrm{temp})(i)})$ and the server stores the outcome in $\boldsymbol{Q}^{(\mathrm{temp})}$. In the honest setting the server holds

$$\{\frac{1}{\sqrt{2}}(|x_0^{(\mathrm{temp})(1)}||x_0^{(\mathrm{temp})(2)}||\cdots||x_0^{(\mathrm{temp})(2n)}\rangle + |x_1^{(\mathrm{temp})(1)}||x_1^{(\mathrm{temp})(2)}||\cdots||x_1^{(\mathrm{temp})(2n)}\rangle) :$$

$$\forall i \in [2n], x_0^{(\mathrm{temp})(i)}, x_1^{(\mathrm{temp})(i)} \in \{0,1\}^m, \mathrm{HW}(x_0^{(\mathrm{temp})(i)} \oplus x_1^{(\mathrm{temp})(i)}) = 1; \mathrm{Parity}(x_0^{(\mathrm{temp})(1)}) = 0\} \tag{61}$$

and the client holds all these keys.

The client sets $\boldsymbol{flag}$ to be fail if this step fails.

2. For each $i \in [n_0/2]$, the client sends the following information to the server:
   - $x_0^{(\mathrm{temp})(2i-1)}, x_1^{(\mathrm{temp})(2i)}$;
   - The bits of $\mathrm{u2b}(x_0^{(\mathrm{temp})(2i-1)} \oplus x_1^{(\mathrm{temp})(2i-1)})$ excluding the first bits; the bits of $\mathrm{u2b}(x_0^{(\mathrm{temp})(2i)} \oplus x_1^{(\mathrm{temp})(2i)})$ excluding the first bits. (The length of this part is $2(\log_2(m_0) - 1)$.)

   With these information, the server could do the following transformation on the state:

   It first calculates and stores the parity of $\boldsymbol{Q}^{(\mathrm{temp})(1)}$ in $\boldsymbol{Q}^{(\mathrm{out})(\mathrm{subs})}$.

   Then for each i:

   (a) It xors $x_0^{(\mathrm{temp})(2i-1)}, x_1^{(\mathrm{temp})(2i)}$ to each block to transform each block into the form of $000\cdots1000\cdots$;

   (b) Then it transforms unary representation to binary representation for the location of non-zero bits.

(c) Then it only keeps the first bits for each block and transform the remaining bits to 0 using the second part of the client-side messages.

Denote the first bit of $\mathrm{u2b}(x_0^{(\mathrm{temp})(2i)} \oplus x_1^{(\mathrm{temp})(2i)})$ as $b_0^{(out)(i)}$ and denote the first bit of $\mathrm{u2b}(x_0^{(\mathrm{temp})(2i-1)} \oplus x_1^{(\mathrm{temp})(2i-1)})$ as $b_1^{(out)(i)}$, the server-side state in the end is

$$\frac{1}{\sqrt{2}}\Big(\underbrace{|0\rangle}_{\boldsymbol{Q}^{(\mathrm{out})(\mathrm{subs})}} \underbrace{|b_0^{(out)(1)}||b_0^{(out)(2)}||\cdots b_0^{(out)(n)}\rangle}_{\boldsymbol{Q}^{(\mathrm{out})(\mathrm{key})}} + |1\rangle\,|b_1^{(out)(1)}||b_1^{(out)(2)}||\cdots b_1^{(out)(n)}\rangle\Big) \quad (62)$$

Note that for each i, $b_0^{(out)(i)}, b_1^{(out)(i)}$ are uniformly random bits from $\{0,1\}^2$. The client then calculates and stores $(b_0^{(out)(1)}||b_0^{(out)(2)}||\cdots b_0^{(out)(n)}, b_1^{(out)(1)}||b_1^{(out)(2)}||\cdots b_1^{(out)(n)})$ in register $(\boldsymbol{x}_0^{(\mathrm{out})}, \boldsymbol{x}_1^{(\mathrm{out})})$.

The completeness and efficiency are from the protocol description. Below we state and prove the soundness.

Theorem 6.7. *Protocol 12 is ϵ-sound.*

Proof. By the soundness of MultiBlock there exists a simulator that simulates the output of the first step from (61).

Then the client reveals lots of information to allow the server to transform the state to (62). Note that the information revealed by the client are all disgarded by the client; the client only calculates and keeps $\boldsymbol{x}_0^{(\mathrm{out})}, \boldsymbol{x}_1^{(\mathrm{out})}$. Then starting from (62), the simulator could simulate the disgarded information on its own and reverse the transformation to to simulate the joint state corresponding to (61) (up to a cnange of locations of client-side registers).

Combining the two simulators above completes the proof. □

6.5 Construction of QFac (RSPV for $|+_\theta\rangle$)

In this section we construct RSPV for $|+_\theta\rangle$ from $\frac{1}{\sqrt{2}}(|0\rangle\,|x_0\rangle + |1\rangle\,|x_1\rangle)$. We will give the preRSPV-with-score protocol for it, analyze its information-theoretic core, and prove the soundness of the preRSPV and amplify it to an RSPV protocol.

6.5.1 PreRSPV-with-score for $|+_\theta\rangle$

Protocol 13. Below we construct a preRSPV with the score for $|+_\theta\rangle, \theta \in \{0,1,2\cdots 7\}$.
Parameters: a temporary approximation error parameter $1^{1/\epsilon_0}$, security parameter 1^κ.
Output registers:

- client-side classical register $\boldsymbol{\theta}$ with value in $\{0,1,2\cdots 7\}$. We also say $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$ where each of them is a classical bit and $\boldsymbol{\theta} = 4\boldsymbol{\theta}_1 + 2\boldsymbol{\theta}_2 + \boldsymbol{\theta}_3$.
- Client-side classical register $\boldsymbol{flag}$ with value in $\{\mathsf{pass}, \mathsf{fail}\}$; client-side classical register $\boldsymbol{score}$ with value in $\{\mathsf{win}, \mathsf{lose}, \perp\}$.
- Server side quantum register $\boldsymbol{q}$ which holds a single qubit.

Take $n = \kappa$.
QFacTest is defined as:

1. Both parties run $\mathsf{KP}(1^n, 1^{1/\epsilon_0}, 1^\kappa)$; store the client-size output in register $\boldsymbol{K} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$ and the server-side output states in registers $(\boldsymbol{q}, \boldsymbol{Q})$:

$$\frac{1}{\sqrt{2}}(\underbrace{|0\rangle}_{\boldsymbol{q}}\underbrace{|x_0\rangle}_{\boldsymbol{Q}} + |1\rangle|x_1\rangle) \tag{63}$$

   The client sets $\boldsymbol{flag}$ to be fail if it fails.

2. In (63), denote the first two bits of x_0 as $b_0^{(1)}$ and $b_0^{(2)}$, denote the first two bits of x_1 as $b_1^{(1)}$ and $b_1^{(2)}$. The server could do control-phase operations to add the following phases to the $\boldsymbol{q}$ register:

$$\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle) \tag{64}$$

$$\to\frac{1}{\sqrt{2}}(e^{\mathrm{i}\pi(2b_0^{(1)}+b_0^{(2)})/4}|0\rangle|x_0\rangle + e^{\mathrm{i}\pi(2b_1^{(1)}+b_1^{(2)})/4}|1\rangle|x_1\rangle) \tag{65}$$

$$=(\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + e^{\mathrm{i}\pi(2(b_0^{(1)}\oplus b_1^{(1)})+(b_0^{(2)}\oplus b_1^{(2)}))/4}|1\rangle|x_1\rangle)) \cdot \text{global phase} \tag{66}$$

   The client stores $b_0^{(1)} \oplus b_1^{(1)}$ in register $\boldsymbol{\theta}^{(2)}$ and stores $b_0^{(2)} \oplus b_1^{(2)}$ in register $\boldsymbol{\theta}_3$.

   Then the server does Hadamard measurements for each bit in $\boldsymbol{Q}$; suppose the measurement result is d. This transforms (66) to

$$(\frac{1}{\sqrt{2}}(|0\rangle + e^{\mathrm{i}\pi(4d\cdot(x_0\oplus x_1)+2(b_0^{(1)}\oplus b_1^{(1)})+(b_0^{(2)}\oplus b_1^{(2)}))/4}|1\rangle)) \cdot \text{global phase}$$

   The client stores $d\cdot(x_0 \oplus x_1) \mod 2$ in register $\boldsymbol{\theta}_1$; the client sets $\boldsymbol{flag}$ to be fail if $d = 0^n$ and pass otherwise. The client keeps $\boldsymbol{\theta}, \boldsymbol{flag}, \boldsymbol{score}$ and disgards all the other registers (including $\boldsymbol{K}$ and d).

3. The client randomly samples $\varphi \leftarrow \{0, 1 \cdots 7\}$ and sends φ to the server.

   The server measures $\boldsymbol{q}$ on basis $|+_\varphi\rangle$ and $|+_{\varphi+4}\rangle$ and gets a measurement result r: $r = 0$ if the result is the former and $r = 1$ if the result is the latter. The server sends back the measurement results r.

   The client sets the $\boldsymbol{flag}$ register as follows:

   - If $\theta - \varphi = 0$, set $\boldsymbol{flag}$ to be pass if $r = 0$ and fail otherwise.
   - If $\theta - \varphi + 4 = 0$ (modulo 8), set $\boldsymbol{flag}$ to be pass if $r = 1$ and fail otherwise.
   - In other cases, simply set $\boldsymbol{flag}$ to be pass.

   The client sets the $\boldsymbol{score}$ register as follows:

- If $|\theta - \varphi| \leq 1$ (where the distance is modulo 8), set $\boldsymbol{score}$ to be win if $r = 0$ and lose otherwise.
- If $|\theta - \varphi + 4| \leq 1$ (where the distance is modulo 8), set $\boldsymbol{score}$ to be win if $r = 1$ and lose otherwise.
- If $|\theta - \varphi| = 2$, simply set $\boldsymbol{score}$ to be win.

$\mathsf{QFacComp}$ is defined as:

1. Same as the first step of $\mathsf{QFacTest}$.
2. Same as the first step of $\mathsf{QFacTest}$.

Note that the approximation error ϵ_0 is only the approximation error for the first step and not the approximation error for the whole protocol. This makes it easier to tune the parameters in later proofs and constructions.

The completeness and efficiency are from the protocol description. The honest server wins in $\mathsf{QFacTest}$ with probability $\frac{1}{2} + \frac{1}{2}\cos^2(\pi/8)$.

6.5.2 Analysis of the information-theoretic core

Let's formalize the set-up for the information-theoretic core of Protocol 13.

Set-up 12. Parameter: 1^n.

$\mathrm{OPT} = \frac{1}{2} + \frac{1}{2}\cos^2(\pi/8)$.

Consider the folowing registers, as used in the honest execution: client-side classical register $\boldsymbol{K} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$ where each of them holds n bits, client-side classical register $\boldsymbol{\theta}$ with value in $\{0, 1 \cdots 7\}$, server-side quantum register $\boldsymbol{q}, \boldsymbol{Q}$ which hold 1 qubit and n qubits each.

Part of the initial state is $\rho_{KP} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{K}} \otimes \mathcal{H}_{\boldsymbol{q},\boldsymbol{Q}})$ where ρ_{KP} is the target state of KP. For modeling the initial states in the malicious setting, consider server-side quantum register $\boldsymbol{S}$ and environment register $\boldsymbol{E}$. The initial states in the malicious setting that we consider could be described as $\rho_{KP} \otimes \rho_{\mathrm{ini}}$ where[21] $\rho_{\mathrm{ini}} \in \mathrm{D}(\mathcal{H}_{\boldsymbol{S}} \otimes \mathcal{H}_{\boldsymbol{E}})$.

We use $(\mathsf{QFacTest}_{\geq 2}, \mathsf{QFacComp}_{\geq 2})$ to denote the protocols that starts from the second step of Protocol 13, and use $\mathsf{QFacTest}_{\text{step 3}}$ to denote the third step of the $\mathsf{QFacTest}$ protocol.

We use $\mathsf{Adv}_{\text{step 2}}$ to denote the adversary's operation on the second step of the protocols; then corresponding to $\rho_{KP} \otimes \rho_{\mathrm{ini}}$ and $\mathsf{Adv}_{\text{step 2}}$, the joint state on the passing space after the step 2 of Protocol 13 could be denoted as

$$\underbrace{|\mathsf{pass}\rangle\langle\mathsf{pass}|}_{\boldsymbol{flag}} \sum_{\theta \in \{0,1\cdots 7\}} (\underbrace{|\theta\rangle\langle\theta|}_{\boldsymbol{\theta}} \otimes \underbrace{\rho_\theta}_{\boldsymbol{q},\boldsymbol{Q},\boldsymbol{S},\boldsymbol{E}})$$

In other words, ρ_θ is the component of the output state of the step 2 where $\boldsymbol{\theta}$ is in value θ and $\boldsymbol{flag}$ is in value pass.

Use $\mathsf{Adv}_{\text{step 3}} = (\mathsf{Adv}_\varphi)_{\varphi \in \{0,1\cdots 7\}}$ to denote the adversary's operation on the third step where Adv_φ corresponds to the client's question φ.

As discussed before, an important notion on the states after the second step is that the states $(\rho_\theta)_{\theta \in \{0,1\cdots 7\}}$ satisfies a condition called *(information-theoretic) basis blindness*, as follows.

[21] Previously we use ρ_0 for this part but below we need to define ρ_θ so we change the notation here to avoid conflicts.

Definition 6.3 ((Information-theoretic) basis blindness [25])**.** We say $(\rho_\theta)_{\theta\in\{0,1\cdots 7\}}$ has (Information-theoretic) basis blindness if $\forall\theta_{2,3}\in\{0,1,2,3\}$, $\frac{1}{2}(\rho_{\theta_{2,3}}+\rho_{\theta_{2,3}+4})\approx_{\mathsf{negl}(n)}\frac{1}{8}\sum_{\theta\in\{0,1\cdots 7\}}\rho_\theta$

Lemma 6.8. *$(\rho_\theta)_{\theta\in\{0,1\cdots 7\}}$ generated in Set-up 12 has information-theoretic basis blindness.*

Proof. This is equivalent to prove that any adversary working on $\rho_{KP}\otimes\rho_{\text{ini}}$ that does not have access to the client-side keys $\boldsymbol{K}$ could only predict $(b_0^{(1)}\oplus b_1^{(1)}, b_0^{(2)}\oplus b_1^{(2)})$ with probability $\frac{1}{4}+\mathsf{negl}(n)$. To calculate the probability that an operation predicts this information from $\rho_{KP}\otimes\rho_{\text{ini}}$, we could write ρ_{KP} as $(\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_0}+\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_1})\rho_{KP}(\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_0}+\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_1})$ to expand the expression for this probability, then this probability could be upper bounded by the sum of the following terms:

- The probability[22] that the operation operating on $\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_0}\rho_{KP}\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_0}$ could predict $(b_0^{(1)}\oplus b_1^{(1)}, b_0^{(2)}\oplus b_1^{(2)})$.
- The probability that the operation operating on $\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_1}\rho_{KP}\Pi^{\boldsymbol{Q}}_{=\boldsymbol{x}_1}$ could predict $(b_0^{(1)}\oplus b_1^{(1)}, b_0^{(2)}\oplus b_1^{(2)})$.
- The norm that the operation starting from $|0\rangle|x_0\rangle$ could predict x_1.
- The norm that the operation starting from $|0\rangle|x_1\rangle$ could predict x_0.

The last two terms are negligibly small and the first two terms sum to $\frac{1}{4}$, which completes the proof. □

Then by [29, 25] we have the optimality of OPT and the self-testing property for the test used in $\mathsf{QFacTest}_{\text{step 3}}$ given the condition that the input state satisfies information-theoretic basis blindness. Below we directly state it as the soundness of $(\mathsf{QFacTest}_{\geq 2}, \mathsf{QFacComp}_{\geq 2})$ under Set-up 12.

Theorem 6.9. *Under Set-up 12, $(\mathsf{QFacTest}_{\geq 2}, \mathsf{QFacComp}_{\geq 2})$ has $(\delta, \mathsf{poly}_1(\delta))$-optimal winning probability OPT and is $(\delta, \mathsf{poly}_2(\delta))$-sound for all δ.*

See Appendix A for details.

6.5.3 Compilation and amplification to preRSPV-with-score and RSPV

The remaining steps are similar to what we did in Section 6.3.3.

Theorem 6.10. *For Protocol 13, OPT is $(\delta-\epsilon_0, \mathsf{poly}_1(\delta)+\epsilon_0)$-optimal.*

Theorem 6.11. *Protocol 13 is $(\delta-\epsilon_0, \mathsf{poly}_2(\delta)+\epsilon_0)$-sound for all δ.*

Proof. The proofs of the two theorems above are by combining the soundness of KP (which leads to an approximation error ϵ_0) and Theorem 6.9. □

Below we give QFac, our RSPV for $|+_\theta\rangle$ states.

Protocol 14 (QFac)**.** Parameters: approximation error parameter $1^{1/\epsilon}$, security parameter 1^κ.

Output registers: client-side classical register $\boldsymbol{\theta}$ with value in $\{0, 1\cdots 7\}$, client-side classical register $\boldsymbol{flag}$ with value in $\mathsf{pass}, \mathsf{fail}$, server-side quantum register $\boldsymbol{q}$ holding a single qubit.

The protocol is the amplification procedure (Protocol 4) running on Protocol 13. The

[22] We mean the trace of the post-projection state onto the space that the event happens

condition $\epsilon_0 < \epsilon$, $\lambda < \frac{1}{6}\delta_0(\epsilon - \epsilon_0)$ in Protocol 4 is satisfied by taking the ϵ_0 in Protocol 13 sufficiently small and tuning δ in Theorem 6.11 to be suitably small.

The completeness, efficiency and soundness are from the properties of Protocol 13 and the preRSPV-to-RSPV amplification procedure (Section 3.5.2).

7 Classical Verification of Quantum Computations from Cryptographic Group Actions

In this section we combine our results with existing results to get new results on CVQC.

7.1 RSPV and CVQC from weak NTCF

Now we are ready to state the weak noisy trapdoor claw-free function (weak NTCF) assumption. We refer to Section 1.3.4 for an intuitive introduction. Below we give its formal definition. Note that there are also multiple styles for defining it; here we use the NTCF definition in [57] and adapt it to weak NTCF:

Definition 7.1 (Weak NTCF)**.** We define weak noisy trapdoor claw-free function (weak NTCF) as follows. It is parameterized by security parameter κ and correctness error μ and is defined to be a class of polynomial time algorithms as below. KeyGen is a sampling algorithm. Dec, CHK are deterministic algorithms. Eval is allowed to be a sampling algorithm. poly' is a polynomial that determines the the range size.

$$\mathsf{KeyGen}(1^{1/\mu}, 1^\kappa) \to (\mathrm{sk}, \mathrm{pk}),$$

$$\mathsf{Eval}_{\mathrm{pk}} : \{0,1\} \times \{0,1\}^\kappa \to \{0,1\}^{\mathsf{poly}'(\kappa)},$$

$$\mathsf{Dec}_{\mathrm{sk}} : \{0,1\} \times \{0,1\}^{\mathsf{poly}'(\kappa)} \to \{0,1\}^\kappa \cup \{\perp\},$$

$$\mathsf{CHK}_{\mathrm{pk}} : \{0,1\} \times \{0,1\}^\kappa \times \{0,1\}^{\mathsf{poly}'(\kappa)} \to \{\mathsf{true}, \mathsf{false}\}$$

And they satisfy the following properties:

- (Correctness)
  - (Noisy 2-to-1) For all possible $(\mathrm{sk}, \mathrm{pk})$ in the range of $\mathsf{KeyGen}(1^{1/\mu}, 1^\kappa)$ there exists a sub-normalized probability distribution $(p_y)_{y \in \{0,1\}^{\mathsf{poly}'(\kappa)}}$ that satisfies: for any y such that $p_y \neq 0$, $\forall b \in \{0,1\}$, there is $\mathsf{Dec}_{\mathrm{sk}}(b, y) \neq \perp$, and

$$\mathsf{Eval}_{\mathrm{pk}}(|+\rangle^{\otimes\kappa}) \approx_\mu \sum_{y:p_y \neq 0} \frac{1}{\sqrt{2}}(|\mathsf{Dec}_{\mathrm{sk}}(0,y)\rangle + |\mathsf{Dec}_{\mathrm{sk}}(1,y)\rangle) \otimes \sqrt{p_y}\,|y\rangle \tag{67}$$

  - (Correctness of CHK) For all possible $(\mathrm{sk}, \mathrm{pk})$ in the range of $\mathsf{KeyGen}(1^\kappa)$, $\forall x \in \{0,1\}^\kappa, \forall b \in \{0,1\}$:

$$\mathsf{CHK}_{\mathrm{pk}}(b, x, y) = \mathsf{true} \Leftrightarrow \mathsf{Dec}_{\mathrm{sk}}(b, y) = x$$

- (Claw-free) For any BQP adversary Adv,

$$\Pr\left[\begin{array}{l}(\mathrm{sk},\mathrm{pk})\leftarrow\mathsf{KeyGen}(1^{1/\mu},1^{\kappa}),\\ \mathsf{Adv}(\mathrm{pk},1^{1/\mu},1^{\kappa})\rightarrow(x_0,x_1,y):\quad x_0\neq\perp,x_1\neq\perp,x_0\neq x_1\\ \mathsf{Dec}_{\mathrm{sk}}(0,y)=x_0,\mathsf{Dec}_{\mathrm{sk}}(1,y)=x_1\end{array}\right]\leq\mathsf{negl}(\kappa)\tag{68}$$

The "noisy" comes from the fact that Eval is allowed to be a sampling algorithm and the "weak" comes from the error term in (67).

We refer [16, 18, 57] for more information on NTCF techniques.

The following theorem could be proved based on the results of [17]:

Theorem 7.1. *Assuming the existence of weak NTCF, there exists an RSPV protocol for BB84 states* $\mathsf{BB84}(1^{1/\mu},1^{1/\epsilon},1^{\kappa})$ *that is* μ*-complete and* ϵ*-sound for any* $\mu,\epsilon=1/\mathsf{poly}(\kappa)$.

Note that the correctness error in weak NTCF leads to a completeness error μ in the protocol.

[17] constructed a test of a qubit from NTCF without assuming the adaptive hardcore bit property, which could be adapted easily to weak NTCF. In Section 4.2 we already give the proof of Theorem 7.1 by tranlating test-of-a-qubit to RSPV for BB84.

Combining it with our results in Section 6, we could prove that all the protocols in Section 6 could be constructed from weak NTCF. Especially, we have:

Theorem 7.2. *Assuming the existence of weak NTCF, there exists an RSPV for state family* $\{\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle+|1\rangle|x_1\rangle),x_0,x_1\in\{0,1\}^n\}$ *that is* μ*-complete and* ϵ*-sound for any* $\mu,\epsilon=1/\mathsf{poly}(\kappa)$..

Theorem 7.3. *Assuming the existence of weak NTCF, there exists an RSPV for state family* $\{|+_\theta\rangle:=\frac{1}{\sqrt{2}}(|0\rangle+e^{\mathrm{i}\pi\theta/4}|1\rangle),\theta\in\{0,1,2\cdots7\}\}$ *that is* μ*-complete and* ϵ*-sound for any* $\mu,\epsilon=1/\mathsf{poly}(\kappa)$..

Then recall that by [27], if the client is allowed to prepare and send lots of $|+_\theta\rangle$ states before the protocol, both parties could do quantum computation verification. As discussed before, this quantum communication could be compiled to classical communication using an RSPV for $|+_\theta\rangle$ states. Finally we note that although RSPV-based compilation introduces some non-negligible completeness error and soundness error, in CVQC problem these errors could be amplified to be exponentially small by sequential repetition as long as there is a significant completeness-soundness gap. Thus we have:

Theorem 7.4. *Assuming the existence of weak NTCF, there exists a CVQC protocol.*

7.2 CVQC from Assumptions on Group Actions

Finally we review the results in [5], which constructs weak TCF (which is stricter than weak NTCF) from cryptographic group actions (for example, isogeny).

Assumption 1 (Repeat of [5])**.** *Assume the extended linear hidden shift assumption holds for some effective group action.*

Theorem 7.5 (Repeat of [5])**.** *There exists a weak TCF assuming Assumption 1*

We refer to [5] for details.

Combining Theorem 7.4, 7.5 we have:

Corollary 7.6. *Assuming 1, there exists a CVQC protocol.*

A Proof of Theorem 6.9

We note that test in Definition 4.1 corresponds exactly to the case of $\varphi \in \{0,2\}, \theta \in \{1,3,5,7\}$ in $\mathsf{QFacTest}_{\text{step 3}}$. In $\mathsf{QFacTest}_{\text{step 3}}$ the cases where the score could be nontrivially accumulated could be decomposed to this test rotated to different angles (that is, $\varphi \in \{0,2\}$, $\varphi \in \{1,3\}$, $\varphi \in \{4,6\}$, $\varphi \in \{5,7\}$). In trivial cases (that is, $|\theta - \varphi| \in \{0,2,4\}$) the value of $\boldsymbol{score}$ is simply win with probability close to 1 (under the condition that the server passes with probability close to 1). Thus we have $\mathrm{OPT} = \frac{1}{2} + \frac{1}{2}\cos^2(\pi/8)$ as in Set-up 12. The soundness property could be proved in the way described in the beginning of Section 4, proof of Lemma 4.8 and Section 4.3 (here we do not need to consider computational indistinguishability so lemmas based on statistical indistinguishability, like Lemma 4.10, 4.6, are sufficient).

B CVQC from Cryptographic Group Actions via [44]

In this section we sketch (without formal proofs) another approach for achieving CVQC from weak NTCF, based on the results in [44].[23]

[44] gives a result on how to achieve verification of quantum computation in a model where trusted center sends BB84 states to the server and sends the classical description to the client. By replacing the state distribution step by callings to the RSPV protocols, we get a CVQC protocol. Note that it seems that RSPV for random BB84 states is not sufficient here; what we need is an RSPV for BB84 states where the client could choose which state to prepare, as discussed in Section 3.3.1. Intuitively this primitive could be constructed by repeating the RSPV-for-BB84 for many times and letting the client to choose the desired state. Finally, as discussed in Section 4 and Theorem 7.1, RSPV for BB84 states could be constructed from cryptographic group actions, which completes the proof.

References

[1] Discussion with thomas vidick on details of paper "computationally secure and composable remote state preparation", 2024.

[2] Dorit Aharonov, Michael Ben-or, and Elad Eban. Interactive proofs for quantum computations, 2017.

[3] Gorjan Alagic, Andrew M. Childs, Alex Bredariol Grilo, and Shih-Han Hung. Non-interactive classical verification of quantum computation. In *IACR Cryptol. ePrint Arch.*, 2020.

[4] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II*, page 411–439, Berlin, Heidelberg, 2020. Springer-Verlag.

[5] Navid Alamati, Giulio Malavolta, and Ahmadreza Rahimi. Candidate trapdoor claw-free functions from group actions with applications to quantum protocols. In *Theory of Cryptography:*

[23] We thank anonymous reviewers for pointing out this approach.

20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I, page 266–293, Berlin, Heidelberg, 2022. Springer-Verlag.

[6] Alexander Poremba Alexandru Gheorghiu, Tony Merger. Quantum cryptography with classical communication: parallel remote state preparation for copy-protection, verification, and more. 2022.

[7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[8] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[9] Kaniuar Bacho, Alexander Kulpe, Giulio Malavolta, Simon Schmidt, and Michael Walter. Compiled nonlocal games from any trapdoor claw-free function. *IACR Cryptol. ePrint Arch.*, 2024:1829, 2024.

[10] Christian Badertscher, Alexandru Cojocaru, Léo Colisson, Elham Kashefi, Dominik Leichtle, Atul Mantri, and Petros Wallden. Security limitations of classical-client delegated quantum computing. In *Advances in Cryptology - ASIACRYPT 2020*, pages 667–696. Springer International Publishing, 2020.

[11] James Bartusek. Secure quantum computation with classical communication. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 1–30, Cham, 2021. Springer International Publishing.

[12] James Bartusek and Dakshita Khurana. On the Power of Oblivious State Preparation. 11 2024.

[13] Charles H. Bennett, David P. DiVincenzo, Peter W. Shor, John A. Smolin, Barbara M. Terhal, and William K. Wootters. Remote state preparation. *Phys. Rev. Lett.*, 87:077902, Jul 2001.

[14] Dolev Bluvstein et al. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456, 2022.

[15] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 41–69, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 320–331, 2018.

[17] Zvika Brakerski, Alexandru Gheorghiu, Gregory D. Kahanamoku-Meyer, Eitan Porat, and Thomas Vidick. Simple tests of quantumness also certify qubits. 2023.

[18] Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. Simpler proofs of quantumness, 05 2020.

[19] Gilles Brassard, Claude Crepeau, Dominic Mayers, and Louis Salvail. A Brief review on the impossibility of quantum bit commitment. 12 1997.

[20] Anne Broadbent. How to verify a quantum computation. *Theory of Computing*, 14, 09 2015.

[21] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 517–526, Washington, DC, USA, 2009. IEEE Computer Society.

[22] Ran Canetti. Universally composable security. *J. ACM*, 67(5), sep 2020.

[23] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. Csidh: An efficient post-quantum commutative group action. In *Advances in Cryptology – ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III*, page 395–427, Berlin, Heidelberg, 2018. Springer-Verlag.

[24] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. Classical verification of quantum computations with efficient verifier. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, pages 181–206, Cham, 2020. Springer International Publishing.

[25] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. Qfactory: Classically-instructed remote secret qubits preparation. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 615–645. Springer, 2019.

[26] Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum homomorphic encryption for polynomial-sized circuits. *IACR Cryptol. ePrint Arch.*, 2016:559, 2016.

[27] Samuele Ferracin, Theodoros Kapourniotis, and Animesh Datta. Reducing resources for verification of quantum computations. *Phys. Rev. A*, 98:022323, Aug 2018.

[28] Joseph F. Fitzsimons, Michal Hajdušek, and Tomoyuki Morimae. Post hoc verification of quantum computation. *Phys. Rev. Lett.*, 120:040501, Jan 2018.

[29] Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1024–1033. IEEE Computer Society, 2019.

[30] Alex B. Grilo. A Simple Protocol for Verifiable Delegation of Quantum Computation in One Round. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[31] Aparna Gupte and Vinod Vaikuntanathan. How to Construct Quantum FHE, Generically. 6 2024.

[32] Qi Zhao Honghao Fu, Daochen Wang. Computational self-testing of multi-qubit states and measurements. 2022.

[33] Zhengfeng Ji. Classical verification of quantum proofs. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 885–898, New York, NY, USA, 2016. Association for Computing Machinery.

[34] Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. Mip* = re. *Commun. ACM*, 64(11):131–138, oct 2021.

[35] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Lisa Yang. Quantum advantage from any non-local game. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1617–1628, New York, NY, USA, 2023. Association for Computing Machinery.

[36] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[37] Yi-Kai Liu, Zhengfeng Ji, and Fang Song. Pseudorandom quantum states. Number 10993. Crypto 2018, Santa Barbara, CA, 2018-08-19 2018.

[38] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 332–338. IEEE Computer Society, 2018.

[39] Urmila Mahadev. Classical verification of quantum computations. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 259–267. IEEE Computer Society, 2018.

[40] Ueli Maurer and Renato Renner. Abstract cryptography. In *International Conference on Supercomputing*, 2011.

[41] Dominic Mayers and Andrew Yao. Self testing quantum apparatus. *Quantum Info. Comput.*, 4(4):273–286, jul 2004.

[42] Tony Metger and Thomas Vidick. Self-testing of a single quantum device under computational assumptions. In *ITCS*, 2021.

[43] Akihiro Mizutani, Yuki Takeuchi, Ryo Hiromasa, Yusuke Aikawa, and Seiichiro Tani. Computational self-testing for entangled magic states. *Phys. Rev. A*, 106(1):L010601, 2022.

[44] Tomoyuki Morimae. Information-theoretically-sound non-interactive classical verification of quantum computing with trusted center, 03 2020.

[45] Anand Natarajan and Tina Zhang. Bounding the quantum value of compiled nonlocal games: from chsh to bqp verification. 2023.

[46] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[47] Arun Kumar Pati. Minimum cbits required to transmit a qubit. *Phys. Rev. A*, 63:014320, 2001.

[48] Sandu Popescu and Daniel Rohrlich. Which states violate bell's inequality maximally? *Physics Letters A*, 169(6):411–414, 1992.

[49] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[50] Ben Reichardt, Falk Unger, and Umesh V. Vazirani. Classical command of quantum systems. *Nature*, 496:456–460, 2013.

[51] Gregory Rosenthal and Henry S. Yuen. Interactive proofs for synthesizing quantum states and unitaries. In *Information Technology Convergence and Services*, 2021.

[52] Omri Shmueli. Public-key quantum money with a classical bank. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 790–803, New York, NY, USA, 2022. Association for Computing Machinery.

[53] Stephen J. Summers and R. Werner. Maximal Violation of Bell's Inequalities Is Generic in Quantum Field Theory. *Commun. Math. Phys.*, 110:247–259, 1987.

[54] Thomas Vidick. Lecture notes: Interactions with quantum devices. Course FSMP, Fall'20, 2020. `http://users.cms.caltech.edu/~vidick/teaching/fsmp/fsmp.pdf` pages 23-31.

[55] John Watrous. The theory of quantum information. 2018.

[56] Wikipedia contributors. Azuma's inequality — Wikipedia, the free encyclopedia, 2024. [Online; accessed 30-September-2024].

[57] J. Zhang. Classical verification of quantum computations in linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society.

[58] Jiayu Zhang. Delegating quantum computation in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 30–60, Cham, 2019. Springer International Publishing.

[59] Jiayu Zhang. *Succinct Blind Quantum Computation Using a Random Oracle*, page 1370–1383. Association for Computing Machinery, New York, NY, USA, 2021.

[60] Qingling Zhu, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lian-Chen Han, Linyin Hong, Heliang Huang, Yongheng Huo, Liping Li, Na Li, Shaowei Li, Yuan Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong-Bo Su, Lihua Sun,

Liangyuan Wang, Shiyu Wang, Dachao Wu, Yulin Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jian Hua Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, You-Wei Zhao, Liang Zhou, Chaoyang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Science bulletin*, 67 3:240–245, 2021.